Semi-automatic Extraction of Cross-Table Data from a Set of Spreadsheets

Alaaeddin Swidan(✉) and Felienne Hermans

Delft University of Technology, Delft, Netherlands
{Alaaeddin.Swidan,F.F.J.Hermans}@tudelft.nl

Abstract. Spreadsheets are widely used in companies. End-users often value the high degree of flexibility and freedom spreadsheets provide. However, these features lead to the development of a variety of data forms inside spreadsheets. A *cross-table* is one of these forms of data. A cross-table is defined as a rectangular form of data, which expresses the relations between a set of objects and a set of attributes. Cross-tables are common in spreadsheets: our exploratory analysis found that more than 3.42% of spreadsheets in an industrial open dataset include at least one cross-table. However, current software tools provide no support to analyze data in cross-tables. To address this, we presents a semi-automatic approach to extract cross-table data from a set of spreadsheets, and transform them to a relational table form. We evaluate our approach in a case study, on a set of 333 spreadsheets with 2,801 worksheets. The results show that the approach is successful in extracting over 92% of the data inside the targeted cross-tables. Further, we interview two users of the spreadsheets working in the company; they confirmed the approach is beneficial and provides correct results.

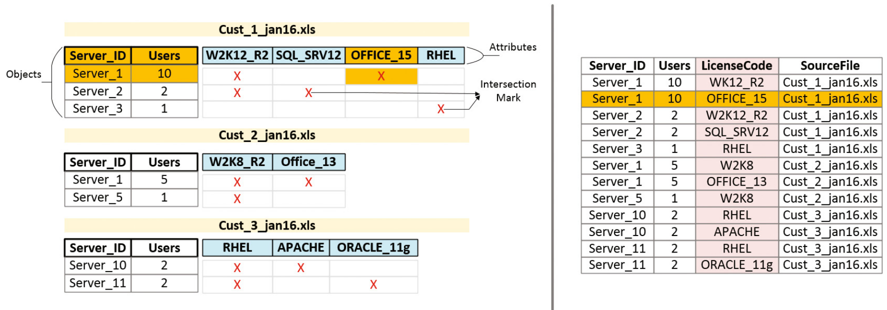
1 Introduction

Spreadsheets are used extensively across various domains of expertise, to perform a wide range of tasks [1–4]. In particular, spreadsheets are often used for data analysis and management [5]. In addition to the powerful set of functionalities, the end-users appreciate the flexibility and freedom provided by spreadsheets [6]. This leads, however, to a variety of forms in which data is represented inside a worksheet. One of the special forms of data found in spreadsheets is the *cross-table*. A cross-table in general aims to represent a relation I. It consists of G rows and M columns, and can be defined as:

> *A rectangular table with one row for each object and one column for each attribute, having a cross in the intersection of row g with column m iff $(g,m) \in I$. [7]*

Cross-tables, an example of which is shown in Fig. 1, are common in spreadsheets. To explore this, we manually investigated more than 1,500 spreadsheets from the industrial dataset Enron [8]. Our investigation revealed that 552 spreadsheets, or 3.42% of the dataset, include at least one cross-table (Sect. 3). As an

S. Barbosa et al. (Eds.): IS-EUD 2017, LNCS 10303, pp. 84–99, 2017.
DOI: 10.1007/978-3-319-58735-6_6


Objects
Attributes
Intersection Mark

(a)

Cust_1_jan16.xls

Server_ID	Users	W2K12_R2	SQL_SRV12	OFFICE_15	RHEL
Server_1	10	X		X	
Server_2	2	X	X		
Server_3	1				X

Cust_2_jan16.xls

Server_ID	Users	W2K8_R2	Office_13
Server_1	5	X	X
Server_5	1	X	

Cust_3_jan16.xls

Server_ID	Users	RHEL	APACHE	ORACLE_11g
Server_10	2	X	X	
Server_11	2	X		X

(b)

Server_ID	Users	LicenseCode	SourceFile
Server_1	10	WK12_R2	Cust_1_jan16.xls
Server_1	10	OFFICE_15	Cust_1_jan16.xls
Server_2	2	W2K12_R2	Cust_1_jan16.xls
Server_2	2	SQL_SRV12	Cust_1_jan16.xls
Server_3	1	RHEL	Cust_1_jan16.xls
Server_1	5	W2K8	Cust_2_jan16.xls
Server_1	5	OFFICE_13	Cust_2_jan16.xls
Server_5	1	W2K8	Cust_2_jan16.xls
Server_10	2	RHEL	Cust_3_jan16.xls
Server_10	2	APACHE	Cust_3_jan16.xls
Server_11	2	RHEL	Cust_3_jan16.xls
Server_11	2	ORACLE_11g	Cust_3_jan16.xls

Fig. 1. An example showing related data presented in three separate spreadsheets using cross-tables in (a). (b) shows the equivalent data migrated into one relational table.

example of a cross-table in a context, consider Fig. 1a. The example is based on our case study at Solvinity[1], which provides virtual environments and IT services. In the example, the finance department uses spreadsheets to record the data related to third-party licenses sold as part of each virtual environment, in a cross-table. In one scenario, an internal auditor, let us call him Bas, aims to answer management questions such as: do actual billed licenses conform to the specification of the providing third-party? do the actual-billing data on licenses conform to the actual software license-configuration (eg. number of users) on the servers? To answer these, and similar related questions, data analysis on the related cross-tables, which are included in separate files should be carried out.

As a human, Bas can easily read a cross-table. For example, in Fig. 1a, he can read that for *"Cust_1_jan16.xls"*, *"Licence: OFFICE_15"* is related to *"Server_ID:Server_1, Users:10"*. Bas does this *mapping* through a sequence of mental operations, which include linking and relating the two-separate data sets: the objects and the attributes. In addition, Bas incorporates his domain knowledge to recognize that the attribute *"OFFICE_15"* is a data value of type *"License"*.

In reality, Bas is *transforming* each X-marked relation in the cross-table, into one *"record"* in the relational table in Fig. 1b. Therefore, if the data was originally formed in a relational table form, the user would be spared from performing the conversion steps, and focuses on the analysis part of the task. In addition, data in a relational table allows the user to leverage a wide-set of analysis tools and visualizations, including Excel itself. These tools are designed to work best with strictly-formed relational tables. In our case, the user can perform the transformation from a cross-table to a relational-table form manually. However, as the number of cross-tables increases, the task becomes time-consuming, tedious and unrealistic. The previous scenario describes a twofold problem for the end-users:

[1] https://www.solvinity.com/.

(a) The data has a special form that is not supported by current software tools.
(b) The data should be consolidated from multiple spreadsheets into one relational table, to be used in other analysis software.

To address this problem, we propose a semi-automatic approach that extracts cross-table data from a set of spreadsheets, and converts them into one (denormalized) relational table. Our approach expands upon previous studies that managed to extract data and hierarchy of relational tables in spreadsheets [9]. In addition to the user input, we incorporate an automatic transformation algorithm that is suitable for a cross-table. Our approach aggregates the data from all the transformed cross-tables, and generates one relational table. The final output is the structure and data of the desired table, encoded in an SQL script.

We perform a mixed method evaluation of our approach, on a set of 333 spreadsheet with 2,801 worksheets found in the company dataset. First, we quantitatively analyze the approach performance in extracting cross-table data. Subsequently, we interview two frequent users of the spreadsheets, to manually validate the approach generated data on a subset of 30 spreadsheets.

The contributions of the paper are:

1. An exploratory study on the incidence of cross-table in spreadsheets. For this, we manually examined a subset of more than 1,500 Enron spreadsheets.
2. A semi-automatic approach that identifies, transforms cross-table data from multiple spreadsheets, aggregating them into one relational table.
3. An industrial evaluation through a case study.

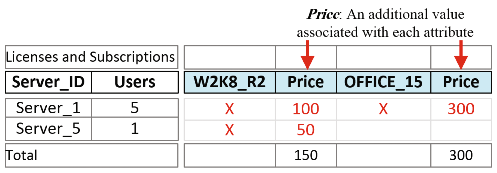
2 Background

Before describing the extraction approach, we provide a brief overview of the preliminaries this paper builds upon.

2.1 Cross-Tables

There are two types of cross-table: single-valued and many-valued [7]. A single-valued cross-table, shown in Fig. 1a, follows the general definition, where a binary relation I between a row-based object g, and a column-based attribute m is marked using the *intersection cell.* A many-valued cross-table, shown in Fig. 2, adds a fourth dimension to the definition, which is the set of data values W_m that are associated with each attribute m.

Cross-tables are sometimes called *"matrices"* [10,11]. We find that *"cross-table"* is the frequently used term [7,12,13], so we use it throughout this paper. A cross-table, as a word, suggests the usage of a *"cross"*, or the *"X"* character in *the intersection cell* which indicates the relationship between an object and its attributes. In practice, however, users are not restricted to use the cross, and they can use other marks in the intersection cell, which can be a specific character, a digit or a shape such as a circle or a triangle.


Price: An additional value associated with each attribute

Licenses and Subscriptions					
Server_ID	**Users**	**W2K8_R2**	**Price**	**OFFICE_15**	**Price**
Server_1	5	X	100	X	300
Server_5	1	X	50		
Total			150		300

Fig. 2. An example of a many-valued cross-table. The column *Price* represents an additional data value associated with each attribute in the adjacent column.

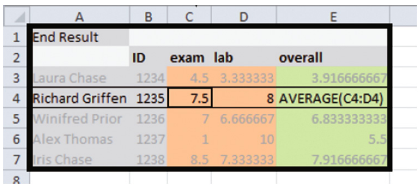
In spreadsheets, cross-tables may be developed for various tasks. For a simple example, a spreadsheet with a cross-table can be used by employees (objects) to reserver meeting rooms (attributes). A more complex example is used in *"software requirement traceability"* [14], where the user follows up the progress of customer requirements (objects) against system components under development (attributes). In Sect. 3, we quantify the popularity of using cross-tables in spreadsheets in a business context.

2.2 Extraction and Transformation of Spreadsheet Data

In this subsection, we present an overview of previous research work, and software products which targeted the extraction of data from spreadsheets for various goals. One example from previous research is the GyroSAT algorithm, which was developed by Hermans *et al.* [9] to help end-users comprehend the structure of spreadsheets. For that purpose, the GyroSAT visualized the hierarchy and relations between blocks of cells, and worksheets, in a spreadsheet. Another research work is UCheck [15], which extracts header and unit information from all cells, as a means to detecting potential errors in spreadsheets. In software, database and data analysis systems, such as Microsoft SQL Server, consider spreadsheets as a data source, and the user is allowed to import data from a spreadsheet. The aforementioned tools and software products, however, work best with relational tables in spreadsheets. For cross-table data, the user should transform the data inside the spreadsheet into a relational table before being able to use these software tools.

Despite being designed to work on relational tables, some components from previous research can be re-used to extract other forms of data, specifically the cross-table. In our case, our approach expands upon the GyroSAT algorithm, which we choose because: First, the GyroSAT is able to identify a rectangle of adjacent cells, which is called a data block. A cross-table, in essence, is a rectangular area of cells, which makes the identification of a data block is the logical first step in extracting data from a cross-table. Second, a recent study benchmarked the GyroSAT and UCheck performances against the selection made by human users of spreadsheets [16]. The study reveals that GyroSAT has better performance in identifying cell types, which is a prerequisite for the data block identification. Following is the description of the two components that are implemented in GyroSAT, and are used in our approach.

- **Cell Classification:** The GyroSAT approach has four classifications of a cell depending on its content and relations. A *"formula"* type is given to a cell which contains a formula inside. The cells which are referred by the formulas are considered *"data"*. If the cell has an empty content, it is given the *"empty"* type, otherwise the cell is given the *"label"* type. Label cells contain data that are not part of any calculation, thus it is supposed they label other data.
- **Data Blocks:** Using the cell classification, and through a cell-to-cell search algorithm, the approach identifies a rectangular area which includes physically-adjacent cells. The data block, as shown in Fig. 3 is not expanded when each of the current four corners is surrounded by empty cells, from the outside of the block, in the diagonal, vertical and horizontal directions. In Fig. 3 for example, the corner cell A7 is not expanded since the outside neighbor cells A8 and B8 are both empty.



	A	B	C	D	E
1	End Result				
2		ID	exam	lab	overall
3	Laura Chase	1234	4.5	3.333333	3.916666667
4	Richard Griffen	1235	7.5	8	AVERAGE(C4:D4)
5	Winifred Prior	1236	7	6.666667	6.833333333
6	Alex Thomas	1237	1	10	5.5
7	Iris Chase	1238	8.5	7.333333	7.916666667
8					

Fig. 3. A data block including the cells A1:E7 [6]

2.3 Motivation

As the spreadsheet usage grows in organizations, the data in spreadsheets becomes more important for decision making processes [4]. When valuable data are stored in spreadsheets, users such as executives, managers or auditors, become more interested in acquiring knowledge out of the spreadsheets. Many software tools might aid these users in understanding their business, and taking actions accordingly. For instance, Business Intelligence (BI) tools, such as Tableau[2], provide reports and visuals. Traditional database systems provide the ability to perform user-specified queries. In addition, data integration tools, such as Microsoft SSIS[3], allow to compare and aggregate data from different sources.

Despite considering spreadsheets as a potential data source, these software tools require the data inside these spreadsheets to be in a strict relational table form [5]. If this is the case, the user can directly leverage the powerful analysis features provided by these tools. However, when the data is in other forms, such as a cross-table, the users are expected to transform the data beforehand. The transformation can be done manually, though it is tedious and time-consuming especially when the data involved is large in scale. In fact, when working on

[2] https://www.tableau.com/.
[3] https://msdn.microsoft.com/en-us/library/ms141026.aspx.

spreadsheet data, a recent survey reveals that users prefer less manual processing and more automatic solutions [4].

In this paper we aim at providing an approach that extracts and transforms data formed in cross-tables, from multiple spreadsheets, into one integrated relational table, with minimum user efforts. The output of our approach, encoded in standard SQL, can be used to further analyze the data by many analysis tools, not limited to the ones described earlier.

3 An Exploratory Analysis of Cross-Tables in Industry

Before we address the extraction of cross-table data, it is important to realize how much widespread cross-tables are in spreadsheets. To achieve this, we analyze the Enron spreadsheet dataset in search for the usage of cross-tables. The Enron dataset [8] provides researchers with the opportunity to study a large number of spreadsheets, compared to other known corpora, which were developed and used in an industrial context.

Setup: We follow a two-step approach to identify a cross-table in Enron spreadsheets. **First**, we automatically identify all of the spreadsheets which include at least one value of a particular intersection cell. Subsequently, the resulted subset includes the spreadsheet candidates which may have one form of a cross-table inside. To filter out the actual cross-tables from other structures of data, our **second step** consists of manually analyzing the resulted subset of spreadsheets. These two steps are based on self-defined criteria which we detail next.

Criteria: For the first step, criteria include the values of intersection cells that the automatic search will use. We recall from Sect. 2.1 that an intersection cell may include many possible values such as a single character, a digit or a shape. Since our analysis is exploratory, and is carried out without previous domain knowledge of the content of Enron spreadsheets, we decide to limit the values we search for to two values:

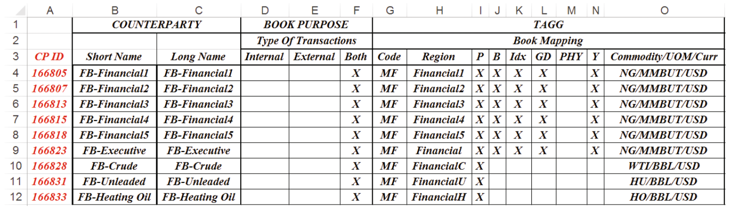
X: Mostly used by users in cross-tables.
Y: An abbreviation of Yes, indicating a positive selection of an attribute.

For the second step, the manual verification of the existence of a cross-table, our criteria follow the general definition of a cross-table, examples shown in Fig. 1a. For this, an actual cross-table is identified if it has the following three criteria: (1) The *"objects"* and *"attributes"* are visually recognizable as two separate sets of data. (2) There are one or more intersection cells found in the area between objects and attributes. (3) The value of the intersection cell is used as an indicator to a relation, not as a data value by itself.

Results: The summary of our analysis is presented in Table 1. Initially, Enron dataset consists of 16,160 spreadsheets. The first step of the analysis generated a subset of 1,524 spreadsheets. After the manual verification in the second step, we established that 552 spreadsheets in the subset contain at least one form of a

Table 1. Results of the analysis performed on the Enron dataset (16,161 spreadsheets), to identify cross-tables structures in the spreadsheets.

Interaction mark (Case Insensitive)	Spreadsheets with a *"candidate"* cross-table	Spreadsheets with a *"verified"* cross-table
X	1,177	542
Y	347	10
Total	1,524	552



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		*COUNTERPARTY*		*BOOK PURPOSE*			*TAGG*								
2				*Type Of Transactions*			*Book Mapping*								
3	*CP ID*	*Short Name*	*Long Name*	*Internal*	*External*	*Both*	*Code*	*Region*	*P*	*B*	*Idx*	*GD*	*PHY*	*Y*	*Commodity/UOM/Curr*
4	*166805*	*FB-Financial1*	*FB-Financial1*			*X*	*MF*	*Financial1*	*X*	*X*	*X*	*X*		*X*	*NG/MMBUT/USD*
5	*166807*	*FB-Financial2*	*FB-Financial2*			*X*	*MF*	*Financial2*	*X*	*X*	*X*	*X*		*X*	*NG/MMBUT/USD*
6	*166813*	*FB-Financial3*	*FB-Financial3*			*X*	*MF*	*Financial3*	*X*	*X*	*X*	*X*		*X*	*NG/MMBUT/USD*
7	*166815*	*FB-Financial4*	*FB-Financial4*			*X*	*MF*	*Financial4*	*X*	*X*	*X*	*X*		*X*	*NG/MMBUT/USD*
8	*166818*	*FB-Financial5*	*FB-Financial5*			*X*	*MF*	*Financial5*	*X*	*X*	*X*	*X*		*X*	*NG/MMBUT/USD*
9	*166823*	*FB-Executive*	*FB-Executive*			*X*	*MF*	*Financial*	*X*	*X*	*X*	*X*		*X*	*NG/MMBUT/USD*
10	*166828*	*FB-Crude*	*FB-Crude*			*X*	*MF*	*FinancialC*	*X*						*WTI/BBL/USD*
11	*166831*	*FB-Unleaded*	*FB-Unleaded*			*X*	*MF*	*FinancialU*	*X*						*HU/BBL/USD*
12	*166833*	*FB-Heating Oil*	*FB-Heating Oil*			*X*	*MF*	*FinancialH*	*X*						*HO/BBL/USD*

Fig. 4. A complex cross-table found in Enron dataset: objects are in two columns, attributes are in multiple (hierarchical) rows.

cross-table. In other words, at least 3.42% of spreadsheets in the Enron dataset include one cross-table form. Figure 4 shows one of the cross-tables found in the Enron dataset.

In addition to identifying cross-tables, the analysis shows that particular cross-tables are context-related and found in a subset of spreadsheet files. For example, cross-tables similar to the one shown in Fig. 4 are developed and used frequently in more than 100 spreadsheets in Enron dataset. The content of the cross-table in this example suggests that it was part of a business process of risk book requests. It mentions an information system called *"ERMS"*, standing for Enron Risk Management Services. However, identifying the exact business context remains difficult without the user domain knowledge.

4 Approach

Our approach aims to extract cross-tables data from a set of related spreadsheets. The approach follows four steps to achieve its aim, as summarized in Fig. 5.

- Step 1 - Identify the dimensions of a *"potential"* cross-table: For a given worksheet, the approach uses an algorithm based on the GyroSAT, in addition to keywords and configuration parameters specified by the end-user, to decide three attributes of a cross-table rectangle: the minimum row and column, the maximum row and column, and the header row number.
- Step 2 - Transform the cross-table into key-value data tuples: For each detected cross-table, the approach perform transformations on the cross-table to construct the related key-value tuples.

– Step 3 - Decide the common keys from the transformed cross-tables: Our approach considers all the cross-tables identified and transformed from multiple spreadsheet files, and subsequently decides the common keys in the data tuples. These keys will be the column names in the target relational table.
– Step 4 - Generate the output file: The output file, encoded as an SQL script, contains the statements to create the structure, and fill the data of the target relational table.

Prior to the application of the approach, the user should provide the following parameters and textual keywords:

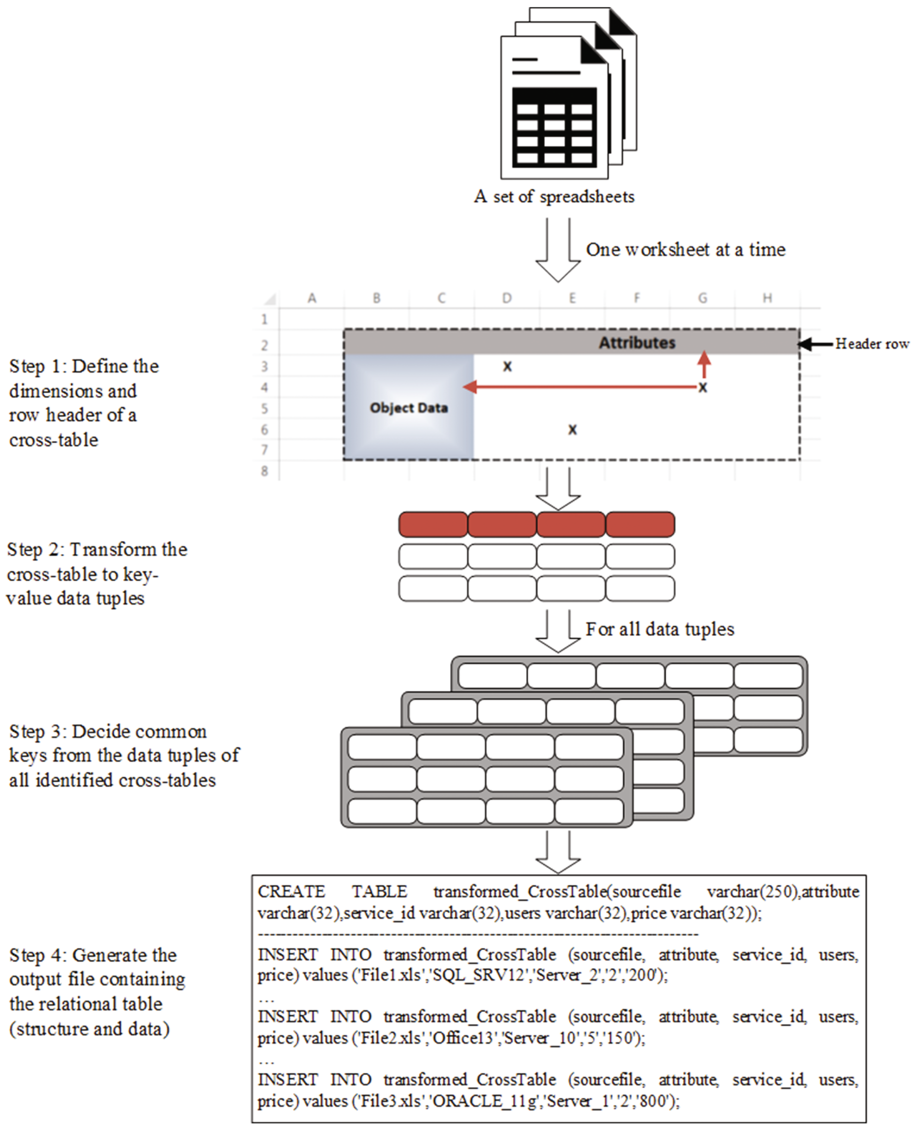
(a) Keywords which can be found in the upper and lower left corners of the cross-table. For example, in Fig. 2, the user may supply {upper: *"licenses"*, upper: *"subscription"*, lower: *"total"*}.
(b) Intersection mark: the value used to indicate an intersection between a cross-table object and its attributes. In Fig. 2, this value is *"X"*.
(c) Number of empty columns to skip: a parameter used in the identification of the cross-table dimensions.
(d) Has adjacent data: A boolean value indicating whether an attribute has related data in the adjacent column to its right. In fact, this value indicates whether the cross-table is of type single-valued or many-valued. Following, we describe in details the algorithm and components of the four steps.

Step 1 - Identify the dimensions and header row of a *"potential"* cross-table:

(a) Dimensions: The user-specified keywords aid our approach to define the top left corner (minimum row, minimum column) and the bottom left corner (maximum row, minimum column) of a cross-table. To identify the maximum column of the cross-table, our approach performs a ***data-block-based*** search, taking advantage of the data block definition in GyroSAT [17]. First, an initial *"maximum column"* value is retrieved from the first identified data block. Thereafter, the maximum column value is recursively updated whenever a new data block is found under the condition that the data block position is within the values of the three known dimensions (minimum row and column, and maximum row). Once no more data blocks are found, we force an extension of the rectangular area to find adjacent data blocks to the right. The extension allows to skip a number of columns, identified by the end-user. Thereafter, the search locates data blocks starting in the extended area. This is performed because cross-tables may contain empty columns, and an empty column forces the GyroSAT to stop expanding a data block horizontally. We apply the extension action, until no further data blocks are found even with the forced extension.

(b) Header row: The header row contains two types of data:

– Label values, which categorize and describe the data cells underneath.
– Data values, when the cells below the header are the intersection cells.


A set of spreadsheets
One worksheet at a time
Step 1: Define the dimensions and row header of a cross-table
A B C D E F G H
1 2 3 4 5 6 7 8
Attributes
Header row
Object Data
x
x
x
Step 2: Transform the cross-table to key-value data tuples
For all data tuples
Step 3: Decide common keys from the data tuples of all identified cross-tables
Step 4: Generate the output file containing the relational table (structure and data)

```
CREATE TABLE transformed_CrossTable(sourcefile varchar(250),attribute
varchar(32),service_id varchar(32),users varchar(32),price varchar(32));
--------------------------------------------------------------------------------
INSERT INTO transformed_CrossTable (sourcefile, attribute, service_id, users,
price) values ('File1.xls','SQL_SRV12','Server_2','2','200');
...
INSERT INTO transformed_CrossTable (sourcefile, attribute, service_id, users,
price) values ('File2.xls','Office13','Server_10','5','150');
...
INSERT INTO transformed_CrossTable (sourcefile, attribute, service_id, users,
price) values ('File3.xls','ORACLE_11g','Server_1','2','800');
```

Fig. 5. Approach summary showing the steps to extract and transform cross-table data from multiple spreadsheets to one relational table encoded in SQL.

Using the cell classification concept of the GyroSAT, the header row is found as *the row with the largest number of cells of type label, within the dimensions already found*. The search is performed from top to bottom.

The output of this step is the dimensions and header row of a cross-table. When the cross-table cannot be identified; the keywords are not matched or the intersection value is not found, then the approach moves to another worksheet.

Step 2 - Transform a cross-table: After identifying the targeted cross-table, data transformations are performed on the cells which lie within the cross-table's dimensions. The transformation starts by identifying the intersection marks, *"X"* in our case. For each intersection mark, the transformation builds a key-value data tuple. Specific transformation actions are followed to generate a data tuple. Following is the description of these actions depending on the type of cell found in the intersection row.

- Intersection: For the intersection cell itself, the data is found in the header. The key which describes this data value is implicit. As a resolution, we introduce the word *"Attribute"* as the key for the intersection cell data. The key-value pair
  $\boxed{\textit{“Attribute”, ValueIn(Intersection column, Header row)}}$
  For Fig. 2, the top-left intersection mark has the key-value pair = (*"Attribute"*, *"W2K8_R2"*).
- Adjacent data: When the cross-table is a many-valued cross-table, a data cell is always adjacent to the right of an intersection cell. In Fig. 2, the cells in the *"Price"* column are always adjacent to the right of the intersection cells. The user defines the boolean parameter *"HasAdjacentData"* to tell the approach whether or not it should consider these data cells in the transformation. In our example, the parameter is set to True and thus we build the key-value pair for the *adjacent data cell* as:
  $\boxed{\textit{ValueIn(Adjacent column, Header row), ValueIn(Adjacent column, Intersection row)}}$
  For Fig. 2, the adjacent cell has the key-value pair = (*"Price"*, *"100"*).
- Related (object) data: According to the cross-table definition, an intersection mark relates objects data with their attributes. The object data cells are found to the left of an intersection mark; non-empty and does not fall in the ignored type of cells. The key-value pair for a *related object cell* is:
  $\boxed{\{\textit{ValueIn(Object column, Header row), ValueIn(Object column, Intersection row)}\}}$
  For Fig. 2, the object cell describing the *"Server ID"* has the key-value pair = (*"Server_ID"*, *"Server_1"*).
- Ignored: Our approach ignores two types of cells. First, we ignore other intersection cells within the same row, since they will have their own key-value tuple. Secondly, we ignore cells adjacent to other intersection cells, since one adjacent cell is linked to each intersection cell.

Step 3 - Decide common keys from the transformed cross-tables: The keys used in the cross-table data tuples are going to be the column names of the target relational table. To decide these keys, from all the transformed cross-tables, we perform two actions. First, we identify the unique keys per cross-table's data tuples. Second, we measure the coverage for each unique key over the whole set of identified cross-tables. The coverage of a key is the division of the count of its unique occurences over the total count of the identified cross-tables. Keys that have a coverage above 80% are included as columns in the target relational table. The integration method we follow may be considered simple, for this we provide further discussion in Sect. 7.

Step 4 - Generate the output file: In this step, the approach generates the target relational table, which is encoded in a standard SQL script. The SQL script includes a *"create"* statement representing the structure, and *"insert"* statements representing the data. The columns of the relational table are the common keys decided from the previous step, in addition to the mandatory columns: *"attribute"* and *"sourcefile"*. All columns are assigned a text data type, which is chosen to eliminate additional complexities of type detection and conversion. The generation of the SQL output file starts by making the create statement of the target table. Thereafter, the approach processes the collection of key-value data tuples for each cross-table. For each data tuple an insert statement is generated. Again, the insert statement includes the pairs whose keys are chosen in advance. When a data tuple does not include a pair for one of the chosen keys, we assign to its key the default value *"empty_by_approach"*.

5 Evaluation: Case Study

We evaluate our approach presented in the previous sections, through a case study. In this section, we assess the performance of our approach first quantitatively by answering the following:

RQ1: How many cross-tables and intersection marks were extracted by the approach?
RQ2: Did the approach fail to detect a targeted cross-table completely? If yes, how many?
RQ3: Did the approach fail to extract particular intersection marks from the successfully detected cross-tables? If yes, how many?

Following the quantitative analysis, we perform a qualitative assessment through an interview with two frequent spreadsheet users in the company.

5.1 Context and Dataset

Solvinity[4] is an IT solution provider. Their finance department uses spreadsheets for calculating the monthly bills for each customer. Within a billing spreadsheet a cross-table, similar to the one in Fig. 2, is used to record the sold third-party licenses. An internal audit required the verification of licenses from all available sources, including the spreadsheets. Collecting the required cross-table data was challenging to users, especially with all the manual work needed. Thus, our approach was applied to extract license data in cross-tables, into one relational table. The output was subsequently used by the auditing team for further analysis. The spreadsheet dataset on which the approach operated includes 333 spreadsheets with 2,801 worksheets.

[4] https://www.solvinity.com/.

5.2 Results

In a mixed method evaluation, we performed both quantitative and qualitative methods to analyze the results of our approach. Quantitative analysis assesses the performance of the extraction and transformation approach by answering the questions (RQ1, RQ2 and RQ3) presented earlier. In addition, we interview two frequent spreadsheet users, to asses the approach qualititively. Before the interviews, these users were asked to manually validate a selected subset of spreadsheets against their extracted data.

Table 2. Summary of the case study results, after applying our approach to extract and transform cross-table data. Further details in Sect. 5.2.

Result class	Cross-table class	Cross-table count	Cross marks count	Performance (Identify Cross-table)	Performance (Transform Cross Mark)
Succeeds	Complete cross-tables	1, 442	14, 099	99.17%	92.83%
	Partial cross-tables	6	1, 182	0.41%	
Misses	Complete cross-tables	6	1, 139	0.41	7.17%
	Partial cross-tables	6	41	0.41%	
Total		1, 454	16, 461	100%	100%

Quantitative Evaluation: Among the 333 spreadsheets, the spreadsheet parser failed to read the contents of two spreadsheets. However, we checked these two files manually, and no cross-tables were detected. This leaves 331 spreadsheet files for the approach to analyze, with a total of 2,801 worksheets. Table 2 represents the summary of the approach performance in extracting and transforming cross-tables, and cross-table marks. To understand the statistics better, we answer the first question:

RQ1: How many cross-tables and intersection marks were extracted and transformed by the approach?
The approach detected 1,448 cross-tables. Within these cross-tables, 15,281 intersection marks were extracted and transformed into one relational database table. SQL statements in the output file were syntactically correct. Out of the 2,801 worksheets, we found that no cross-table data were extracted from 1,353 worksheets (48.3%). The reasons for not extracting a cross-table may be that the worksheet did not include a cross-table in the first place, or that the approach detection failed due to wrong keywords for example. To understand the actual reason, we further analyze this result in RQ2 and RQ3.

RQ2: Did the approach fail to detect a targeted cross-table completely? If yes, how many?

Among the 1,353 worksheets from which no cross-tables were extracted, 1,347 worksheets did not include any cross-table inside. The remaining 6 worksheets included 6 cross-tables, one per worksheet. The approach failed to detect these six cross-tables, because it failed to handle the extra number of empty columns in these cross-tables. The six *"missing"* cross-tables includes 1,139 marks, which as a result, were not transformed to the target relational table. Nevertheless, in another run of the approach, we increased the parameter *"Empty Columns to Skip"* to 2, and these cross-tables were successfully extracted. We refrain from setting the *"Empty Columns to Skip"* parameter to a large value, since this will increase the risk of extracting other irrelevant data lying next to the cross-table form in the worksheet.

RQ3: Did the approach fail to extract particular intersection marks from successfully detected cross-tables? If yes, how many?

In total 1,448 cross-tables were detected by the approach, one cross-table per worksheet. To verify that all the marks in these cross-table were transformed, we performed a separate analysis. We calculated the counts of all cells with the mark value X, per worksheet. Subsequently, we validated these counts against the number of marks extracted by our approach for each cross-table. What we found is that 1,442 cross-tables were completely migrated into the relational table, while 6 cross-tables were partially transformed. The missing cross-table marks in this case are 41. The root cause for missing these cross-table marks is the same that caused the approach to miss the complete cross-tables: the large number of empty columns within the cross-tables structure.

Qualitative Evaluation: In addition to the quantitative evaluation, we interviewed two spreadsheet users. The users worked at the finance department, and their daily job is directly related to the spreadsheets under analysis. One user has been developing and maintaining these spreadsheets for six years within the accounting team. The other user has experience in accounting as a business controller, and he joined the team recently. Prior to the interviews, the end-users were provided with a subset of 30 spreadsheets, selected from both successful and failed cases of the approach. We additionally provided the relational table corresponding to the cross-table data from these 30 files. With these data, the end-users were asked to validate the completeness and contextual correctness of the extracted data by our approach.

Users perspective: When asked about the correctness of the information retrieved, according to the context, both users found them, completely correct. One user highlights that *"no factual errors"* were identified in the generated relational table. On the completeness of the retrieved data, one user only was able to detect the missing cross-table marks from the target relational table. The users, after completing their verification, gave a high rating for the approach's extraction capabilities, and regarded it as *"the only gathered data for the sold licenses"*. In addition, the approach was described as *"useful"*, especially for *"these kinds of auditing projects"*, and saving a lot of manual work. However, one user in particular highlighted that *"there will always be a risk"* of extracting *"wrong data"*. He reasoned that the spreadsheets are *"manually designed"* and do not follow a strict template. He concluded that adopting *"more standardized"* designs in the spreadsheets will minimize this risk in the future.

6 Related Work

Cunha *et al.* [18] and Hermans *et al.* [17] targeted the transformation of data inside a spreadsheet into a class diagram, for the sake of improving spreadsheet comprehension through visualization. Both works focused on tabular data inside spreadsheets, and were effective in the detection and transformation to class diagrams, one spreadsheet at a time. However, the prototypes were evaluated on a small number of spreadsheets, and no aggregation approach was considered. Cunha *et al.* [19] implemented an approach that maps spreadsheet data into a relational database, with the aim of normalizing data in a spreadsheet. The work, however, was evaluated on simple spreadsheets without addressing any semi-structured data forms. All mentioned work aimed at keeping the end-user working within the spreadsheets, but with an improved environment and understanding. Among the work that aimed at migrating from spreadsheets is Senbazuru [20,21]. They targeted the extraction of hierarchical data from spreadsheets into a relational database. Their approach, however, is more domain specific, and does not consider aggregating similar data in multiple spreadsheets.

7 Discussion

In prior sections, we described an approach that extracts data from cross-tables in spreadsheets. In this section, we highlight some issues related to our approach.

Using the Spreadsheet Formulas: Formulas are an essential part of spreadsheet development. One cell of data may be the output of a series of formulas which use multiple data cells and ranges. Our current approach considers data in cross-tables, for a more complete migration, the formulas should be considered. Considering formulas' parameters and calculations as part of the data migration may help in building a better relational structure of the extracted data. However, the automatic parsing and translation of formulas to another language is a complex process, therefore it can be an area to explore in future work.

More Complex Cross-Tables: Cross-tables may vary in design to some extent because of the flexibility provided to the end-user, and the nature of the cross-table itself. In our approach, we considered cross-tables with simple attributes, with one row of data. However, users may build more complex, and hierarchical headers. Since previous research extracted hierarchical data structures from spreadsheets [20,22], it is a viable option in future to incorporate hierarchical data detection to widen the application of our approach.

Data Integration: The process of data integration includes the activities performed, and the approaches followed to combine heterogeneous data from multiple sources [23]. Our approach followed a two-step method: first is to decide the common columns from multiple files, and second is to do an exact match of key names. Even though the method is considered simple, it showed near perfect results in the study evaluation, due to a high level of consistency in the naming of columns. However, variation in the naming may occur in another setup. One way to improve is to consider a general schema resource in order to compare and integrate column headers, or a to build a schema dictionary from within the spreadsheet dataset, prior to the extraction [22].

8 Conclusions and Future Work

In this paper we provided the design and implementation of an approach aimed to extract and transform data in cross-tables from multiple spreadsheet files, into one relational table. Our approach was successfully evaluated in an industrial case study, where the resulted table was used in further analysis. The approach succeeded in minimizing the human efforts in the extraction, compared to a manual process. Results show that the approach was able to transform 92.83% of the data in the targeted cross-tables, based on the intersection mark detection. In future work, we aim at eliminating the user role in the extraction, through adopting a machine learning algorithm.

References

1. Clarke, S., Tobias, A.: Corporate modelling in the UK: a survey. OR Insight **8**(3), 15–20 (1995)
2. Chan, Y.E., Storey, V.C.: The use of spreadsheets in organizations: determinants and consequences. Inf. Manage. **31**(3), 119–134 (1996)
3. Croll, G.J.: The importance and criticality of spreadsheets in the city of London. In: Proceedings of European Spreadsheet Risks Interest Group (EuSpRIG), pp. 82–92 (2005). ISBN: 1-902724-16-X
4. ClusterSeven: ClusterSeven Annual State of the Spreadsheet 2016: The Spreadsheet is Here to Stay (2016)
5. Chen, Z.: Information extraction on para-relational data. Ph.D. thesis, The University of Michigan (2016)
6. Hermans, F.: Analyzing and visualizing spreadsheets. Ph.D. thesis, Technische Universiteit Delft (2012)
7. Bernhard Ganter, G.S.: Formal concept analysis: methods and applications in computer science (2003). http://www.math.tu-dresden.de/~ganter/cl03/stumme/chapter1_2.pdf
8. Hermans, F., Murphy-Hill, E.: Enron's spreadsheets and related emails: a dataset and analysis. In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (2015)
9. Hermans, F., Pinzger, M., van Deursen, A.: Supporting professional spreadsheet users by generating leveled dataflow diagrams. In: Proceeding of the 33rd International Conference on Software Engineering - ICSE 2011 (2011)

10. Ben Nasr, S., Bécan, G., Acher, M., Ferreira Filho, J.B., Baudry, B., Sannier, N., Davril, J.M.: Matrixminer: a red pill to architect informal product descriptions in the matrix. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2015 (2015)
11. Sannier, N., Acher, M., Baudry, B.: From comparison matrix to variability model: the wikipedia case study. In: 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE) (2013)
12. Belohlavek, R.: Introduction to formal concept analysis. Palacky University, Department of Computer Science, Olomouc (2008)
13. Tilley, T.: Formal concept analysis applications to requirements engineering and design. Ph.D. thesis, The University of Queensland (2003)
14. Vitek, D.: Requirements traceability matrix template. http://www2a.cdc.gov/cdcup/library/templates/CDC_UP_Requirements_Traceability_Matrix_Template.xls
15. Abraham, R., Erwig, M.: Ucheck: a spreadsheet type checker for end users. J. Vis. Lang. Comput. **18**(1), 71–95 (2007)
16. Roy, S., Hermans, F., Aivaloglou, E., Winter, J., van Deursen, A.: Evaluating automatic spreadsheet metadata extraction on a large set of responses from MOOC participants. In: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER) (2016)
17. Hermans, F., Pinzger, M., van Deursen, A.: Automatically Extracting Class Diagrams from Spreadsheets. Springer, Heidelberg (2010)
18. Cunha, J., Erwig, M., Saraiva, J.: Automatically inferring classsheet models from spreadsheets. In: 2010 IEEE Symposium on Visual Languages and Human-Centric Computing (2010)
19. Cunha, J., Saraiva, J., Visser, J.: From spreadsheets to relational databases and back. In: Proceedings of the 2009 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation - PEPM 2009 (2008)
20. Chen, Z., Cafarella, M., Chen, J., Prevo, D., Zhuang, J.: Senbazuru: a prototype spreadsheet database management system. Proc. VLDB Endow. **6**(12), 1202–1205 (2013)
21. Chen, Z., Cafarella, M.: Automatic web spreadsheet data extraction. In: Proceedings of the 3rd International Workshop on Semantic Search Over the Web - SS@ 2013 (2013)
22. Chen, Z., Cafarella, M.: A semiautomatic approach for accurate and low-effort spreadsheet data extraction. Technical report, University of Michigan (2014)
23. Magnani, M., Rizopoulos, N., Mc.Brien, P., Montesi, D.: Schema integration based on uncertain semantic mappings. In: Delcambre, L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, O. (eds.) ER 2005. LNCS, vol. 3716, pp. 31–46. Springer, Heidelberg (2005). doi:10.1007/11568322_3