Schema Evolution and Reproducibility of Long-term Hydrographic Data Sets at the IOW

Tanja Auge[1], Erik Manthey[1], Susanne Jürgensmann[2], Susanne Feistel[2], and Andreas Heuer[1]

[1] University of Rostock, Germany
{firstname.lastname}@uni-rostock.de
http://dbis.informatik.uni-rostock.de

[2] Leibniz Institute for Baltic Sea Research Warnemünde, Germany
{firstname.lastname}@io-warnemuende.de
https://www.io-warnemuende.de

Abstract. National and international exploration of the Baltic Sea ecosystem can be traced back to the 19th century. In its quite long history, the *Leibniz Institute for Baltic Sea Research Warnemünde* (IOW) is the only research institution in Germany that has made interdisciplinary research of the Baltic Sea its central mission. The IOW hosts data from more than 130 years of research work.
Using the example of hydrographic datasets that have been created over a period of about 50 years, this paper examines changes in the data and the associated schemes that have resulted from the continuous development and refinement of measurement methods over time. The paper focuses on the schema development operators: What kind of schema development has taken places over the years, and what are the important basic schema development operators that can be identified? It classifies well-known schema evolution operators which can be expressed as schema mappings, and defines two new operators for merging and splitting attributes, up to now not considered in other research works. These operators have proven to be essential for development of a new universal schema for the central oceanographic database on the IOW – the *IOWDB*.

Keywords: Schema Evolution · Baltic Sea · Long-term Data · Research Data Management.

1 Introduction

National and international exploration of the Baltic Sea ecosystem can be traced back to the 19th century. In its quite long history, the Leibniz Institute for Baltic Sea Research Warnemünde (IOW) is the only research institution in Germany that has made interdisciplinary research of the Baltic Sea its central mission. The IOW hosts data from more than 130 years of research work. Due to their origin they were stored in various formats and on diverse storage media.

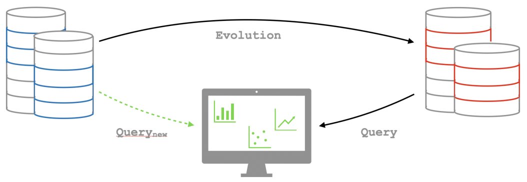

Fig. 1: Unification of Query Evaluation, Provenance and Evolution [1]

The optimal preparation of such research data is part of the so-called *FAIR principles* (**F**indable, **A**ccessible, **I**nteroperable, **R**eusable). These principles define "characteristics that contemporary data resources, tools, vocabularies and infrastructures should exhibit to assist discovery and reuse by third-parties" [12]. They should ensure sustainable research data management by processing the data and their metadata.

The IOW publishes many of its newer data online on IOWMeta[3], realizing the F and A of FAIR. We are therefore concentrating on interoperability (I) and reusability (R) to ensure the reproducibility of the data evaluations. To be able to determine exactly those research data that have an influence on the evaluation result, we have to automatically compute the evaluation queries, the scheme and data evolution stages as well as the provenance queries in a homogeneous framework (see Fig. 1). This is important not only for the reproducibility of evaluation results, but also for the updating of evaluations over time.

In this paper, we focus on the process of data collection with a so-called *CTD probe* (see Section 3) in the period from the 1970s to the present. Within this time, the scientific requirements changed dramatically and were accompanied by significant improvements in instrumentation, data acquisition and processing on board of the research vessels and, last but not least, data storage on land.

Within the scope of hydrographic data collection with a CTD probe, new sensors were developed or existing ones improved. Correspondingly new measurement parameters were defined or replaced. The measurements could then be carried out faster and more accurately. Therefore, not only the physical storage media used at the IOW developed enormously during this time, from simple paper records, punch cards and magnetic tapes to modern databases (see Fig. 2), but also the underlying concepts for data processing and structuring had to be reconsidered continuously.

However, this change in the data formats and structures leads to problems nowadays when newer evaluations of the data have to be performed on old data sets or long-term data ranging over some decades [4]. Thus, one goal is the reproducibility of old results with the help of a new evaluation technique.

An evaluation of the data over the entire period of time and the tracing of individual tuples is only possible with considerable effort. In database terms the development of the new structures can result in the creation of new attributes or tables as well as a reassignment of the affected tuples. It is possible that

[3] https://iowmeta.io-warnemuende.de

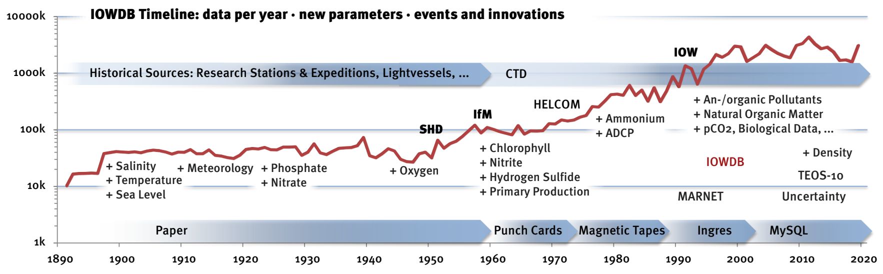

Fig. 2: (Logarithmic) Data increase at IOW [3], showing only the new data per year

the variety of measured variables changes over time. Thus, past and current schemas differ significantly in this respect as well. Columns are renamed, created or deleted. Merging or splitting of attributes can be observed.

To be able to support reproducibility over time, we have to combine (1) the data evaluations (represented as database queries including some linear algebra functions), (2) the schema evolution steps, and (3) the process of inverting these steps by means of data provenance techniques (***why-*** and ***how-***provenance). Since we already developed formal techniques based on schema mappings (such as s-t tgds, i.e. source-to-target tuple-generating dependencies) [8] for steps (1) and (3), we have to integrate schema evolution steps by the use of schema mappings, as well [2]. This can be done by using formally defined schema modification operators such as the ones introduced by the PRISM++ project [5].

The combination of data provenance with schema and data evolution enables us to evaluate provenance queries on evolving data and schemas. Through inverse evolution steps, the new database can be transferred to the old schema. Formally, we use the CHASE algorithm, a universal tool of database theory. Our application of the CHASE for schema (and corresponding instance) mappings is based on the s-t tgds mentioned above. It is shown in [5], that the *Schema Modification Operators* (SMOs) and the *Integrity Constraint Modification Operators* (ICMOs) can be transformed to such kind of schema mappings.

In Section 2 we first introduce the original SMOs and ICMOs of PRISM++. In this paper, we focus on the analysis of the schema evolution process at the IOW, that is briefly presented in Section 3. We identify the most important SMOs and ICMOs which are necessary to describe the IOW schema evolution, and define two new operators `MERGE Column` and `SPLIT Column` (see Section 4).

2 State of the Art

Our work is based on three different schema definitions: Schema Mapping, Schema Evolution and Schema Operator. We understand *Schema Mapping* as a concrete evaluation query, *Schema Evolution* as the overall process of evolution and *Schema Modification* as the individual step itself.

Table 1: Schema Modification Operators, based on [6, 10]

SMOs
`COPY Table` R `INTO` S
`CREATE Table` R(a,b,c)
`DECOMPOSE Table` R(a,b,c) `INTO` S(a,b), T(b,c)
`DISTRIBUTE Table` / `PARTITION Table` R `INTO` S `WITH` cond, T
`DROP Table` R
`JOIN Table` R, S `INTO` T `WHERE` cond
`MERGE Table` R, S `INTO` T
`RENAME Table` R `INTO` S

SMOs
`ADD Column` d [`AS` const|func(a,b,c)] `INTO` R
`COPY Column` c `FROM` R `INTO` S `WHERE` cond
`DROP Column` c `FROM` R
`MOVE Column` c `FROM` R `INTO` S `WHERE` cond
`RENAME Column` b `IN` R `TO` d
`NOP`

Schema Mapping: *Schemas* and *Schema Mappings* are two fundamental components of heterogeneous data management. While schemas specify the structure of the various databases, schema mappings describe the relationships between them. Schema mappings can be used in particular to transform data between two different schemas.

Schema Evolution: *Schema evolution* describes the schema changes of a database over time. It therefore give a description to the overall process. This includes changes in the relational schemas themselves as well as changes in the key and integrity conditions. In order to enable earlier states to be analyzed retrospectively and queries on other schema versions to be made, starting from the current database state, PRISM++ [5, 6, 10] introduced special operators which describe the changes in schemas and integrity conditions in a compact way. For defining these operators, the authors examined several schemas from practice concerning their modifications [6]. The most frequently occurring operators formed the core of these *Modification Operators* which are divided into *Schema Modification Operators* (SMOs) and *Integrity Constraint Modification Operators* (ICMOs). While SMOs can describe the direct changes of relations, such as the creation or deletion of individual attributes or entire relations, ICMOs are used whenever key or integrity conditions in relations change. Thus, these conditions can be generally tightened or weakened by adding or discarding restrictions.

Schema Modification Operators (SMOs): The so called *Schema Modification Operators* are basic operators used to describe changing database schemas. Thirteen operators are defined in [6, 10], summarized in Table 1. Each operator captures an atomic change and by combining them, it is possible to express complex evolutions. The most common ones are `CREATE Table`, `DROP Table`, `ADD Column`, `DROP Column` and `RENAME Column` [11, 13]. These also correspond to the operators relevant to the IOW, gray highlighted.

Integrity Constraints Modification Operators (ICMOs): In addition to the SMOs, there is another class of Schema Evolution Operators. These *Integrity Constraints Modification Operators* (ICMOs), first introduced in [5], describe changes in integrity conditions. More precisely, the six ICMOs handle the

Table 2: Integrity Constraints Modification Operators, based on [5]

ICMOs
`ALTER Table` R `ADD PRIMARY KEY` pk1(a,b) <policy>
`ALTER Table` R `ADD FOREIGN KEY` fk1(c,d) `REFERENCES` T(a,b) <policy>
`ALTER Table` R `ADD VALUE CONSTRAINT` vc1 `AS` R.e = "0" <policy>
`ALTER Table` R `DROP PRIMARY KEY` pk1
`ALTER Table` R `DROP FOREIGN KEY` fk1
`ALTER Table` R `DROP VALUE CONSTRAINT` vc1

three restrictions `PRIMARY KEY`, `FOREIGN KEY`, and `VALUE CONSTRAINT`. Attention must be paid to the Enforcement Policy <policy>, which is necessary for the first three ICMOs. Here we can choose between

(1) `CHECK`: The system checks whether the current table satisfies the new primary key. If not, no primary key is added.
(2) `ENFORCE`: The primary key is added in any case. Tuples that violate the new key are deleted.

However, before we begin to investigate the schema evolution, we will briefly introduce the IOW in general as well as the analyzed data sets of the IOW.

3 Research at the IOW

National and international exploration of the Baltic Sea ecosystem can be traced back to the 19th century. In its quite long history, the Leibniz Institute for Baltic Sea Research Warnemünde is the only research institution in Germany that has made interdisciplinary research of the Baltic Sea its central mission. The IOW hosts data from more than 130 years of research work. Due to their origin they were stored in various formats and on diverse storage media.

Leibniz Institute for Baltic Sea Research Warnemünde (IOW): The story of the IOW goes back to the German Democratic Republic (GDR) of the 1950s. At that time, the Seehydrographic Service of the GDR was founded in order to be able to operate independently of the west-German equivalent. It included, among other things, a separate department for oceanography, which was known as the *Institute of Oceanography Warnemünde* (IfM-W) in 1960. The institute achieved international recognition through independent research cruises and was able to establish itself as a permanent fixture in worldwide marine research.

The systematic exploration of the Baltic Sea finally began in 1964 with the *International Synoptic Recording of the Baltic Sea*. In the following decades, these regular scheduled cruises, i.e. cruises at fixed times and places, became an important part of Baltic Sea research within the IfM-W.

After the German reunification in 1990, research and science in the whole of Germany had to be newly regulated and ordered, which led to the foundation of the *Institute for Baltic Sea Research Warnemünde*[4] in 1992.

[4] https://www.io-warnemuende.de

IOWDB: The *Oceanographic Database of IOW* (IOWDB) had originally been designed for particular internal requirements of the IOW. The IOWDB has always been aimed at the management of historical and recent oceanographic measurements.

Research cruises have been conducted since 1949, and their data systematically collected. The post-processing and analysis of the resulting observational data was carried out by varying methods and with changing quality. A substantial fraction of legacy data was successively transferred to modern storage media, their quality was controlled and, if possible and necessary, improved by detailed individual scientific review.

The content of the database includes oceanographic readings and metadata (mainly Baltic Sea) from 1877 to 2020 obtained during 951 research campaigns of the IOW and cooperating institutions. As of June 2020, the IOWDB contains more than 78 million measured samples representing georeferenced point data from the water column, primarily from CTD profiles, hydrochemical and biological sampling, current-meter time series, trace metal sampling and long-term monitoring. Phyto- and zooplankton data are available for 1988 to 2018.

CTD measurement: One third of the stored data in the IOWDB are obtained with a so-called *CTD probe*. Primary parameters of this instrument are **C**onductivity (electrical conductivity, which is used to determine salinity), water **T**emperature and **D**epth, which is determined by the prevailing pressure.

Optional sensors, e.g. for oxygen, fluorescence and other parameters may be added. The CTD probe is surrounded by several water samplers, allowing additional water samples to be taken. These are then used for further analyses.

The examined CTD data originates from regularly conducted monitoring cruises. On these cruises fixed locations at the Baltic Sea are travelled, so the changes of CTD values can be recorded. After first recording the primary data from the cruise, the resulting records are validated. In this case validation means that the values from missing depths are interpolated to get an approximated value. Within this paper's research only these validated data was examined.

Next we will take a closer look at the schema evolution at the IOW. We focus here on the evolution of the CTD data.

4 Schema Evolution at IOW

We describe the schema and the corresponding changes over a period from 1977 to the present day solely for the data resulting from CTD measurements. It should be noted that adding measurement data of a new type would of course result in a new schema.

4.1 Schema Changes over the Years

Although data from CTD measurements exist from the early 1950s on, the data files from the years before 1977 are very difficult to interpret. Essentially, the

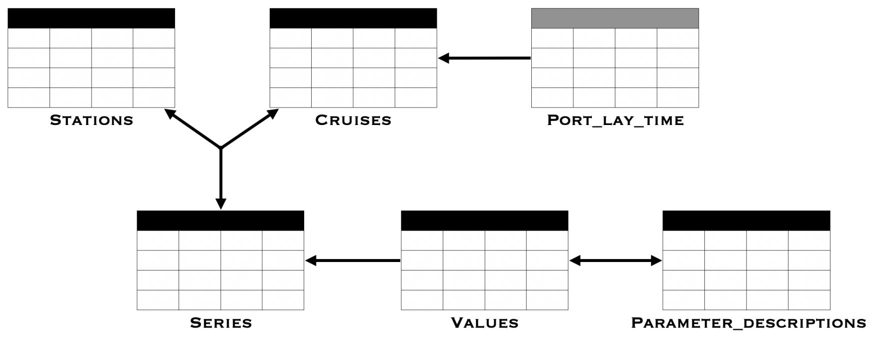

Fig. 3: Universal schema from 2017

files contain a large amount of measured values, but there are hardly any descriptions or parameter names and thus insufficient schematic information. Without historical background knowledge of the underlying process of data collection, only vague assumptions can be made to identify the underlying scheme. For this reason, we start with a more detailed investigation in 1977.

Because no structural changes occurred during each period, the schema changes over the years can be divided into three blocks: one period between 1977 and 1996, another between 1997 and 2016 and the years after 2017 [9].

1977-1996: The schema is clearly defined and divided into five relations describing the metadata of the Cruises, of the measurement Stations, the Series, as well as the actual recorded CTD Values and the corresponding Parameter_descriptions. The five relations are connected by three relationships (see Fig. 3): In case of CTD measurements there is a trivalent relationship (⇜⇖⇘) between the cruise, station and series relations. The relation with the CTD values is dependent (←) on the relation containing information about the measurement series. And the relation with the parameter descriptions is connected to the relation with the CTD values by a "belongs to" relationship (↔). The set of attributes corresponding to the relations and relationships is quite limited compared to the schemas of the following years. For example, in 1977 the schema contained only 34 attributes, whereas in 1997 it already contained 61 attributes, and the trend is rising.

1997-2016: However, the 1997 scheme brings some major changes. The previous entities remain unchanged, but we can record a lot of new attributes, a new relation (highlighted in Fig. 3) and associated changes in integrity conditions. Thus, in the Series and Cruises relations, the attributes to be stored increase from 15 to 26 and from 5 to 17 attributes, respectively. The new relation Port_lay_time records the laytime of a research vessel in a port and is therefore dependent on the Cruise relation. Stations, Values and Parameters_Description remain unchanged. Furthermore, some attributes are split while others are merged. All in all these are some minor and major changes. We will discuss this in more detail later on.

After 2017: The previous structure of the schemas from 1997 to 2016 remains unchanged, only seven attributes are added to the relations. Overall, there are no major changes and all schemas described previously can be mapped to it. The 2017 schema is therefore used as the universal schema, shown in Fig. 3.

4.2 Specification of Schema and Data Changes

Operations that are repeatedly performed when converting schemas before 2017 include, in particular, the addition, merging and splitting of attributes (see Table 3 (a)). Since the schemas gain more information over the years, this is not surprising. This continuous process of expansion is only supplemented a few times by renaming individual attributes, creating the new table PORT_LAY_TIME and adapting related integrity constraints in 1997. However, we will investigate this further.

Most evolution steps can be described easily using the SMOs and ICMOs defined for PRISM++ [9]. Only merging and splitting attributes requires a new operator can be understood as a sequence of the basic operators `ADD Column` and `DROP Column`. They are defined in Section 4.3. Because of their special importance at the IOW they shall be defined here as separate operators. But first of all let's have a look on the obvious schematic changes.

Adding and deleting attributes: These types of schema changes can be identified by the SMOs `ADD Column` or `DROP Column`. Adding attributes is one of the easiest described schema changes and occurs comparatively often but least frequently. There are almost fifty new attributes in total. This observation is also consistent with the results of other studies such as [7]. Here 38.7% of all schema changes were due to the addition and 26.4% to the deleting of attributes.

Looking at the relation SERIES, we observe the almost complete substitution of all associated attributes. However, contrary to our initial assumption, these attributes are not deleted irreversibly. They are rather merged into new tuples or split into new attributes. This results in the fact that we have to develop the two new operators `MERGE Column` and `SPLIT Column`.

Table 3: Detected schema changes at IOW

	# changes	percentage share
`CREATE Table`	1	1.6%
`ADD Column`	45	72.6%
`DROP Column`	15	24.2%
`RENAME Column`	1	1.6%

(a) with basic SMOs

	# changes	percentage share
`CREATE Table`	1	2.3%
`ADD Column`	35	79.5%
`DROP Column`	0	0%
`RENAME Column`	1	2.3%
`MERGE Column`	4	9.2%
`SPLIT Column`	3	6.8%

(b) extended with `MERGE Column` and `SPLIT Column`

Example 1. The SMO `ADD Column` extends Cruises by the possibility of a comment, formally :

```
ADD COLUMN Comment INTO Series.
```

Creating relations: This form of schema change occurs only once in the data sets examined. Since 1997, the relation Port_lay_time is part of the schema. It should be noted that in addition to the SMO `CREATE Table` itself, additional integrity conditions must be specified. On the one hand, it is necessary to create a primary key. On the other hand, due to the dependence on Port_lay_time and Cruises, a suitable foreign key must be specified.

Adding and deleting primary keys: Creating relation Port_lay_time implies the requirement of a specialized attribute, the so-called primary key, which clearly identifies the tuples of the new relation Residence. Furthermore, when mapping the schema from 1977 to the universal schema, the primary key of Cruises is changed. This is equivalent to deleting the old primary key and adding a new one.

The primary key can be modified using the ICMOs `ADD PRIMARY KEY` or `DROP PRIMARY KEY`. When creating a primary key, the enforcement policy must be observed as well. This checks whether individual tuples violate the new integrity condition or not. If so, the ICMO will not be applied (`CHECK` policy), if not, all tuples violating the newly introduced constraint are removed (`ENFORCE` policy). Accordingly, the primary key should always be created directly after or during the creation of the new table.

Adding foreign keys: Since the new relation Port_lay_time is dependent on Cruises, a foreign key must be specified additionally to the primary key. This is derived as usual from the primary key of the higher-level relation.

Example 2. Let us take a closer look on the relation Port_lay_time. Introduced in 1997, it is dependent on Cruises. Besides the creation of the relation itself using the SMO

```
CREATE Table Port_lay_time
        (ArchiveNo, Reason_for_stay, Date, Harbour, Duration),
```

a primary key consisting of the attributes *ArchiveNo*, *Date*, *Harbour* and *Duration* must be defined and the foreign key condition of *ArchiveNo* be specified. Adding key and integrity constraint can be realized using the ICMOs

```
ALTER Table Port_lay_time
ADD PRIMARY KEY pkLZ
        (ArchiveNo, Date, Harbour, Duration)
CHECK
```

and

```
ALTER Table Port_lay_time
ADD FOREIGN KEY fk (ArchiveNo) REFERENCES Stations(ArchiveNo)
CHECK.
```

Table 4: SMOs for merge and split

MERGE Column a,c AS func(a,c) IN R TO d
ADD Column d AS func(a,c) INTO R;
DROP Column a FROM R;
DROP Column c FROM R;

SPLIT Column a IN R TO d USING func_1, e USING func_2
ADD Column d AS func_1(a) INTO R;
ADD Column e AS func_2(a) INTO R;
DROP Column a FROM R;

Renaming an attribute: Also the SMO `RENAME Column` exists only a few times. Even if this looks rather harmless on first sight, it must be ensured that the old evaluations can no longer be easily reproduced on the new schema. It occurs for example in the evolution of the relation SERIES.

Example 3. When mapping the schema from 1997 to the universal schema, the Attribute *ws* is renamed to *ws-ID*. This can be described by:

```
RENAME COLUMN ws IN Series TO ws-ID.
```

Merging and splitting attributes: In the course of the years and decades, attributes are repeatedly summarized at the IOW. According to PRISM++ there is no independent Schema Modification Operator for these changes. So let us next define these two operators `MERGE Column` and `SPLIT Column`.

Example 4. The aim of the two new operators is a representation of the form

```
MERGE Column BeginDate AS func(StartYear,StartMonth,StartDay)
    INTO Series.
```

and

```
SPLIT Column Date IN Series TO
    StartDate USING func1(Date), EndDate USING func2(Date).
```

4.3 `MERGE Column` and `SPLIT Column`

Besides the schema changes, which can already be described by the operators defined in [5] and [6], there are at least two more, which can be expressed by composites of these and adding one or more functions. These additional changes consist in the merging and splitting of columns as seen in Table 4. At the IOW they occur, among other things, when time and date columns are changed.

`MERGE Column`: As seen in Table 4, merging two columns into a new column is realized by executing `ADD Column` and `DROP Column` one after the other. For creating the new attribute values a function `func` is used, which combines the old attributes. Accordingly, merging consists of creating a new column and then deleting the old ones.

In the case of relation SERIES the three columns *StartYear*, *StartMonth* and *StartDay* are merged into the new column *BeginDate*. sThe SMO `MERGE Column` can be interpreted as

```
ADD Column BeginDate AS func(StartYear,StartMonth,StartDay)
    INTO Series
DROP Column StartYear FROM Series;
DROP Column StartMonth FROM Series;
DROP Column StartDay FROM Series.
```

Assuming that the date type of the attribute *BeginDate* is a string of the format DD.MM.YYYY, we can choose the function `func` as concatenation of the form

```
func := CONCAT(StartDay,'.',StartMonth,'.',StartYear).
```

However, this choice is in no way binding.

SPLIT Column: Splitting a column has a similar structure. Here too, new columns are created and old ones deleted. One major difference, however, is that each new column requires its own function, implemented in SQL, for example.

So, in the case of CRUISES, the column *Date* is split into *StartDate* and *EndDate*. The SMO `SPLIT Column` can be interpreted as

```
ADD Column StartDate AS func1(Date) INTO Cruises;
ADD Column EndDate AS func2(Date) INTO Cruises;
DROP Column Date FROM Cruises.
```

Assuming the format DD.MM.YYYY we can divide the string into two substrings of length 10 using the functions

```
func1 := SUBSTRING(Date,1,10),
func2 := SUBSTRING(Date,12,10).
```

This static approach is only a simplified example because of the high error rate. Date values may have been saved incorrectly (e.g. typing errors) or in a different format (e.g. American date format). Functions that automatically split the source string are of course better, but would go beyond the scope of this example.

In total, we could identify seven merge and split operations. These contain 10 times `ADD Column` and 15 times `DELETE Column`. Compared with the previous studies we get the new distribution shown in Table 3 (b). The most common operator remains `ADD Column`, but we no longer need the `DROP Column` operator, at least not explicitly. Metadata is often given as strings or intervals. Since intervals were repeatedly represented as a single attribute or as interval boundaries (divided into two attributes), further changes are expected in the future.

4.4 Implementation

Even though the selection of our operators is based on the works [5] and [6], we have not implemented them in the PRISM++ system. We have decided for an implementation in a schema modification interface to MySQL which is already used at the IOW. All operators from Section 4.2 are implemented as described above. The auxiliary functions $\texttt{func}_i$ mentioned in Section 4.3 were implemented with appropriate `Update` operations, so we did not have any problems with the prototypical implementation here either.

5 Conclusion and Future Work

In this paper we classified – using CTD data as an example – the schema evolution operators relevant for research data management at the IOW. In particular, the addition of attributes using `ADD Column` as well as the merging and splitting of attributes were considered relevant. We have redefined the required operators `MERGE Column` and `SPLIT Column` based on the existing SMOs of [5].

To support reproducibility over time, we must combine (1) data analysis, (2) schema development steps, and (3) the process of reversing these steps using data prevention techniques. Since we have already developed formal techniques based on schema mappings for steps (1) and (3), the schema evolution steps must also be integrated by using schema mappings [1, 2]. Now, with our extended PRISM++ approach, we can apply the evolution operators expressed as s-t tgds against the given research database using the CHASE algorithm. And the IOW would thus be able to track the oxygen content of the Baltic Sea as well as other interesting parameters over a period of decades.

References

1. Auge, T: Extended Provenance Management for Data Science Applications. PhD@VLDB, CEUR Workshop Proceedings, CEUR-WS.org (2020)
2. Auge, T.; Heuer, A.: Combining Provenance Management and Schema Evolution. IPAW, 222–225 (2018)
3. Bock, S.; Feistel, F.; Jürgensmann, S.: Data Management at IOW. Poster (2014)
4. Bruder, I.; Klettke, M.; Möller, M. L.; Meyer, F.; Jürgensmann, S.; Feistel, S.: Daten wie Sand am Meer - Datenerhebung, -strukturierung, -management und Data Provenance für die Ostseeforschung. Datenbank-Spektrum **17**(2), 183–196 (2017)
5. Curino, C.; Moon, H. J.; Deutsch, A.; Zaniolo, D.: Update Rewriting and Integrity Constraint Maintenance in a Schema Evolution Support System: PRISM++. PVLDB **2**(4), 117–128 (2010)
6. Curino, C. A.; Moon, H. J.; Zaniolo, C.: Graceful Database Schema Evolution: the PRISM Workbench. PVLDB **1**(1), 761–772 (2008)
7. Curino, C. A.; Tanca, L.; Moon, H. J.; Zaniolo, C.: Schema Evolution in Wikipedia: Toward a Web Information System Benchmark. ICEIS(1), 323–332 (2008)
8. Fagin, R.; Kolaitis, P.G.; Popa, L.; Tan, W.C.: Schema Mapping Evolution Through Composition and Inversion. In: Schema Matching and Mapping, Springer, 191–222 (2011)
9. Manthey, E.: Beschreibung der Veränderungen von Schemata und Daten am IOW mit Schema-Evolutions-Operatoren. Bachelor Thesis, University of Rostock (2020)
10. Moon, H. J.; Curino, C.; Deutsch, A.; Hou, C.-Y.; Zaniolo, C.: Managing and Querying Transaction-time Databases under Schema Evolution. PVLDB **1**(1), 882-895 (2008)
11. Qiu, D.; Li, B.; Su, Z.: An Empirical Analysis of the Co-evolution of Schema and Code in Database Applications. ESEC/SIGSOFT FSE, ACM, 125–135 (2013)
12. Wilkinson, M.; Dumontier, M.; Aalbersberg, I. et al.: The FAIR Guiding Principles for scientific data management and stewardship. Sci Data 3, 160018 (2016)
13. Wu, S.; Neamtiu, I.: Schema evolution analysis for embedded databases. ICDE Workshops, IEEE Computer Society, 151–156 (2011)