DEUTSCHE BUNDESBANK
EUROSYSTEM

Statistical Disclosure Control (SDC) for results derived from combined confidential microdata

Technical Report 2023-02

The views expressed in this technical report are personal views of the authors and do not necessarily reflect the views of the Deutsche Bundesbank or the Eurosystem.

Deutsche Bundesbank, Research Data and Service Centre

Jannick Blaschke
Christian Hirsch
Robin Kollmann

Research Data and Service Centre

Abstract

Most researchers combine two or more datasets in their projects. In addition to technical feasibility and the meaningfulness of content, it is also necessary for researchers working with confidential Bundesbank microdata to consider possible consequences for Statistical Disclosure Control (SDC). This Technical Report highlights three possible SDC challenges that may arise from appending and merging datasets: (1) loss of rows, (2) newly generated missing values in SDC variables, and (3) newly generated duplicates. It then presents possible solutions and provides simplified examples and rules of thumb.

Keywords: statistical disclosure control, sdc, output control, research data centre, rdc, appending, merging, joining

Version: 1.0

Citation: Blaschke, J., Hirsch, C., and Kollmann, R. (2023)[1)]. Statistical Disclosure Control (SDC) for results derived from combined confidential microdata, Technical Report 2023-02 – Version 1.0. Deutsche Bundesbank, Research Data and Service Centre.

1 The authors gratefully acknowledge the contributions from our colleagues at the RDSC and Matthias Gomolka.

Contents

1 Introduction

Most empirical research papers contain results based on multiple datasets. We distinguish two kinds of data operations in this paper. "Appending" datasets, i.e. extending a dataset with additional rows from another dataset, and "merging" datasets, i.e. combining columns from two datasets.

Whenever merging two datasets, researchers must consider not only technical feasibility (i.e. that there is at least one overlapping column in both datasets that can be used to identify the rows that should be combined), but also the meaningfulness of the resulting dataset. If, for example, we are merging two datasets with company information, we should first ask ourselves whether the companies in both datasets are defined in the same way. E.g. if one dataset contains information on the entire group and the second only on the German part, the company names may be the same, but caution is advised when interpreting the variables of the merged dataset.

In addition, when working at the Research Data and Service Centre (RDSC) of the Deutsche Bundesbank, all possible effects on Statistical Disclosure Control (SDC) must be kept in mind at all times[2)]. Results that do not pass SDC cannot be taken out of the RDSC's secure environment or used in publications. Confidential microdata differ in this regard from other data sources where researchers can focus exclusively on obtaining results.

The purpose of this technical report is to raise awareness among researchers that working with confidential microdata means focusing on two equally important objectives: (i) achieving interesting research findings while also (ii) checking that these results comply with SDC. Furthermore, these two objectives arise concurrently as researchers who only focus on results usually risk time-consuming retroactive adjustments to their code to ensure compliance.

This technical report is structured as follows. In Section 2, we briefly introduce the concepts of appending and merging. Section 3 explains the three SDC challenges - loss of rows, missing values in SDC variables and newly generated duplicates - and presents a solution for each challenge. Bringing together the two previous sections, Section 4 shows under which conditions, these challenges occur when appending or merging datasets. Section 5 concludes with some rules of thumb. Finally, we have included an Appendix with a few examples of merges based on actual Bundesbank datasets as well as a Glossary with main technical terms.

2 More information on the RDSC's SDC rules can be found in the "*Rules for visiting researchers at the RDSC*" (Research Data and Service Centre (2021)), which are available at https://www.bundesbank.de/resource/blob/826176/ffc6337a19ea27359b06f2a8abe0ca7d/mL/2021-02-gastforschung-data.pdf

2 Brief theoretical foundation

Before we take a detailed look at the issues that can arise when combining different datasets, we must distinguish between the most important terms. We are aware that the shape of the resulting dataset can be influenced by additional options in the append or merge command[3]. However, for the sake of clarity, we assume that the respective commands are used in their simplest form[4].

2.1 Appending

When a dataset B is added from below to a dataset A, we speak of "appending" dataset A to dataset B (see Figure 1). To append successfully, the columns of interest should be included in both A and B. The shape of the resulting dataset C will be as follows:

- The number of rows will be the sum of the rows in A and in B.
- The number of columns will be the union of the columns in A and in B.

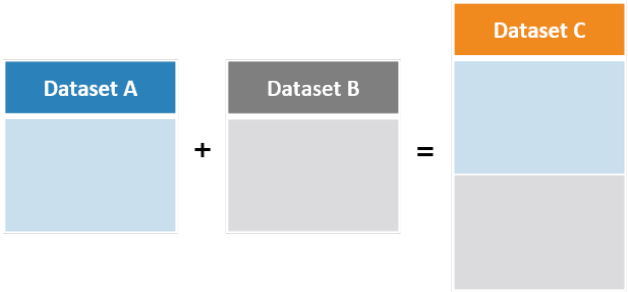


Figure 1: Appending two datasets, A and B

In Stata[5] for example, the commands to append two datasets named A and B could look like below:

```
// Load dataset A
use A, clear

// Append datasets A and B
append using B
```

3 For example, Stata allows the "keep(varlist)" option in their append command, which allows the user to specify that the variables from the "using" dataset should be kept (StataCorp. (2021a)).
4 For a complex example of merging multiple company datasets from the Deutsche Bundesbank using machine-learning-based classification see Schild et al. (2017) as well as Doll et al. (2021) and Gábor-Tóth & Schild (2021) for the practical implementation.
5 For more information on the Stata **append** command, please see the Stata reference guide (StataCorp. (2021a)).

2.2 Merging

The "merging" command can be used to combine the columns from two datasets into a single dataset (see Figure 2)[6]. As the datasets are combined sidewards we often speak of a left and a right dataset. After merging two datasets, A and B, the shape of the resulting dataset C will be as follows:

- The number of rows will be smaller than, equal to or larger than the number of rows in A or B.
- The number of columns will be equal to or larger than the number of columns in A or B.

Figure 2: Merging two datasets, A and B

Merges are usually done using **indexes** or **key variables**. When merging using an index, the first observation in A will be matched to the first observation in B, the second in A with the second in B and so on. When merging using one (or multiple) key variable(s), this variable must be included in both datasets so that the program can match similar values (e.g. if the key variable is a company ID, then information from A and B is merged per instance of this ID, i.e. per company.)[7]. If merging using key variables, indexes will be ignored. For the remainder of this paper, we assume merges will use key variables.

Depending on the dataset structure, we distinguish between the following **key variable relationship types**, which describe how the key variables of two datasets are related to each other when they are merged:

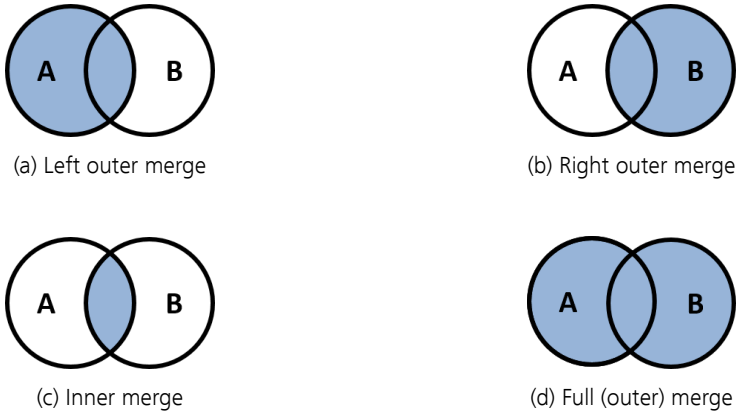
- **One-to-one (1:1)**: Each expression of the key variables used to merge occurs once in each of the two datasets.
- **Many-to-one (m:1)**: Each expression of the key variables used for the merge occurs 1-m times in the left dataset and - exactly once in the right data set.
- **Many-to-many (m:m)**: Each expression of the key variables used for the merge occurs 1-m times in both datasets.
- **One-to-many (1:m)**: Each expression of the key variables used for the merge occurs exactly once in the left dataset and 1-m times in the right data set.

6 Similar to merging, "joining" also allows the columns of two datasets to be combined. Depending on the programming language used, there can be slight differences between merge and join commands. In Python for example, the method *join*() is used to combine two DataFrames based on their indexes whereas *merge*() requires specifying columns as a (merge) key (McKinney et al. (2010)). However, for the purposes of this paper, we will use merge as a synonym for join.
7 Key variables can (but do not have to) be SDC variables (see Glossary), i.e. a company ID. For example if a row in a dataset is identified by the combination of a date variable, which is not an SDC variable, and the company ID.

In addition, we distinguish the following four **merge types**. They determine which rows will be kept and which ones will be dropped in the merged dataset:

- **Left outer merge**: Returns all rows from the left dataset and only the merged ones from the right dataset.
- **Right outer merge**: Returns all rows from the right dataset and only the merged ones from the left dataset.
- **Inner merge**: Returns only merged rows from both datasets.
- **Full (outer) merge**: Returns all rows from both datasets regardless of whether they were merged or not.

Figure 3 visualises the four merge types using datasets A (left) and B (right) as an example:



(a) Left outer merge (b) Right outer merge

(c) Inner merge (d) Full (outer) merge

Figure 3: The four merge types

The relation type of the key variables and the merge type are specified within the `merge` command or directly afterwards. Note, that Stata uses the terms "master" for left and "using" for the right dataset[8]. In this case, the commands for a merge of two datasets named A and B could look like the following:

```
// Load dataset A
use A, clear

// 1:1 left outer merge with "ID_var" as key variable (in one step)
merge 1:1 ID_var using B, keep(master)

// 1:m right outer merge with "ID_var" as key variable (in two steps)
merge 1:m ID_var using B
keep if _merge == 2 | _merge == 3

// m:1 inner merge with "ID_var" as key variable (in one step)
merge m:1 ID_var using B, keep(match)
```

8 For more information on the Stata merge command please see the Stata reference guide (StataCorp. (2021b)).

3 What hampers SDC?

At this point we would like to highlight that this report attempts to describe SDC challenges in a short and concise way, while explicitly acknowledging that there will always be **special cases** for which the rules mentioned do not fully apply. We therefore strongly recommend considering the particular structure of the data provided by the RDSC and any changes made to it. All resulting SDC requirements should be kept in mind at all times.

As part of their projects, researchers generate results based on confidential Bundesbank microdata. These could be descriptive statistics (e.g. a mean or a frequency table) as well as regressions. When performing SDC on such results, the dataset that was used to compute them (analysis dataset) ideally has the following properties:

1. Usually, results are not calculated directly using the unmodified Bundesbank data (original dataset), but may also be based on previously self-generated variables. Thus, for the SDC of results based on previously self-generated variables, it is imperative that all **rows** that were used for the generation of these variables are also still contained in the analysis dataset.
2. SDC variables do not contain **missing values**.
3. The dataset does not contain **duplicated observations**.

In reality, however, dataset transformations often lead to one or more of these assumptions being violated[9]. In this section, we show these violations cause SDC challenges and how researchers can address them. In Section 4, we then explain whether and when this is the case when appending and merging two datasets.

Challenge 1: Loss of rows

We describe this challenge and possible solutions in detail in our Technical Report "***SDC for results derived from aggregated confidential microdata***" **(Blaschke, Gomolka, et al. (2022))**[10]. Therefore, we will only give a brief overview in this paper.

Assume we have a dataset that contains company information with two SDC variables "Company_ID" and "Group_ID" (see Table 1a). For our analysis, we are only interested in the companies' groups rather than the individual companies. Therefore, we aggregate the data on the group level (see Table 1b). However, if we want to publish any results computed on "Group_Value", we cannot perform SDC procedures based on Table 1b as we no longer know the number of individual companies that belong to each group (see Principle O.1.3 "Adherence to minimum sample size" in Research Data and Service Centre (2021)) and we also cannot check for dominance (see Principle O.1.4, "Adherence to p% or dominance rule" in Research Data and Service Centre (2021)).

As described in Blaschke, Gomolka, et al. (2022), we have to ensure that no SDC variables get

9 Even some of the datasets provided by the RDSC contain SDC variables with missing values or duplicated observations. This is not necessarily a sign of data quality issues but can instead have other origins (e.g. missing values in the ID of a party that is only involved in certain transactions and therefore missing otherwise).
10 The Technical Report is available at https://www.bundesbank.de/resource/blob/745168/3501912ca41dc894f605a9d24413dee1/mL/2022-01-sdc-data.pdf

Company_ID	Group_ID	Value
111	AAA	1000
222	AAA	2000
333	AAA	1400
111	BBB	2100
111	CCC	100
111	DDD	100

(a) Company dataset before aggregation

Group_ID	Group_Value
AAA	4400
BBB	2100
CCC	100
DDD	100

(b) Company dataset after aggregation

Table 1: Loss of rows - Challenge

lost when using a summary function on a dataset. Summary functions calculate results based on multiple values from multiple rows (e.g. an aggregation as in Table 1). Therefore, we have the following three options to ensure SDC compliance:

1. Perform SDC before using the summary function and ensure that each row in the aggregated dataset will satisfy the SDC criteria.
2. Use summary functions only when taking into account the full set of SDC variables.
3. Do not drop duplicated rows after using the summary function. Compute the results by using a dummy variable while performing SDC on the full dataset[11)].

In our example, we choose the third option and define a dummy that is 1 each time "Group_ID" takes on a new value for the first time. Table 2 shows the result:

Company_ID	Group_ID	Value	Group_Value	Dummy
111	**AAA**	1000	**4400**	**1**
222	AAA	2000	4400	0
333	AAA	1400	4400	0
111	**BBB**	2100	**2100**	**1**
111	**CCC**	100	**100**	**1**
111	**DDD**	100	**100**	**1**

Table 2: Loss of rows - Solution

Table 14 in the Appendix provides an example when the loss of rows hampers SDC, while Table 15 shows an SDC compliant loss of rows.

11 Always working with the full dataset can be quite time consuming. To optimise the performance of your code in the RDSC's environment, we recommend our Technical Report "*Working with large data at the RDSC*" (Gomolka et al. (2021)).

Challenge 2: Missing values in SDC variables

Assume we are using a dataset on transactions where company A is selling a product to company B and where, for some transactions, sales is through an intermediary. In addition, the dataset also includes the ID of company A's group (ownership information). All four parties (companies A and B, the intermediary and company A's group) must be protected in the SDC (i.e. "Company_A_ID", "Group_A_ID", "Inter_ID" and "Company_B_ID" in Table 3 are SDC variables). Tables 3a and 3b show two examples of this dataset with missing values in one of the SDC variables.

Company_A_ID	Group_A_ID	Inter_ID	Company_B_ID	Value
F05	C		L01	1100
G02	D	Y	V73	20
J08	E		R99	9800
P11	F		C22	3200
Z21	G		K09	4000
L99	H		Q33	6600

(a) Missing ID values are uncorreleated with other SDC variables

Company_A_ID	Group_A_ID	Inter_ID	Company_B_ID	Value
F05	C	X	L01	1100
G02		Y	V73	20
J08	C	Y	R99	9800
P11	D	Z	C22	3200
Z21	E	X	K09	4000
L99		Y	Q33	6600

(b) Missing ID values are correlated with other SDC variables

Table 3: Newly generated missing values - Challenge

Let us assume that, we would like to compute the sum of the "Value" column in Table 3a and perform SDC afterwards. The result shows that the aggregate would be based on six different IDs in "Company_A_ID", "Group_A_ID", and "Company_B_ID" and that the dominance criteria would not be violated[12]. However, we only have a single observation with an intermediary and thus, the overall SDC check would result in a disclosure problem[13].

Does this mean that it is generally very unlikely to get results through SDC if a dataset contains a rarely filled SDC variable? This question should be answerable from the **context**. To do so, RDSC researchers must proceed as follows:

1. Test whether the missing values of the SDC variable can be filled by **interpolation**. Some SDC variables are correlated with other SDC variables and can thus be adequately explained by them. In the example in Table 3b, the "Group_A_ID" is missing for some transactions. The reason for this is that not all companies may belong to a group. In these cases, the missing group IDs may be imputed from the related ID variable (here "Company_A_ID"). Please refer to the respective data report to understand whether this procedure can be applied to an SDC variable in a specific dataset.
2. If interpolation is **not possible** and the SDC variable with the missing values is not sufficiently

12 Computation of dominance: (9800 + 6600)/(1100 + 20 + 9800 + 3200 + 4000 + 6600) = 66.34% $\leq$ 85%
13 Computation of dominance: (20)/(20) = 100% > 85%

correlated with any other SDC variable, the missing values can be **chronologically numbered**. As previously mentioned, not all transactions in Table 3a are processed via an intermediary. As "Inter_ID" cannot be imputed from any other SDC variable, all missing values in "Inter_ID" can be assigned to random ID values that are unique per missing value. To avoid any confusion with the real IDs, the following naming convention must be observed: *"sdc_" + consecutive number* (e.g. "sdc_15").

Table 4 shows hypothetical SDC results if each approach were applied to the cases presented in Table 3. The two additional variables indicate the sample size ("SDC_N") and the dominance ("SDC_Dom") for "Group_A_ID"[14] and "Inter_ID"[15]:

					for "Group_A_ID"	
Company_A_ID	Group_A_ID	Inter_ID	Company_B_ID	Value	SDC_N	SDC_Dom
F05	C	X	L01	1100		
G02	**G02**	Y	V73	20		
J08	C	Y	R99	9800	6	70.79%
P11	D	Z	C22	3200		
Z21	E	X	K09	4000		
L99	**L99**	Y	Q33	6600		

(a) Interpolated missing IDs

					for "Inter_ID"	
Company_A_ID	Group_A_ID	Inter_ID	Company_B_ID	Value	SDC_N	SDC_Dom
F05	C	**sdc_1**	L01	1100		
G02	D	Y	V73	20		
J08	E	**sdc_3**	R99	9800	6	66.34%
P11	F	**sdc_4**	C22	3200		
Z21	G	**sdc_5**	K09	4000		
L99	H	**sdc_6**	Q33	6600		

(b) Consecutively numbered missing IDs

Table 4: Newly generated missing values - Solutions

The missing values in Table 4b can be replaced with Stata as follows:

```
// Create a new string variable with continuous numbering
gen id_temp = _n
tostring id_temp, replace

// Add prefix to this variable to comply with naming convention
replace id_temp = "sdc_" + id_temp

// Replace missing values in "Inter_ID"
replace Inter_ID = id_temp if missing(Inter_ID)
drop id_temp
```

14 Computation of dominance: ((1100 + 9800) + 6600)/(1100 + 20 + 9800 + 3200 + 4000 + 6600) = 70.79% $\leq$ 85%
15 Computation of dominance: (9800 + 6600)/(1100 + 20 + 9800 + 3200 + 4000 + 6600) = 66.34% $\leq$ 85%

Challenge 3: Duplicates

Table 5a shows a dataset on company groups. Since we are also interested in the associated values for the individual companies, we want to enrich this dataset with data from Table 5b. As some groups own more than one company, we use a 1:m inner merge[16]. Table 5c shows the merged dataset. Although there are no missing values or deleted rows, there are duplicates in the rows previously contained in Table 5a.

Group_ID	Group_Value
AAA	1000
BBB	2100

(a) Company group dataset

Group_ID	Company_ID	Company_Value
AAA	F22	40
AAA	G54	50
AAA	O99	60
AAA	W71	70
AAA	A01	80
BBB	J77	10
BBB	C30	20

(b) Company dataset

Group_ID	Group_Value	Company_ID	Company_Value
AAA	1000	F22	40
AAA	1000	G54	50
AAA	1000	O99	60
AAA	1000	W71	70
AAA	1000	A01	80
BBB	2100	J77	10
BBB	2100	C30	20

(c) Merged dataset with company and group information

Table 5: Newly generated duplicates - Challenge

If we now calculate the mean of the variable "Group_Value", we cannot simply divide the sum of all values in "Group_Value" by the number of rows in "Group_ID". Instead, we must find a way to account for the duplicates in "Group_ID" and "Group_Value" that were generated when merging. Here, the best method is to follow the approach presented in *Challenge 1: Loss of rows* where we used a dummy variable (see Table 2). Hence, the mean of "Group_Value" would have to be (1000 + 2100)/2 = 1550. Table 6 shows the resulting dataset including the SDC variables for sample size ("SDC_N") and dominance ("SDC_Dom")[17] for the SDC variable "Group_ID":

					for "Group_ID"	
Group_ID	Group_Value	Company_ID	Company_Value	Dummy	SDC_N	SDC_Dom
AAA	**1000**	F22	40	**1**		
AAA	1000	G54	50	0		
AAA	1000	O99	60	0		
AAA	1000	W71	70	0	2	100%
AAA	1000	A01	80	0		
BBB	**2100**	J77	10	**1**		
BBB	2100	C30	20	0		

Table 6: Newly generated missing values - Solution

16 Note, that since all Group IDs are contained in both datasets, the result would have been the same for a left and right outer merge or a full (outer) merge.
17 Computation of dominance: (1000 + 2100)/(1000 + 2100) = 100% > 85%

4 When does appending or merging hamper SDC?

Readers should differentiate between existing challenges in datasets and newly created challenges that appear because of appending or merging datasets. This section explains the origins of the latter. For SDC, all challenges need to be adressed irrespective of their origin.

4.1 Appending

A dataset C that is generated by appending two datasets A and B will always contain all columns and rows from A and B. Hence, no columns or rows are lost. However, appending can generate **duplicates** and **missing values**. Duplicates occur if at least one row exists in both A and B. Missing values are generated if there is at least one column that is only included in A or in B:

- Columns only included in A will lead to missing values in all rows in C that came from B.
- Columns only included in B will lead to missing values in all rows in C that came from A.

Table 7 shows how data for a new month (dataset B) are appended to an existing dataset (dataset A). The length of dataset C is the sum of the rows in datasets A and B while the number of columns in C is the union of A and B. Variable "Group_ID" was only included in dataset B and therefore all previous months in C are empty in "Group_ID". Finally, note that no observations from SDC variables ("Company_ID", "Group_ID") were dropped. If "Group_ID" is an SDC variable, we have to address this challenge as described in the previous section.

Date	Company_ID	Company_Value
2021-01	F55	10
2021-01	L91	20
2021-02	F55	20
2021-02	L91	30

(a) Dataset A

Date	Company_ID	Group_ID	Company_Value
2021-03	F55	AAA	10
2021-03	L91	BBB	20

(b) Dataset B

Date	Company_ID	Group_ID	Company_Value
2021-01	F55		10
2021-01	L91		20
2021-02	F55		20
2021-02	L91		30
2021-03	F55	AAA	10
2021-03	L91	BBB	20

(c) Dataset C

Table 7: Example of appending datasets A and B

If missing values are generated in "Company_Value" these must not be filled in with a value (e.g. zeros) but instead should be preserved as missing. Otherwise, it would be impossible to

distinguish between a reported value (even if zero) and a missing value that must not be included in the SDC computation.

4.2 Merging

Whether SDC challenges occur after merging datasets depends on the

1. Relationdhip between the key variables (e.g. 1:m vs. 1:1)
2. Merge type (e.g. left outer merge vs. inner merge)
3. Content of the datasets (e.g. if both datasets contain the same codes in the merge key variable(s))

As we have these three dimensions determining whether SDC challenges need to be considered. Table 8 an initial overview of potential challenges. Note, that the table only shows **merge-induced** SDC challenges but does not contain any information about pre-merge characteristics of the individual datasets (e.g. dataset A could have already had SDC variables with missing values before the merge):

	Left dataset (dataset A)			Right dataset (dataset B)		
	Deleted rows	Duplicates	Missing values	Deleted rows	Duplicates	Missing values
1:1	None	None	None	Non-merged rows	None	Possible
m:m		Possible			Possible	
1:m					None	
m:1		None			Possible	

(a) Left outer merge

	Left dataset (dataset A)			Right dataset (dataset B)		
	Deleted rows	Duplicates	Missing values	Deleted rows	Duplicates	Missing values
1:1	Non-merged rows	None	Possible	None	None	None
m:m		Possible			Possible	
1:m					None	
m:1		None			Possible	

(b) Right outer merge

	Left dataset (dataset A)			Right dataset (dataset B)		
	Deleted rows	Duplicates	Missing values	Deleted rows	Duplicates	Missing values
1:1	Non-merged rows	None	None	Non-merged rows	None	None
m:m		Possible			Possible	
1:m					None	
m:1		None			Possible	

(c) Inner merge

	Left dataset (dataset A)			Right dataset (dataset B)		
	Deleted rows	Duplicates	Missing values	Deleted rows	Duplicates	Missing values
1:1	None	None	Possible	None	None	Possible
m:m		Possible			Possible	
1:m					None	
m:1		None			Possible	

(d) Full (outer) merge

Table 8: Possible SDC challenges by merge type

For example, in case of a left outer 1:m merge, Table 8 (a) states that when merging A and B, we keep all rows that are either only in A or in both A and B, while we drop those only occurring in B. For those rows that only exist in A, missing values are generated in SDC variables only existing in B. As we are using a 1:m merge we know that each expression of the key variables used for the merge occurs exactly once in A and 1-m times in B. Therefore, duplicates in A will be generated for each expression of key variables that exists multiple times in B.

5 Rules of thumb for resolving these SDC challenges

This section concludes the main findings on addressing SDC challenges arising from appending and merging datasets. Note again, that SDC challenges might already exist in the original data as provided by the RDSC. They must be addressed irrespective of their origin.

5.1 Appending

Appending two datasets does not usually lead to serious challenges when performing SDC. However, you should keep in mind that you might generate missing values in SDC variables. Also, you should make sure not to drop any SDC variables even when they are rarely filled and not the focus of your research.

5.2 Merging

It is important that researchers understand that the three SDC challenges are not equally serious. While an analysis dataset with missing values or duplicates can usually be accounted for during SDC, this is no longer possible when rows are missing which are needed for SDC. In the latter case, the previous program code would have to be reworked. Therefore, it can be said that **missing rows are by far the biggest SDC challenge**.

As a general rule, we learn from Table 8 that:

- "Deleted rows" and "Missing values" depend on the merge type.
- "Duplicates" depend on the relation of the key variables

Thus, we conclude the following rules of thumb:

- **Understand your data structure**. Key variables identify rows in datasets. The origin of many SDC challenges lies in the relationship between these key variables in the original datasets. Once you know if you have e.g. a 1:m or a 1:1 relationship in your set of merge key variables, you can use Table 8 to understand which SDC challenges you are likely to encounter.
- **Understand the impact of your selection of merge key variables**. Merging datasets will never create new missing values in SDC variables that are in the set of merge key variables. However, sometimes (e.g. when the two datasets have different SDC variables) we cannot merge all SDC variables in a dataset. This often happens when one of the datasets contains multiple parties (e.g. buyer and seller or borrower and lender) or reflects ownership relations (e.g. parent and subsidiary). These cases will most likely lead to new missing values being created in an SDC variable.
- **If you cannot reverse it, try to avoid it**. As mentioned above deleted rows cannot be reversed. When researchers decide to first compute a summary function based on multiple rows and then drop some of those rows, they should carefully plan in advance how to address the challenge of deleted rows. We strongly recommend first merging the original datasets and then following the approach from Blaschke, Gomolka, et al. (2022).

- **Use full (outer) merges**. All merge types other than full (outer) merges involve deleting rows from one or both datasets if key variables do not overlap perfectly.
- **Use 1:1 merges**. For all merge types, 1:1 merges perform best. However, we are well aware that 1:1 merges are rarely possible in reality. On the other end, m:m merges always perform worst and even the Stata reference manual for the `merge` command states, it says "*m:m specifies a many-to-many merge and is a bad idea. (…) If you think that you need an m:m merge, then you probably need to work with your data so that you can use a 1:m or m:1 merge.*" (StataCorp. (2021b)). By definition, 1:m (m:1) merges can duplicate a single row in the left (right) dataset up to m times.
- **Understand that duplicated rows make untidy data**. Duplicated rows and tidy data make for a formidable trade-off (Wickham (2014)). A dataset is considered tidy if the same set of key variables identifies all variables in the dataset. By its very definition tidy datasets do not have duplicated rows. However, merging tidy datasets can lead to untidy datasets. For these cases, we recommend sticking with the untidy dataset rather than trying to make it tidy again. Tidying a dataset almost always requires deleting rows, which may severely impair SDC.
- **Consider using dummy variables to navigate between different key variables.** Navigating untidy data for SDC requires understanding which key variables identify an observation (e.g. as done in Table 2). For these cases we recommend using a dummy variable to switch different key variables as described in Blaschke, Gomolka, et al. (2022).

The Appendix shows seven examples of merges that further illustrate these rules of thumb.

Appendix

In this appendix we present seven examples of two datasets being merged. The example datasets are modelled on real datasets provided by the RDSC for research purposes. Examples 1-3 show the results of a hypothetical merge between the "Banks' profit and loss accounts (GuV)" data (Stahl & Scheller (2022)) and the "MFI interest rate statistics (ZISTA)" data (Blaschke, Eiff, et al. (2022)) . Subsequently, examples 4-7 show different merges between GuV and the "Markets in Financial Instruments Directive (MiFID)" data (Cagala et al. (2021)).

Example 1 - Merging GuV and ZISTA

Suppose we want to merge the two datasets in Table 9a and 9b using the variables "Year_Month" and bank-ID "BAID". We see that this set of key variables uniquely identifies a row in both datasets. We can therefore do a **1:1 full (outer) merge**[18]. Table 9c shows the merged dataset.

Year_Month	BAID	GuV_Value
2022-12	A1	100
2022-12	B1	200
2022-12	C1	300

(a) Simplified GuV

Year_Month	BAID	ZISTA_Value
2022-10	A1	20
2022-11	A1	30
2022-12	A1	40
2022-10	B1	110
2022-11	B1	120
2022-12	B1	130

(b) Simplified ZISTA

Year_Month	BAID	GuV_Value	ZISTA_Value
2022-10	A1		20
2022-11	A1		30
2022-12	A1	100	40
2022-10	B1		110
2022-11	B1		120
2022-12	B1	200	130
2022-12	C1	300	

(c) Merged dataset of simplified GuV and ZISTA

Table 9: Example 1

Regarding the three potential SDC challenges we find that

- No rows were deleted.
- No duplicates were generated in either GuV or ZISTA.
- Missing values were generated in the ZISTA data because it did not contain any information on Bank C1. One possible explanation is that the ZISTA only contains a sample of all Monetary Financial Institutions in Germany.

However, missing values were not generated in the SDC variable "BAID". Therefore, there is no SDC challenge in Table 9c. We would only have to keep in mind when computing a summary function (e.g. a mean) on "ZISTA_Value" that the result will be based on 2 different BAIDs (A1 and

18 Note that in this case, a 1:m, m:1 or even m:m would also produce the same result.

B1) and not on all three BAIDs. Therefore, it is important to treat missing values in "ZISTA_Value" as such and not as zeros[19]. In the case of the calculation of a mean this would lead to a wrong result anyway.

Example 2 - Merging GuV and ZISTA

Now, let us assume that we want to repeat this merge for Table 10a and 10b. This time, we would like to merge via the year rather than the month. We have therefore already created the variable "Year" in both datasets. Now, each row in the ZISTA is no longer uniquely identifiable by "Year" and "BAID" and thus, a 1:1 merge would not be possible. Instead, we use a **1:m full (outer) merge**. Table 10c shows the merged dataset.

Year	BAID	GuV_Value
2022	A1	100
2022	B1	200
2022	C1	300

(a) Simplified GuV

Year	Year_Month	BAID	ZISTA_Value
2022	2022-10	A1	20
2022	2022-11	A1	30
2022	2022-12	A1	40
2022	2022-10	B1	110
2022	2022-11	B1	120
2022	2022-12	B1	130

(b) Simplified ZISTA

Year	Year_Month	BAID	GuV_Value	ZISTA_Value
2022	2022-10	A1	100	20
2022	2022-11	A1	**100**	30
2022	2022-12	A1	**100**	40
2022	2022-10	B1	200	110
2022	2022-11	B1	**200**	120
2022	2022-12	B1	**200**	130
2022		C1	300	

(c) Merged dataset of simplified GuV and ZISTA

Table 10: Example 2

Regarding the three potential SDC challenges we find that

- No rows were deleted.
- **Duplicates in the GuV** data were generated (but not in the ZISTA). This confirms, that in a 1:m or m:1 merge, duplicates are always generated for the dataset where the set of merge key variables uniquely identifies a row.
- Missing values were generated in the ZISTA data, however not in a SDC variable. The reason and recommended SDC approach are the same as in the previous example.

To use this merged dataset that now contains both missing values and duplicates, we would need to use a dummy variable both to compute a summary function on "GuV_Value" and for any SDC for a result based on "GuV_Value"[20].

19 Moreover, when creating a new variable, e.g. a mean, this variable must be missing where the underlying value was missing. In Table 9c the new variable would have to be missing in the last row (e.g. `if ZISTA_Value != missing`).
20 For a detailed explanation, see Blaschke, Gomolka, et al. (2022)

Example 3 - Merging GuV and ZISTA

Another possibility to merge via year and BAID would be to first collapse the ZISTA by "Year" and "BAID". As recommended in Blaschke, Gomolka, et al. (2022), all SDC variables (here only "BAID") were included in the collapse command. Table 11b shows the result after the collapse where "mean_ZISTA_Value" is now the mean per year. Now, we can use an *1:1 full (outer) merge* to combine Tables 11a and 11a to table 11c.

Year	BAID	GuV_Value
2022	A1	100
2022	B1	200
2022	C1	300

(a) Simplified GuV

Year	BAID	mean_ZISTA_Value
2022	A1	30
2022	B1	120

(b) Simplified ZISTA

Year	BAID	GuV_Value	mean_ZISTA_Value
2022	A1	100	30
2022	B1	200	120
2022	C1	300	

(c) Merged dataset of simplified GuV and ZISTA

Table 11: Example 3

Regarding the possible three SDC challenges we find that

- Rows from the not-collapsed ZISTA dataset (see Table 10b) were **deleted**. However, as we collapsed in an SDC compliant way, this is **not problematic**.
- No duplicates were generated.
- Missing values were generated in the ZISTA data, but not in an SDC variable. The reason and recommended SDC approach are the same as in the previous example.

Example 4 - Merging GuV and MiFID

Examples 1-3 showed the relatively simple case of merging two datasets with one SDC variable each. However, in reality, the RDSC provides several datasets with multiple SDC variables. The following four examples explain what happens when we merge a dataset with one SDC variable with a dataset with two SDC variables. Again, all examples are modelled on real RDSC datasets which are simplified for the purpose of this paper.

Table 12a shows GuV data similar to the one in the examples above Table 12b shows transactional data from the MiFID. Each row identifies a transaction of a certain value ("MiFID_Value") between a bank ("BAID") and a counterparty ("CP"). While the GuV is collected on a yearly basis[21)], the MiFID contains daily information[22)].

To merge over "Date" and "BAID", we use a **1:m full (outer) merge** as the keys Date-BAID only uniquely identify a row in the GuV but not in the MiFID. Table 12c shows the result.

Date	BAID	GuV_Value
2022-12-31	A1	100
2022-12-31	B1	200

(a) Simplified GuV

Date	BAID	CP	MiFID_Value
2022-12-31	A1	X1	20
2022-12-15	A1	ZZ1	20
2022-12-31	A1	ZZ1	20
2022-12-31	C1	X1	50
2022-12-31	C1	ZZ1	50
2022-12-31	C1	ZZ1	50

(b) Simplified MiFID

Year	BAID	GuV_Value	CP	MiFID_Value
2022-12-31	A1	100	X1	20
2022-12-15	A1		ZZ1	20
2022-12-31	A1	100	ZZ1	20
2022-12-31	B1	200		
2022-12-31	C1		X1	50
2022-12-31	C1		ZZ1	50
2022-12-31	C1		ZZ1	50

(c) Merged dataset of simplified GuV and MiFID

Table 12: Example 4

Regarding the three potential SDC challenges we find that

- No rows were deleted.
- **Duplicates in the GuV** data were generated (but not in the MiFID). As in the previous examples, we would need to use a dummy variable both to compute a summary function on "GuV_Value" and for any SDC for a result based on "GuV_Value".
- **Missing values** were generated in both the GuV and the MiFID data. In contrast to our previous examples, we also generated missing values in an SDC variable ("CP"). As the missing counterparty ID cannot be imputed from the other SDC variable in the dataset ("BAID"), we have to

21 We already added the month and day in the "Date" column to facilitate this example.
22 Note that only bank A1 is included in both datasets, while B1 is only included in the GuV and C1 only in the MiFID. We used these features to better illustrate potential SDC challenges. In reality, this could be the case if both datasets had been filtered based on different criteria beforehand.

replace it with a random value (here "sdc_4"). As before, we must ensure that the observation from Bank B1 is not included in the computation of the result nor the SDC. We could for example, use a dummy whose value would be one if the row is included in the MiFID and zero otherwise.

Example 5 - Merging GuV and MiFID

Similarly to our approach in Example 2, we could generate a "Year" variable in both datasets and use a **1:m full (outer) merge** via "BAID" and "Year" to combine the two datasets 13a and 13b. Table 13c shows the result.

Year	BAID	GuV_Value
2022	A1	100
2022	B1	200

(a) Simplified GuV

Date	Year	BAID	CP	MiFID_Value
2022-12-31	2022	A1	X1	20
2022-12-15	2022	A1	ZZ1	20
2022-12-31	2022	A1	ZZ1	20
2022-12-31	2022	C1	X1	50
2022-12-31	2022	C1	ZZ1	50
2022-12-31	2022	C1	ZZ1	50

(b) Simplified MiFID

Date	Year	BAID	GuV_Value	CP	MiFID_Value
2022-12-31	2022	A1	100	X1	20
2022-12-15	2022	A1	100	ZZ1	20
2022-12-31	2022	A1	**100**	ZZ1	20
2022-12-31	2022	B1	200		
2022-12-31	2022	C1		X1	50
2022-12-31	2022	C1		ZZ1	50
2022-12-31	2022	C1		ZZ1	50

(c) Merged dataset of simplified GuV and MiFID

Table 13: Example 5

Regarding the three potential SDC challenges we find that

- No rows were deleted.
- **Duplicates in the GuV** data were generated (but not in the MiFID). In comparison to the previous example, we see that the number of duplicates in "GuV_Value" actually *increased*. See Example 2 for our recommended approach.
- **Missing values** were generated in both the GuV and the MiFID data. See Example 4 for our recommended approach.

Example 6 - Merging GuV and MiFID

Similarly to our approach in Example 3, we could first collapse the MiFID data by "Year" and "BAID" (see Table 13b, where "sum_MiFID_Value" is now the sum per year) and then use a **1:1 full (outer) merge** via "BAID" and "Year". Table 14c shows the result.

Date	BAID	GuV_Value
2022-12-31	A1	100
2022-12-31	B1	200

(a) Simplified GuV

Year	BAID	CP	sum_MiFID_Value
2022	A1	X1	60
2022	C1	X1	150

(b) Simplified MiFID - collapsed

Year	BAID	GuV_Value	CP	sum_MiFID_Value
2022	A1	100	X1	60
2022	B1	200		
2022	C1		X1	150

(c) Merged dataset of simplified GuV and MiFID

Table 14: Example 6

Regarding the three possible SDC challenges we find that

- As we did **not** collapse the MiFID by **all SDC variables** (i.e. "BAID" and "CP"), we have already **lost rows** in Table 14b. Therefore, in Table 14c we cannot perform an SDC for "sum_MiFID_Value" as we do not have any information on the number and distribution of counterparties for which "sum_MIFID_Value" was calculated.
- No duplicates were generated.
- **Missing values** were generated in both the GuV and the MiFID data. See Example 4 for our recommended approach.

Due to the missing rows, an SDC for "sum_MiFID_Value" based on Table 14c is not possible!

Example 7 - Merging GuV and MiFID

In the previous example, we saw what happens when we do not collapse data in an SDC compliant way. This final example shows how the collapse should have been carried out using all SDC variables (i.e. "BAID" and "CP") in the MiFID as explained in Blaschke, Gomolka, et al. (2022). Table 15b shows the result of a collapse by "Year", "BAID" and "CP" and Table 15c shows the combined dataset after a **1:m full (outer) merge** via "Year" and "BAID". Note, that we have to use a 1:m merge again as Year-BAID does not uniquely identify a row in Table 15b.

Date	BAID	GuV_Value
2022-12-31	A1	100
2022-12-31	B1	200

(a) Simplified GuV

Year	BAID	CP	sum_MiFID_Value
2022	A1	X1	20
2022	A1	ZZ1	40
2022	C1	X1	50
2022	C1	ZZ1	100

(b) Simplified MiFID - collapsed

Year	BAID	GuV_Value	CP	sum_MiFID_Value
2022	A1	100	X1	20
2022	A1	100	ZZ1	40
2022	B1	200		
2022	C1		X1	50
2022	C1		ZZ1	100

(c) Merged dataset of simplified GuV and MiFID

Table 15: Example 7

Regarding the three potential SDC challenges we find that

- Rows from the non-collapsed MiFID dataset (see Table 15b) were **deleted**. However, as we collapsed in an SDC compliant way, this is **not problematic**.
- **Duplicates in the GuV** data were generated (but not in the MiFID). See Example 2 for our recommended approach.
- **Missing values** were generated in both the GuV and the MiFID data. See Example 4 for our recommended approach.

An SDC based on Table 15c is possible.

Glossary

Analysis dataset

The "**analysis dataset**" is the dataset that is used to compute results. Normally, the analysis dataset differs from the original dataset (see below) due to data transformation steps (e.g. dropping of variables and rows, merging with other datasets, or generation of additional variables).

Key variables

One or more variables (combined) enable the distinct identification of each cell of the other columns. As these variables are the key to identifying the other cells, we refer to them as "**key variables**". We refer to the key variables in their entirety as the "**set of key variables**". Database terminology may also call the set the "compound key" or "unit of observation". **Merge key variables** are key variables that are used to identify rows in a merge.

Original dataset

The "**original dataset**" is the unmodified dataset as provided by the RDSC.

SDC variables

Variables containing the IDs of sensitive entities (e.g. SYSNR, BAID or real names) are referred to as "**SDC variables**". As the classification of SDC variables is purely context-related and depends on whether the entities contained therein need to be protected or not, the RDSC specifies SDC variables for each original dataset in its data reports. These reports identify SDC variables as such for each individual dataset, which is why a variable may be an SDC variable for one original dataset but not for another.

Summary functions

Some functions may calculate results based on multiple values from multiple rows, which leads to an aggregation of information. Examples include sums or means across several rows. We refer to these functions as "**summary functions**".

Appendix 3 - Formal proof

In this section we provide a formal proof of the results presented in Table 8. The proof will start from the statement that merging does not lead to duplicates, empty values or loss of rows. We employ this strategy because it is easy to produce examples for the existence of duplicates, empty values or loss of rows. Thus, we will proof entries with "None". To do so, we utilise relational algebra described by Codd (1970) to describe datasets[23].

We will use the notation Π and σ for projection and selection. As merge notation: $\bowtie$, $\bowtie^{l}$, $\bowtie^{R}$ and $\bowtie^{O}$ are inner, left, right and outer merge. Additionally, we need w as the empty value. In all definitions and proofs we say R and S are relations. While, $A = \{C_1, ..., C_l, A_1, ..., A_n\}$ is the attribute set of R, $B = \{C_1, ..., C_l, B_1, ..., B_m\}$ is the attribute set of S. Let $C = \{C_1, ..., C_l\}$ be the shared attributes of R and S and let $\alpha \subseteq A$. Additionally, we need some own definitions to describe duplicates, empty values and loss of rows.

i. Definitions

Loss of rows

R loses rows from α with an operation $\star$ (this can be a join or aggregation), if:

$$LoR_{\alpha}(R) := \begin{cases} True & \text{if } \Pi_{\alpha}(R) \nsubseteq \Pi_{\alpha}(\star(R)) \\ False & \text{else.} \end{cases}$$

So if there is one element in the projection over α from R which is not included in the projection from the operation, there is a loss of rows.

Duplicates

A relation R has a duplicate in the columns $\alpha \subseteq A$, if:

$$Dup_{\alpha}(R) := \begin{cases} True & \text{if } \exists r^1, r^2 \in R : r^1_{[\alpha]} = r^2_{[\alpha]} \wedge r^1 \neq r^2 \\ False & \text{else.} \end{cases}$$

This seems complicated, but it is not. Imagine:

$R =$

A_1	A_2
1	X
2	X
3	Y

We can say R has a duplicate over $\alpha = \{A_2\}$, because there is $r^1 = (1, X)$ and $r^2 = (2, X)$, then $r^1_{[\alpha]} = X = r^2_{[\alpha]}$, but $r^1 = (1, X) \neq (2, X) = r^2$.

23 We refer readers interested in background information on merge and append operations in relational algebra to Studer (2016).

On the other side, there is no duplicate over A_1, because we do not find elements with $r^1_{[A_1]} = r^2_{[A_1]}$, without $r^1 = r^2$.

Empty values

We say R has empty entries in the columns $\alpha = \{A_k, ..., A_{k+1}\} \subseteq A$ if:

$$EE_\alpha(R) := \begin{cases} True & \text{if } \sigma_{\alpha=w}(R) \neq \emptyset \\ False & \text{else.} \end{cases}$$

If we select all empty values over the attributes α and we get a result which is not the empty set, then there are empty values over α.

ii. Proofs

We will only look at the left dataset and $1:1$ and $m:1$ relationships. W.l.o.g follows that the proof is valid for a right dataset with $1:1$ or $1:m$ relationships, if $\alpha \subseteq B$.

Loss of Rows (LoR)

Theorem LoR-1

R has no loss of rows under the operation $R \bowtie^L S$.

Proof: We need to show: $R \subseteq \Pi_A(R \bowtie^L S)$: Let $r \in R$. Here, we have two cases:

1. $r \in \Pi_A(R \bowtie S)$, then $r \in \Pi_A(R \bowtie^L S)$
2. $r \notin \Pi_A(R \bowtie S)$, then $r \notin \Pi_A(R \bowtie S)$. That followed, we can say: $r \in R - \Pi_A(R \bowtie S)$. Now, we only build a cross-product over attributes we do not care about. Therefore, $r \in \Pi_A(R \bowtie^L S)$.

Theorem LoR-2

S has no loss of rows under the operation $R \bowtie^S S$.

Proof: Same as LoR-1.

Theorem LoR-3

R and S have no loss of rows under the operation $R \bowtie^O S$.

Proof: Since we append $R \bowtie^L S$ and $R \bowtie^R S$ in $R \bowtie^O S$ and with LoR-1 and LoR-2, it follows that they both have no loss of rows. q.e.d.

Duplicates

Here, we need the $m:1,1:1$ relation between datasets. We can express them as $Dup_C(S) = False$. With this expression we say, that there are no duplicates in the shared attributes of the right dataset. So it is equal to $m:1,1:1$.

Theorem Dup-1 (Inner Merge)

If $Dup_\alpha(R) = False$ and $Dup_C(S) = False$ then $Dup_\alpha(R \bowtie S) = False$.

Proof: Let $t^1, t^2 \in R \bowtie S = \{(r, s_{[B_1,...,B_m]}) | r \in R \wedge s \in S \wedge r_{[C]} = s_{[C]}\}$, it follows that t^1 is as follows $t^1 = (r^1, s^1_{[B_1,...,B_m]})$, and the same for $t^2 = (r^2, s^2_{[B_1,...,B_m]})$.

Since R has no duplicates over α, we can say: $\forall r^1, r^2 \in R : r^1_{[\alpha]} = r^2_{[\alpha]} \wedge r^1 = r^2$. So the first part of t^1 and t^2 must also be equal. Now we know that $r_{[C]} = s_{[C]}$. Combined with no duplicates over C in S, it follows: $r^1_{[C]} = r^2_{[C]} \Rightarrow s^1_{[C]} = s^2_{[C]} \wedge s^1 = s^2$. That followed, we know $t^1 = (r^1, s^1_{[B_1,...,B_m]}) = (r^2, s^2_{[B_1,...,B_m]}) = t^2$. q.e.d.

Theorem Dup-2 (Left Merge)

If $Dup_\alpha(R) = False$ and $Dup_C(S) = False$ then $Dup_\alpha(R \bowtie^L S) = False$.

Proof: Let $t^1, t^2 \in R \bowtie^L S = (R \bowtie S) \cup ((R - \Pi_A(R \bowtie S)) \times \{(w, ..., w)\})$. So here we need to show three things:

1. $R \bowtie S$ has no duplicates in α
2. $(R - \Pi_A(R \bowtie S)) \times \{(w, ..., w)\}$ has no duplicates in α.
3. Appending $R \bowtie S$ and $(R - \Pi_A(R \bowtie S)) \times \{(w, ..., w)\}$ do not produce duplicates in α.
4. follows directly from Theorem Dup-1.

Let us now show 2.: We will start with $R - \Pi_A(R \bowtie S)$. First, we know that R has no duplicates over α from our conditions. Since we take the difference from R, we are just eliminating tuples and do not append any tuples. There will not be any duplicates. Formally: $r^1, r^2 \in R - \Pi_A(R \bowtie S) \Leftrightarrow r^1, r^2 \in R \wedge r^1, r^2 \notin \Pi_A(R \bowtie S)$, but still: $r^1, r^2 \in R$ and then with the precondition: $r^1_{[\alpha]} = r^2_{[\alpha]} \wedge r^1 = r^2$. The final step is the cross-product, since there were no duplicates before and the new attributes get all the same value w, so there are no duplicates at all. Formally: $t^1, t^2 \in (R - \Pi_A(R \bowtie S)) \times \{(w, ..., w)\}$, then $t^1 = (r^1, w, ..., w)$ and $t^2 = (r^2, w, ..., w)$, where $r^1, r^2 \in R - \Pi_A(R \bowtie S)$, then from $r^1_{[\alpha]} = r^2_{[\alpha]} \wedge r^1 = r^2 \Rightarrow t^1 = (r^1, w, ..., w) = (r^2, w, ..., w) = t^2$.

Now 3. remains. We want to show if $t \in R \bowtie S \Leftrightarrow t \notin (R - \Pi_A(R \bowtie S)) \times \{(w, ..., w)\})$. This is

obvious, from $R - \Pi_A(R \bowtie S)$. Formally:

$$\begin{aligned} t = (r, s_{[B_1,\ldots,B_m]}) \in R \bowtie S \Leftrightarrow & \quad r \in \Pi_A(R \bowtie S) \\ \Leftrightarrow & \quad r \notin R - \Pi_A(R \bowtie S) \\ \Leftrightarrow & \quad t \notin (R - \Pi_A(R \bowtie S)) \times \{(w, \ldots, w)\} \end{aligned}$$

Theorem Dup-3 (Right Merge)

If $Dup_\alpha(R) = False$ and $Dup_C(S) = False$ then $Dup_\alpha(R \bowtie^R S) = False$.

Proof: Same as Dup-2.

Theorem Dup-4 (Outer Merge)

If $Dup_\alpha(R) = False$ and $Dup_C(S) = False$ then $Dup_\alpha(R \bowtie^O S) = False$.

Proof: If t^1, t^2 are both in one side of $(R \bowtie^L S) \cup (R \bowtie^R S)$, then with Dup-2 and Dup-3 it automatically follows that there will not be any duplicates. So, we just need to be aware of whether one of them is only in the left side and the other one is only in the right side. So let w.l.o.g. be: $t^1 \in R \bowtie^L S$ and $t^2 \in R \bowtie^R S$ and $t^2 \notin R \bowtie^L S$. Now, we want $t^1_{[\alpha]} = t^2_{[\alpha]}$, but this is a contradiction, because if $t^2 \in R \bowtie^R S$, but not in $R \bowtie^L S$, then $t^2 \notin R \bowtie S$. So t^2 has to be an element of $(\{(w, \ldots, w)\} \times (S - \Pi_B(R \bowtie S))$. But then $t^2_{[\alpha]} = (w, \ldots, w)$, because $\alpha \subseteq A$. Finally, we have the a contradiction: Since $t^1 \in R \bowtie^L S$, we know that $t^1_{[\alpha]} \neq (w, \ldots, w)$. Therefore, it follows: $t^1_{[\alpha]} \neq t^2_{[\alpha]}$. q.e.d.

Empty Entries

Let us move to the empty entries. They are again independent of the key relationships.

Theorem EE-1 (Inner Merge)

If $EE_\alpha(R) = False$, if $EE_\alpha(R \bowtie S) = False$ too.

Proof: $EE_\alpha(R \bowtie S) = False$ is equivalent to: $\sigma_{\alpha=w}(R \bowtie S) = \emptyset$. So let us check:

$$\begin{aligned} \sigma_{\alpha=w}(R \bowtie S) &= \sigma_{\alpha=w}((r, s_{[B_1,\ldots,B_m]}) | r \in R \wedge s \in S \wedge r_{[C]} = s_{[C]}) \\ &\overset{\alpha \in A}{=} \sigma_{\alpha=w}(r | r \in R) \\ &\overset{EE_\alpha(R)=False}{=} \emptyset \end{aligned}$$

q.e.d.

Theorem EE-2 (Left Merge)

If $EE_{\alpha}(R) = False$, if $EE_{\alpha}(R \ltimes^{\perp} S) = False$ too.

Proof:

$$\sigma_{\alpha=w}(R \ltimes^{\perp} S) = \sigma_{\alpha=w}((R \bowtie S) \cup ((R - \Pi_A(R \bowtie S) \times \{(w, ..., w)\}))$$

The left side of the appending is free from empty values since EE-1. So we just need to look at the right side: $\sigma_{\alpha=w}((R - \Pi_A(R \bowtie S) \times \{(w, ..., w)\})$. Since $\alpha \subseteq A$, we just need to consider $\sigma_{\alpha=w}((R - \Pi_A(R \bowtie S))$. Because $EE_{\alpha}(R) = False$, we know $\sigma_{\alpha=w}((R - \Pi_A(R \bowtie S)) = \emptyset$. q.e.d.

References

Blaschke, J., Eiff, J., Friederich, K., & Krüger, M. (2022). *MFI interest rate statistics (ZISTA), Data Report 2022-13 – Metadata Version ZISTA-Data-Doc-v5-1*. Deutsche Bundesbank, Research Data and Service Centre. https://www.bundesbank.de/resource/blob/745204/64f993b6a9d5f50a180f2a5743184a46/mL/2022-13-zista-data.pdf

Blaschke, J., Gomolka, M., & Hirsch, C. (2022). *Statistical Disclosure Control (SDC) for results derived from aggregated confidential microdata, Technical Report 2022-01 – Version 1.0*. Deutsche Bundesbank, Research Data and Service Centre. https://www.bundesbank.de/resource/blob/745168/3501912ca41dc894f605a9d24413dee1/mL/2022-01-sdc-data.pdf

Cagala, T., Gomolka, M., Krüger, M., & Sachs, K. (2021). *Markets in Financial Instruments Directive (MiFID), Data Report 2021-13 – Metadata ID 1.0*. Deutsche Bundesbank, Research Data and Service Centre. https://www.bundesbank.de/resource/blob/634886/3414886ff73afe8ce02a009084b812e5/mL/2021-13-mifid-data.pdf

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, *13*(6), 377–387.

Doll, H., Gábor-Tóth, E., & Schild, C.-J. (2021). *Linking Deutsche Bundesbank Company Data, Technical Report 2021-05*. Deutsche Bundesbank, Research Data and Service Centre. https://www.bundesbank.de/resource/blob/624432/207c774d468e82d76ec19ef6bfa1c8a7/mL/2021-05-company-data.pdf

Gábor-Tóth, E., & Schild, C.-J. (2021). *Understanding Overlaps between Different Company Data, Technical Report 2021-06*. Deutsche Bundesbank, Research Data and Service Centre. https://www.bundesbank.de/resource/blob/624628/4a0ec7a2e5ad7bcfb9274abad8f5a885/mL/2021-06-company-data.pdf

Gomolka, M., Blaschke, J., & Hirsch, C. (2021). *Working with large data at the RDSC, Technical Report 2021-04 – Version 1.1*. Deutsche Bundesbank, Research Data and Service Centre. https://www.bundesbank.de/resource/blob/623988/62b8c17881d63bccc61efd11af0b47db/mL/2021-04-large-data-data.pdf

McKinney, W. et al. (2010). Data structures for statistical computing in python. *Proceedings of the 9th Python in Science Conference*, *445*, 51–56.

Research Data and Service Centre. (2021). *Rules for visiting researchers at the RDSC, Technical Report 2021-02 - Version 1-0*. Deutsche Bundesbank, Research Data and Service Centre. https://www.bundesbank.de/resource/blob/826176/ffc6337a19ea27359b06f2a8abe0ca7d/mL/2021-02-gastforschung-data.pdf

Schild, C.-J., Schultz, S., & Wieser, F. (2017). *Linking Deutsche Bundesbank Company Data using Machine-Learning-Based Classification, Technical Report 2017-01*. Deutsche Bundesbank, Research Data and Service Centre. https://www.bundesbank.de/resource/blob/604828/f3227be248777d3f414b89b395e875cd/mL/2017-01-company-data.pdf

Stahl, H., & Scheller, N. (2022). *Statistics of the banks' profit and loss accounts (GuV), Data Report 2022-20 – Documentation version 11*. Deutsche Bundesbank, Research Data and Service Centre. https://www.bundesbank.de/resource/blob/898994/edabfff3916d7a66db04c154476cfbe5/mL/2022-20-guv-data.pdf

StataCorp. (2021a). *Stata 17 Base Reference Manual - append*. College Station, TX: Stata Press. https://www.stata.com/manuals/dappend.pdf

StataCorp. (2021b). *Stata 17 Base Reference Manual - merge*. College Station, TX: Stata Press. https://www.stata.com/manuals/dmerge.pdf

Studer, T. (2016). *Relationale datenbanken*. Springer.

Wickham, H. (2014). Tidy data. *Journal of Statistical Software*, *59*(10), 1–23. https://doi.org/10.

18637/jss.v059.i10