Data Anamnesis: Admitting Raw Data into an Organization

Sebastian Kruse, Thorsten Papenbrock, Hazar Harmouch, and Felix Naumann
Hasso Plattner Institute, 14482 Potsdam, Germany
firstname.lastname@hpi.de

Abstract

Today's internet offers a plethora of openly available datasets, bearing great potential for novel applications and research. Likewise, rich datasets slumber within organizations. However, all too often those datasets are available only as raw dumps and lack proper documentation or even a schema. Data anamnesis is the first step of any effort to work with such datasets: It determines fundamental properties regarding the datasets' content, structure, and quality to assess their utility and to put them to use appropriately. Detecting such properties is a key concern of the research area of data profiling, which has developed several viable instruments, such as data type recognition and foreign key discovery.

In this article, we perform an anamnesis of the MusicBrainz dataset, an openly available and complex discographic database. In particular, we employ data profiling methods to create data summaries and then further analyze those summaries to reverse-engineer the database schema, to understand the data semantics, and to point out tangible schema quality issues. We propose two bottom-up schema quality dimensions, namely conciseness and normality, that measure the fit of the schema with its data, in contrast to a top-down approach that compares a schema with its application requirements.

1 Engaging in Raw Data

Data are a key driver in nowadays businesses as they are involved in most operational and strategic processes, such as decision-making and market predictions. For various companies, such as news agencies, booking platforms, and map providers, the data themselves are the main requisite to offer their services. Gartner also argues that companies should value their data as an asset [17]. Consequently, more and more data are being collected in today's world. Many datasets are published on the internet, with the Linked Open Data Cloud and open government data being well-known representatives. Companies have also started to gather their raw datasets in so-called *data lakes*. However, the mere availability of datasets is not sufficient to employ them for any project or application. Before any use, it is crucial to *understand* what content datasets actually contain, how to query them, and what implicit properties they have. Without these metadata, the actual data are basically worthless. Unfortunately, proper documentation of datasets is scarce. In fact, many raw datasets do not even provide a definition of their schema. Moreover, even when documentation is available, it might be imprecise, incomplete, or outdated. We claim that practically all datasets obtained by a company, data scientist, or anyone else, must undergo an adoption process – *data anamnesis* – before being put to use.

In medicine, the concept of anamnesis, which is Greek and literally means reminiscence, has been practiced for several centuries. When treating a patient for the first time, the doctor systematically asks that patient directly for the medical history to obtain a comprehensive view and determine any information relevant to the patient's treatment, such as current conditions, blood pressure, etc. A similar proceeding can be applied for data management. Here, a user (≈ the doctor) faces an unknown dataset (≈ the patient) that he or she wants to employ, e.g., for data integration or data mining. However, assume that the data do not come with a schema or other documentation or that one simply cannot trust the given metadata. The dataset, therefore, appears as an unusable black box. To cope with this situation, the user needs to determine relevant dataset properties and establish an understanding (≈ the anamnesis) directly from the data.

Definition 1 (Data anamnesis): Data anamnesis describes the process and result of discovering and assessing the structure of a raw (semi-)structured dataset. This discovery process is data-driven, i.e., all insights of the data anamnesis should come directly from the dataset so as to ensure that they accurately describe the data.

In medicine, there are standardized proceedings and questionnaires to obtain a patient's anamnesis. Analogously, frameworks for database reverse engineering have been conceived, such as [11]. Such frameworks rather focus on expert interviews and associated application code analysis, which might yield imprecise, incomplete, or incorrect insights for several reasons. For instance, an expert's knowledge about data might be outdated, and application code might not cover all relevant aspects of a dataset. Furthermore, the data themselves might be erroneous or incomplete and therefore do not adhere to alleged characteristics. These examples demonstrate the necessity to inspect the *actual* dataset at hand, instead of (or in addition to) deliberating on a *supposed* dataset.

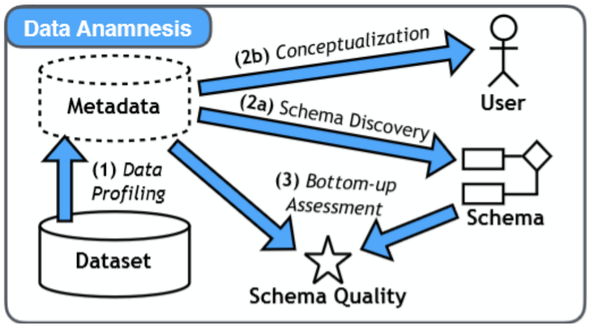


Figure 1: Data anamnesis enables the usage of raw datasets by means of data profiling and analyzing the acquired metadata.

To this end, we propose a data anamnesis process as shown in Figure 1. At first, the dataset in question should be analyzed for relevant metadata (Step (1)). Metadata discovery is a computationally complex task that has been tackled in the area of *data profiling* for at least two decades and still has many open questions [1]. The produced metadata can then be used to derive a technical schema definition for the data (Step (2a)) and should be explored to obtain an understanding of this schema (Step (2b)). Both are prerequisite to work with the data. Data anamnesis is also important in cases where documentation or a schema definition is given to diagnose *schema quality issues* (Step (3)): Do schema and data match? Are the data types and formats suitable? Are the constraints sound and complete? When Coelho et al. analyzed 512 open-source database schemata, they found these questions insufficiently answered by the provided metadata and attested overall poor schema quality [8]. So apart from data quality, which is out of the scope of this paper, we observe schema quality to be an important building block to assess the fitness for use of a given dataset. Given only the data and a schema, specified or discovered, data anamnesis judges schema quality from the viewpoint of the data. As the counterpart of data quality, which describes to what extent data fulfill (schema) constraints, this data-driven schema quality is the degree to which a schema fits its data. We refer to this approach as *bottom-up* schema quality assessment contrary to the traditional top-down assessment that judges schemata from the viewpoint of users, applications, and requirements. In particular, we propose two schema quality dimensions in this paper, namely conciseness and normality, and use metadata to detect suspicious parts of schemata that violate these.

To make the notion of data anamnesis tangible, we carry out an anamnesis for the openly available dataset MusicBrainz [18]. This user-maintained music encyclopedia collects a wide range of information from songs over relationships of album releases to acoustic fingerprints. Moreover, it is not only used in practice by multiple applications but also is the subject of various research projects. In addition, MusicBrainz' complexity is challenging: Its published export[1] consists of 324 headerless CSV files with a data volume of 35.1 GB, but no schema definition. Fortunately, in this case, the dataset's schema can be extracted from the MusicBrainz Server application source code[2], which allows us to evaluate the quality of our data anamnesis. Besides data types, the schema specifies various constraints, e.g., string formats, number ranges, keys, and foreign keys. Simultaneously, we shall diagnose the quality of this schema – i.e., determine how well it fits its data.

The remainder of this paper is organized as follows: In Section 2, we present the research area of *data profiling*, which addresses the problem of extracting metadata from given datasets, and apply state-of-the-art data profiling algorithms to the MusicBrainz dataset. In Section 3, we employ the discovered metadata to derive the dataset's schema. This step effectively turns the raw data into a queryable dataset. Section 4 then assesses the two aforementioned schema quality aspects, conciseness, and normality, for the dataset. Finally in Section 5, we use the gained insights to discuss further challenges in the area of data profiling to improve data anamnesis.

2 Data Profiling for Data Anamnesis

Descriptive metadata, such as integrity constraints, reveal inherent properties of a dataset and are useful for the preparation of various data-oriented tasks, e.g., query processing, data mining, and data integration [10]. As proposed in the previous section, metadata furthermore help to reconstruct database schemata and diagnose schema quality issues and are, thus, the key enabler for data anamnesis. Data profiling is the act of determining such descriptive metadata for a given dataset, including *single-column statistics*, such as data types and value distributions of columns, and *multi-column dependencies*, such as inclusion dependencies and functional dependencies. While the former can be obtained with most commercial data profiling tools, the latter are a focus in research [1]. As such, data profiling is the first step of data anamnesis. To this end, we describe two data profiling platforms that bundle important state-of-the-art profiling techniques and their produced metadata. Then, we apply these to the MusicBrainz dataset and report on the results, which form the input for the next data anamnesis steps.

2.1 Metanome and the Metadata Management System

For the analysis of the MusicBrainz dataset, we utilized two data profiling frameworks: Our *Metanome* tool [19] to compute the metadata and our *Metadata Management System* to harvest interesting insights from the results. Both frameworks are available online[3].

The Metanome tool is a hub between datasets and profiling algorithms. Research in the area of data profiling has devised many different metadata discovery algorithms that automatically detect *all* statistics or dependencies of a certain type in a dataset. This is a notoriously difficult task, even for machines. Indeed, most commercial profiling tools lack real *discovery* features and allow only the examination of a few user-chosen dependency candidates. In contrast, Metanome bundles state-of-the-art metadata discovery algorithms with a unified interface and provides them to data scientists. In our data anamnesis process, we feed the MusicBrainz dataset into Metanome to extract several single-column statistics (e.g., number of distinct values) and several multi-column dependencies, namely *unique column combinations (UCCs)* [13] (combinations of columns that do not contain duplicate values), *inclusion dependencies (INDs)* [15] (dependencies stating that all values in one column combination are also contained in another column combination), and *functional dependencies (FDs)* [21] (dependencies stating that the values in a particular column combination determine the values in a particular another column). These four types of metadata can be used to determine data types (statistics), key constraints (UCCs), foreign-key constraints (INDs), and normal forms (FDs), which are the most important schema properties for relational databases and, hence, the most relevant metadata types for data anamnesis. Note that many other types

[1] `http://ftp.musicbrainz.org/pub/musicbrainz/data/fullexport/`

[2] `https://github.com/metabrainz/musicbrainz-server/tree/master/admin/sql`

[3] `http://www.metanome.de/` and `https://github.com/stratosphere/metadata-ms`

of metadata exist, such as order dependencies to reason about table sortation criteria and matching dependencies to spot data inconsistencies, but for this article they are of lower relevance.

Because data are not always correct, each type of metadata has several relaxation degrees, e.g., *approximate* metadata (valid with a certain confidence only), *conditional* metadata (only valid for records fulfilling a certain condition), or *partial* metadata (only valid for some percentage of records). However, data quality issues are not the focus of this article: We do not intend to clean, but to describe the data. Therefore, we focus on *exact* metadata. Nonetheless, one might also perform a data anamnesis with relaxed metadata. Another interesting extension of above mentioned metadata types addresses null values. In contrast to regular domain values, null values can be interpreted in various ways, such as *not applicable*, *no value*, or *unknown value*. Specialized metadata types accommodate such semantics. Amongst others, certain keys [14] and strong FDs [16], respectively, describe UCCs/FDs that are always obtained from a relation by replacing nulls with domain values, no matter which values are chosen. In this article, we take a rather pragmatic approach: In our applications of UCCs and FDs, the SQL standard prohibits null values, so we just treat null as another domain value; and for INDs, we ignore tuples with null values in accordance with the default semantics of SQL foreign keys.

Efficiently extracting statistics and dependencies from a dataset is a crucial, but only the first step towards a data anamnesis. So far, these metadata are a loose set of facts about the original dataset and still far from the desired reconstructed schema or an assessment of the same. For instance, a discovered dependency might reflect an actual database constraint or merely occur by chance. To obtain practical insights on the original dataset, it is necessary to interlink the different types of metadata into a cohesive "metadata knowledge base" and then thoroughly comb through it. This step is challenging, because metadata are highly heterogeneous and often huge. The Metadata Management System (MDMS for short) approaches this challenge by mapping the diverse types of metadata onto a unified vocabulary and saving them in a scalable storage. This storage layer is further complemented with an analytical layer on top that offers ad-hoc queries and visualizations on the metadata for interactive explorations. We store the metadata produced by Metanome in the MDMS and, from there, we manually explore the metadata but also devise specialized metadata refinement operators (e.g., to designate the primary keys among given UCCs), both for the purpose of schema reconstruction and diagnosing schema quality.

2.2 Raw data profiling results

The MusicBrainz dataset comprises 324 headerless CSV files, each of which constitutes a table, with a total volume of 35.1 GB. This size renders a manual inspection for statistics and dependencies virtually impossible, particularly facing the exponential complexity of dependency discovery. However, we were able to profile MusicBrainz with Metanome on a computer with 128 GB of RAM (statistics, FD, and UCC discovery) and on a commodity hardware cluster of 10 desktop machines, respectively (IND discovery). Table 1 summarizes for each analyzed metadata type, the number of elements that Metanome has discovered and the time it took to discover them.

Metadata type	Results	Discovery time
Column/table statistics	1,835	4.4 h
Minimal functional dependencies	4,193	0.5 h
Minimal unique column combinations	675	0.5 h
Maximal inclusion dependencies	93,502	2.0 h

Table 1: Basic profiling results for the MusicBrainz dataset.

At first, we remark that for 73 of the 324 tables ($\approx 23\,\%$) are completely empty, so that we could not produce any profiling results. In fact, the profiling lacks any evidence to state even how many columns these tables are supposed to have. Apart from that, we observe that the different types of metadata yield vastly different numbers of results, with INDs being the most frequent metadata type – on average 288 INDs for each table tables. INDs are different from the other dependencies, because they reach across tables, while the other dependencies regard tables in isolation of one another. Nonetheless, there are also on average more than two UCCs (28 at most) and almost 17 FDs (295 at most) per table. These numbers demonstrate the amount of metadata that a data anamnesis typically entails and a purely manual evaluation of these metadata would require a substantial amount of time. In addition, for all types of metadata we observe a trend: With increasing numbers of columns in a table, there are more dependencies to be found. In particular FDs exhibit a superlinear, somewhat exponential growth. It is therefore notable that MusicBrainz has at most 20 columns in a single table, while datasets with tables of several hundred columns are not unusual [20].

3 Schema Discovery

Schema discovery constitutes the data-driven reconstruction of a dataset's schema. It is at the core of any data anamnesis. Even if a dataset is accompanied by a schema, it is still advisable to verify the correctness of that schema. And indeed, in our experiments, we found that our version of the MusicBrainz dataset slightly differs from the schema obtained from GitHub (see Section 1). We advocate that schema discovery should not directly inspect the bare dataset but operate on its previously discovered metadata, which not only summarize the actual data but also expose their latent properties, such as INDs, FDs, that are essential to data anamnesis but need substantial effort to gather from the data. However, as described in the previous section, the output of metadata discovery algorithms can be fairly extensive itself. In consequence, these raw metadata are not yet useful to be directly presented to users and leave them alone with the task of schema discovery. To this end, we identify two different schema discovery phases, similar to [11], and show for both of them, which techniques support the user in carrying them out. The first phase, the *constraint reconstruction*, recovers technical schema details. The second one, *conceptualization*, aims at helping the user to build a mental model of the schema.

3.1 Constraint reconstruction

The goal of the constraint reconstruction is to detect meaningful database constraints for a given dataset, such as primary keys and value ranges. In contrast to any profiled metadata, which is *descriptive*, database constraints are *prescriptive*. Consequently, the constraint reconstruction is a heuristic process: We can exactly tell which constraints a given dataset satisfies, but we cannot guarantee that these constraints are meaningful. For instance, for an attribute track_number, we might propose the constraint track_number ≤ 33, assuming 33 is the maximum value for that attribute. Intuitively, however, this is not a reasonable constraint.

In our case study, we focus on reconstructing four core constraint types, namely data types, nullability, primary keys (PKs), and foreign keys (FKs), using automated algorithms that exclusively rely on previously extracted metadata. Further constraint types, such as inferring `CHECK` clauses from denial constraints [6], are out of the scope of this paper. To evaluate the quality of the applied reconstruction algorithms, we derived a constraint gold standard from the existing schema definition from GitHub. We discovered, however, that this schema definition contains 54 tables that are not in our dataset export and also lacks one exported table. This is an unintentional example for incorrect metadata and endorses the idea of data-driven anamnesis. Nevertheless, we use the constraints of the 323 tables that are both in the dataset export and the existing schema definition as our gold standard. In addition, we use a second gold standard that further neglects the 73 empty tables (see Section 2.2) for two reasons: First, it is plausible that in many scenarios, the user has no interest in reconstructing the schema for empty tables. Second, any data-driven approach to reconstruct the schema for empty tables is inherently condemned to fail due to the lack of data. Thus, the empty tables bloat the error of our reconstruction algorithms.

Constraint	Recon-structed	with empty tables				without empty tables			
		Gold	Accuracy	Precision	Recall	Gold	Accuracy	Precision	Recall
Data types	1,314	1,880	0.68	–	–	1,511	0.84	–	–
`NOT NULL`	1,256	1,454	–	0.89	0.77	1,112	–	0.89	1.00
PKs	250	315	–	0.77	0.62	242	–	0.77	0.80
FKs	310	554	–	0.77	0.44	421	–	0.77	0.57

Table 2: Qualitative results of the constraint reconstruction with the two different gold standards. *Reconstructed* and *Gold* show the number of reconstructed constraints and gold standard constraints, respectively.

At first, we reconstruct the data type and nullability of each column. We propose the most specific SQL data type (e.g., `SMALLINT` is more specific than `INT`, which is more specific than `FLOAT`) that accommodates all values of a column. If a column does not contain null values, we additionally propose accordingly a `NOT NULL` constraint. Then, we determine for each table a primary key (PK), which has to be a unique column combination (UCC) that consists exclusively of `NOT NULL` columns. If a table comprises more than one such UCC, we designate a PK heuristically: (i) PKs are typically among the first columns of a table, (ii) their columns are typically contiguous, and (iii) PKs usually involve mostly short values.

Finally, we propose foreign keys (FKs) from the INDs in three steps: We first consider only those INDs whose right-hand side is a UCC, because FKs can only reference either PKs or `UNIQUE` columns. Then we apply four FK heuristics (inspired by [22, 25]) to quantify the eligibility of each IND as an FK: (i) the referencing and referenced columns of an FK should have similar value distributions and (ii) a similar number of distinct values; further, (iii) the names of the referencing and referenced table are usually similar, where (iv) referenced table names are supposedly rather short. Finally, in a greedy approach, we iteratively promote the most eligible INDs to FKs, thereby ensuring that each attribute appears as a referencing attribute in at most one FK. It is notable that neither of the previous works were directly applicable in our case study. Not all of the features proposed in [22] were available (e.g., column names) and moreover this approach requires training data. Also, relying only on value distributions as in [25] was not sufficient, as we explain later in this section. Instead, we found it to be crucial to leverage the peculiarities of the given data. In our case, all features pertain almost equally to the ranking of the FK candidates and are configured rather conservatively, e.g., an FK should cover at least 90 % of the referenced columns' distinct values (Criterion (i)).

All above reconstruction algorithms finished within seconds, because we use light-weight, greedy heuristics, but also because these algorithms process only metadata, not the data. Table 2 displays the quality of our reconstructed constraints. Our simple heuristics did not deliver perfect results, yet, they were able to reconstruct the majority of constraints correctly. The errors in the data type reconstruction arise, because 6 % of all columns contain only null values, but also due to unexpected exotic data types, such as `BYTEA`, and incorrectly predicted `(VAR)CHAR` lengths. The detection of `NOT NULL` constraints discovered all true `NOT NULL` constraints, but also reveals that roughly 10 % of all null-free columns would actually permit null values.

In the reconstruction of the PKs, we identify two major problems: Firstly, eight tables have no specified PK, violating Codd's entity integrity. Nevertheless, we propose a PK for each of them. Secondly, in 43 tables the PKs are superkeys, i.e., a subset of their attributes already uniquely identifies each tuple. For these 43 tables, we propose exactly these subsets as PKs. In general, smaller keys are preferable, because they need smaller indices, can be enforced more efficiently, and allow for more efficient join processing. In these cases, a domain expert should determine whether our proposed PKs are indeed semantically correct.

The reconstruction of FKs achieved the lowest recall, but also was the most tricky: Of the 93,502 INDs, 43,290 reference an alleged PK and are thus FK candidates; so on average, there is only one FK among 102 FK candidates. It naturally ensues that among the many non-FKs there are some that look like FKs. The reason is surrogate keys: In MusicBrainz, tuples in a table are identified by an auto-incremented integer. Consequently, many tables have similar sets of PK values and, hence, FK values; and these form the pseudo-FKs among each other. Surrogate keys are popular for efficiency reasons and due to their robustness w.r.t. schema changes, so that the above described problem is likely faced in other datasets, too.

In general, our results indicate that automatic constraint reconstruction methods are a viable aid and our simple methods can most likely be further improved. Nevertheless, they should be employed in a semi-automatic reconstruction process, in which they support a human who decides upon the eventual set of constraints.

3.2 Conceptualization

While the constraint reconstruction reveals technical properties of a dataset, it does not yet tell users how to work with that dataset, e.g., *what* information they can query and *how*. Therefore, schema discovery should contain a complementary conceptualization phase, in which users familiarize themselves with the structure of the dataset. Indeed, various conceptualization approaches exist. One line of research conceived frameworks to derive logical and/or conceptual schemata from a database schema, usually as (extended) entity-relationship diagrams [5, 11]. Such higher-order schemata support human understanding by scrapping technical concerns from the database schema and providing comprehensible abstractions. However, for the same reasons, they are rather suited to understand the domain of a dataset while concealing potentially important details of the database schema. A second line of research aims at enriching database schemata using data mining methods [10, 23, 24]. These works particularly create informative summaries *directly on* the database schemata, e.g., clustering tables or detecting composite fields. Such summarization techniques are a very useful schema exploration tool. To exploit these techniques in an interactive fashion in practical scenarios, we propose to embed them in a metadata query language, complemented with rich visualizations. Note that this approach fits well with notebook applications, such as IPython[4], which are recently gaining in popularity in the data science community. In the following, we exemplify this idea with a conceptualization workflow for the MusicBrainz dataset.

Assuming that not all of the 324 tables in the dataset are equally important, we want to find the "points of interest", that is, the most relevant tables, to get an intuition of the dataset's contents. The simple heuristic that voluminous tables are important is a sufficient starting point in our case study, but note that more sophisticated relevance criteria exist [23]. We query against the MDMS: *Retrieve all tables, sorted descendingly by their number of columns multiplied by their number of tuples.* Among the top 20 results are the tables track, release, and artist. Intuitively, the concepts of these tables are easily grasped, so they form a good starting point to fan out the further exploration with follow-up queries, e.g., by looking for similar tables [10, 23].

As we proceed, we are also interested in grasping the extent of the MusicBrainz dataset. We query: *Group the tables by their number of rows and columns and count the number of tables in each group. Visualize the result in a bubble chart.* Figure 2 displays the result and reveals that the majority of tables have few columns and few or even no values. For the sake of visual clarity, we alter the query to group the tables only by their number of columns and choose a bar chart visualization. This yields Figure 3. It is in a way intuitive that the majority of tables have only very few columns. However, the number of tables with exactly 5, 9 and 16 columns are surprisingly high given their neighbor groups. Accordingly, we pose queries to drill down into these groups. The result reveals regularities: Tables with 5 columns are named ****_type*, tables with 9 columns are named *l_***_****, and tables with 16 columns are called ****_alias*. Apparently, these 104 tables implement only three different table templates, i.e., type-, link-, and alias-tables. This insight can accelerate the familiarization process, e.g., when names for the columns of these tables should be devised.

This simple example demonstrates the utility of integrated metadata in conjunction with data mining methods and interactive tool support when conceptualizing database schemata. Obtaining the above insights by manually browsing through the dataset would have been very tedious and less exact. Nevertheless, we believe that this approach still bears much potential to be unlocked by further research. At first, more data mining methods

[4]`https://ipython.org/`

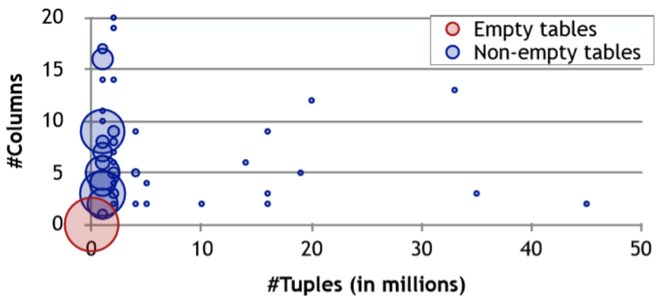


Figure 2: Number of tables (= circle area) grouped by their number of columns and rows.

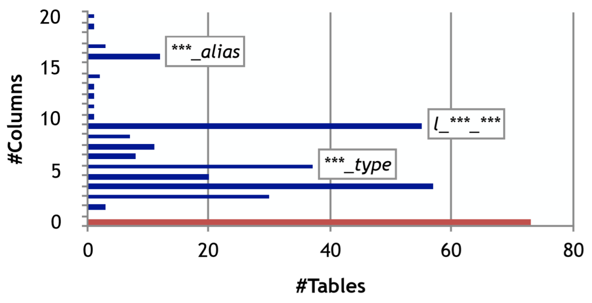


Figure 3: Number of tables grouped by their number of columns.

should be applied to metadata to answer novel questions, e.g., detecting schema design patterns using frequent itemset mining. Data mining and summarization could also be a viable tool to handle unwieldy data profiling results (cf. Table 1). Second, a query language should be devised to integrate the various data mining methods. A major challenge of such a query language is that it has to accommodate multiple different facets. For instance, database schemata are typically regarded as a set of tables, but they can also be interpreted as graphs, where the tables are the vertices and foreign key relationships form the edges. Finally, the visualization of the query results should be carefully addressed. Devising elaborate and interactive visualizations can drastically improve the understandability of those results. Also, picking and configuring the right visualizations automatically can enhance the schema exploration considerably.

4 Schema Quality Diagnosis

Regardless of whether a schema was available with the dataset or whether it (or parts of it) were discovered, it is a worthwhile endeavor to assess its quality. Not only do schemata model data and are in consequence an important factor to assure data quality aspects, such as completeness and consistency. Schemata also mediate between applications and data, thereby influencing the application, e.g., in terms of complexity. The description and assessment of schema quality is a well-known topic in the literature, to which we add a new perspective. Previous work has proposed a *top-down* assessment [4]: Given a specification, a list of requirements, an application, or an ER-model, determine the quality of a database schema along various dimensions, so as to predict whether a database with this schema can manage all incurring data of the intended use case. This proceeding is suitable when engineering new schemata and applications. However, in the context of data anamnesis, we face a reverse setting: Instead of applications or requirements, we are given only the data. In this scenario, the traditional, anticipatory schema quality assessment cannot be applied and a retrospective assessment is needed instead. To this end, we propose a *bottom-up* view of the schema quality problem: Given a database instance and a corresponding schema (specified or discovered), determine the quality of that database schema along various dimensions. Some of these dimensions are the same as for the top-down analysis, others are new. The two approaches can be interpreted as psychic (soul) and somatic (body) anamnesis and complement each other.

Next, we briefly outline existing work on top-down schema quality. Then, and in more detail, we discuss how to assess the quality of a schema using only metadata derived from an instance.

4.1 Top-down schema quality dimensions

We briefly summarize related work on assessing the quality of a schema. According to our classification, the following approaches take a top-down view, i.e, they are independent of any instance data.

Arguably, Heath, Codd, Bernstein and others were the first to address schema quality by introducing various normal forms, i.e., constraints to reduce redundancy while maintaining same semantics [12, 7, 3]. Burkett provides a comprehensive survey of frameworks to explicitly assess schema quality [4]. Among them, a typical work is by Batini et al. about the quality of a conceptual (ER-based) schema [2]. The authors define a set of schema quality dimensions, namely *completeness* (schema represents all relevant features of the application domain), *syntactic and semantic correctness* (proper use of concepts of ER-model), *minimality* (every aspect of requirements appears only once), *expressiveness* (schema represents requirements in a "natural way and can be easily understood), *readability* (graceful diagram), *self-explanation* (prefer model constructs over natural language explanations), *extensibility* (easily adaptable to changing requirements), and *normality* (adherence to normal forms). Clearly, most dimensions relate the schema to a concrete application and can only be assessed with respect to the application's requirements. The normality dimension is an exception, and we address it in our data-driven schema quality assessment.

Another typical opportunity to assess schema quality is for integrated schemata, because here the reference point is not a possibly vague set of requirements but the source schemata that contributed to the integration effort. For instance, Batista and Salgado examine minimality, consistency, and completeness of integrated schemata, and offer transformations to improve each of them [9].

4.2 Bottom-up schema quality assessment

While traditional top-down schema quality assessment judges schemata from the viewpoint of users, applications, and requirements, our proposed bottom-up assessment judges schemata from the viewpoint of the data. Instead of measuring the fit of a conceptual domain and a technical schema, this approach measures the fit of the schema and its data. In the following, we want to show that it is generally possible to assess schema quality based on the data and its metadata. For this purpose, we devise assessment methods for two schema quality dimensions, namely *conciseness* and *normality*, and exemplify them using the MusicBrainz dataset. The purpose of these methods is not to certify low or high schema quality, but to point out tangible schema quality issues.

4.2.1 Conciseness

We refer to conciseness as the schema counterpart of data completeness: If a schema is concise, it allows to exactly express the data. In contrast, a schema that is too general is also more complex, making it harder to understand that schema and increasing the complexity of queries and applications working on top of it.

An obvious indicator for a lack of conciseness are unused schema elements, i.e., empty tables and columns with all or mostly null values. As a counter-measure, such schema elements might be pruned. In the MusicBrainz dataset, 22.5 % of the tables do not contain any tuples. We regard the remaining tables in more detail and analyze the *fill level* of their columns, i.e., the percentage of non-null values. This yields a fill profile for each table, as exemplified in Figure 4, which reveals the inner structure of selected tables: Each series corresponds to a table and each data point in the series represents a column and its fill level. For visual clarity, the columns of each table are ordered by this percentage. We find that 5 % of all columns are completely empty and another 6 % have a fill level of less than 20 %. Apparently, the MusicBrainz schema models some properties that apply to only few or even no actual data instances. Those findings do not necessarily entail low quality, however, they show that the MusicBrainz schema is not as concise as it could be.

Another indicator for conciseness are relationships among tables. In the relational model, 1-to-1 relationships can be satisfied using a single table, 1-to-n relationships are realized by foreign keys, and m-to-n relationship need a link table. More general relationships require more complex modeling, so a concise schema should model the most specific possible type for each relationship. To check this, we proceed as follows: At first, we search for FKs "A references B", where B is a key candidate and $B \subseteq A$ is a valid IND. This reveals 1-to-1 relationships that are modeled as 1-to-n. Second, we look for link tables $R(A, B)$, where A and/or B are

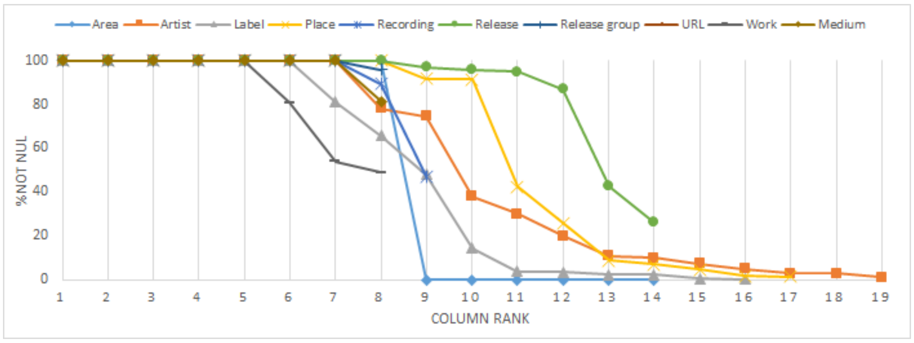


Figure 4: Visualization of the fill level profile for 10 core tables in MusicBrainz.

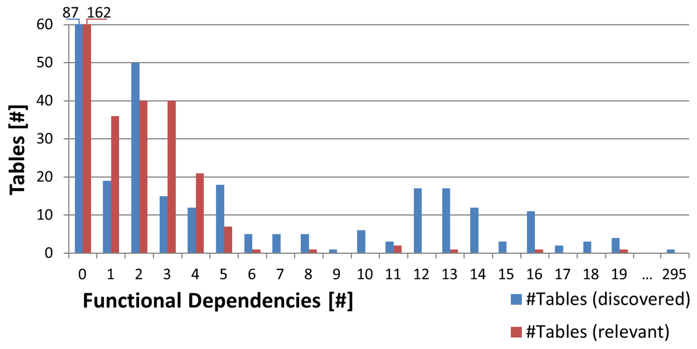
key candidates, thereby revealing modeled m-to-n relationships that actually accommodate only 1-to-n or even 1-to-1 relationships. Indeed, we find examples for both in MusicBrainz: The tables release and release_meta are linked via an FK but are de facto in a 1-to-1 relationship and the tables artist and annotation are interlinked by the table artist_annotation into an m-to-n relationship, although each annotation belongs to exactly one artist. It might well be that a deliberate design decision stands behind this mismatch, but it might also indicate a lack of schema conciseness.

The two above diagnoses are by far not exhaustive. For instance, one might check if the schema always uses the most specific datatype for its data. The choice of the datatype can influence the disk requirements and computation speed of databases (e.g., `SMALLINT` and `INT`). Furthermore, INDs could be used to detect duplicate data, allowing to further reduce the schema.

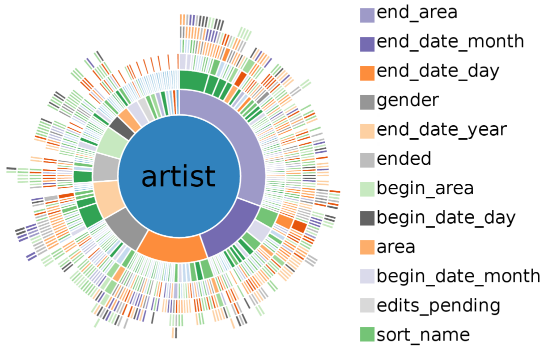
4.2.2 Normality

Ideally, a schema should not impose redundancy to its data, i.e., the schema should ensure that any fact in a database is stored only once. Redundancy typically entails various problems, such as increased data volume and the potential for inconsistencies. In the context of relational data, a major cause of redundancy are denormalized schemata. To diagnose denormalization, FDs are a most valuable asset. Basically, an FD reveals implicit 1-to-n relationships between columns in a single table. As discussed in Section 2.2, we discovered 4,193 FDs in the 250 non-empty tables of the MusicBrainz dataset. Compared to typical evaluation datasets for FD algorithms, where many tables with 20 attributes alone contain more than 8,000 FDs [20], this is a low number. Moreover, combining the FDs with the UCCs shows that 61 % of the FDs have a left-hand side that is a key or superkey. Such FDs are not relevant for the typical normalization goal of Boyce-Codd normal form [7]. However, there are still approximately ten relevant FDs per table on average, which is not negligible. Thus, we investigate the distribution of the discovered and relevant FDs over the MusicBrainz tables in Figure 5a.

Apparently, 162 tables (including the 73 empty tables) contain no relevant FDs at all. For the other 162 tables, it must be determined manually, which FDs reflect actual denormalizations and which hold only by incident. Here, we can exploit our observations that many tables are created from templates (see Section 3.2) and reduce the amount of manual work by grouping FDs that appear multiple times among tables with the same template. Then, there are a few tables that accommodate relatively many FDs. With a number of 245, the artist table is the one with the most relevant FDs. Here, visualization techniques can help to understand and classify those FDs, as shown for the artist table in a sunburst diagram in Figure 5b: The circle in the center of this diagram represents the table itself, the inner-most ring represents the dependent attributes, and all further rings represent the determining attributes, so that each sector forms an FD. With this visualization, we directly see that most FDs determine end_area, followed by end_date_month and end_date_day. This is, because these attributes are all very sparse, i.e., they contain 98%, 97%, and 96% null values. If they would be entirely empty



(a) Number of tables with certain numbers of FDs.



(b) FDs in artist table as sunburst diagram: First ring represents RHS attributes and outer circles LHS attributes.

Figure 5: Visualizations of the "denormalizing" FDs in MusicBrainz.

or more populated, fewer FDs would exist, because each attribute alone determines an empty attribute and it is much more difficult to determine an attribute with many distinct values. For this reason, most of these FDs are accidental and simply a consequence of sparsely populated attributes.

In our manual review, we could identify no indisputable violation of schema normality. Note, however, that types of redundancies other than denormalization exist. It is therefore interesting to inspect how other integrity constraints, particulary INDs and multi-valued dependencies, can be employed for that purpose as future work.

5 Conclusion & Outlook

Data quality is an important topic whenever data integration, data analytics, or big data in general are mentioned. However, before data quality can be measured, the data's schema must be captured and understood. Data anamnesis is the process of automatically determining the schema from a given dataset and then supporting the expert in understanding the schema and its quality, thereby exclusively resorting to the data themselves to avoid any error or bias from external sources. We substantiated and demonstrated this notion of data anamnesis with the help of the MusicBrainz dataset. At first, we used data profiling techniques to extract relevant metadata and then reconstructed a database schema from the metadata using classification algorithms and data exploration techniques. Finally, we used the metadata to assess the quality of that schema. For this purpose, we proposed a novel bottom-up approach that relates schema quality to a schema's actual data instance (and its metadata, respectively) and not to its fitness for an application. Thus, our approach complements existing work while putting to use the recent advances in data profiling. We illustrated this approach with a new and an existing schema quality dimension, thereby learning that the MusicBrainz schema is highly normalized but overly extensive.

Throughout the paper, we pointed out that much work remains to be done. In particular, we demonstrated the great potentials of data profiling results for database-reverse engineering and quality assessment. However, research is still in its infancy regarding the interpretation of these results: How can we make the leap from measured metadata to semantics? Most discovered functional dependencies, inclusion dependencies, data types, and other constraints are likely to be spurious. How can we separate the wheat from the chaff? Any existing data can be viewed as a (possibly skewed) sample of the data yet to come. Understanding data and its schema is a topic not likely to vanish from the research agenda.

Acknowledgements This research was partially funded by the German Research Society (DFG grant no. FOR 1306) and the German Academic Exchange Service (DAAD program no. 57169181).

References

[1] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. Profiling relational data: A survey. *VLDB Journal*, 24(4):557–581, 2015.

[2] Carlo Batini, Stefano Ceri, and Shamkant B. Navathe. *Conceptual Database Design*, chapter 6. Benjamin/Cummings Publishing Company, Redwood City, 1992.

[3] Philip A. Bernstein. Synthesizing third normal form relations from functional dependencies. *ACM Transactions on Database Systems (TODS)*, 1(4):277–298, 1976.

[4] William C. Burkett. Database schema design quality principles, 1997.

[5] Roger HL. Chiang, Terence M. Barron, and Veda C. Storey. Reverse engineering of relational databases: Extraction of an EER model from a relational database. *Data and Knowledge Engineering (DKE)*, 12(2):107–142, 1994.

[6] Xu Chu, Ihab Ilyas, and Paolo Papotti. Discovering denial constraints. *Proceedings of the VLDB Endowment*, 6(13):1498–1509, 2013.

[7] Edgar F. Codd. Recent investigations into relational data base systems. Technical Report RJ1385, IBM, 1974.

[8] Fabien Coelho, Alexandre Aillos, Samuel Pilot, and Shamil Valeev. On the quality of relational database schemas in open-source software. *International Journal on Advances in Software*, 4(3 and 4):378–388, 2011.

[9] Maria da Conceição Moraes Batista and Ana Carolina Salgado. Information quality measurement in data integration schemas. In *Proceedings of the International Workshop on Quality in Databases (QDB)*, pages 61–72, 2007.

[10] Tamraparni Dasu, Theodore Johnson, Shanmugauelayut Muthukrishnan, and Vladislav Shkapenyuk. Mining database structure; or, how to build a data quality browser. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 240–251, 2002.

[11] Jean-Luc Hainaut, Jean Henrard, Vincent Englebert, Didier Roland, and Jean-Marc Hick. Database reverse engineering. In *Encyclopedia of Database Systems*, pages 723–728. 2009.

[12] I. J. Heath. Unacceptable file operations in a relational data base. In *Proceedings of the ACM SIGFIDET Workshop on Data Description, Access, and Control*, pages 19–33, 1971.

[13] Arvid Heise, Jorge-Arnulfo Quiané-Ruiz, Ziawasch Abedjan, Anja Jentzsch, and Felix Naumann. Scalable discovery of unique column combinations. *Proceedings of the VLDB Endowment*, 7(4):301–312, 2013.

[14] Henning Köhler, Sebastian Link, and Xiaofang Zhou. Possible and certain SQL keys. *Proceedings of the VLDB Endowment*, 8(11):1118–1129, 2015.

[15] Sebastian Kruse, Thorsten Papenbrock, and Felix Naumann. Scaling out the discovery of inclusion dependencies. In *Proceedings of the Conference Datenbanksysteme in Business, Technologie und Web (BTW)*, pages 445–454, 2015.

[16] Mark Levene and George Loizou. Axiomatisation of functional dependencies in incomplete relations. *Theoretical Computer Science*, 206(12):283 – 300, 1998.

[17] Heather Levy. Why and how to value your information as an asset. `http://www.gartner.com/smarterwithgartner/why-and-how-to-value-your-information-as-an-asset/`, 2015. Accessed: 11-25-2015.

[18] MusicBrainz. `https://musicbrainz.org/`. Accessed: 11-25-2015.

[19] Thorsten Papenbrock, Tanja Bergmann, Moritz Finke, Jakob Zwiener, and Felix Naumann. Data profiling with Metanome. *Proceedings of the VLDB Endowment*, 8(12):1860–1863, 2015.

[20] Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. Functional dependency discovery: An experimental evaluation of seven algorithms. *Proceedings of the VLDB Endowment*, 8(10):1082–1093, 2015.

[21] Thorsten Papenbrock and Felix Naumann. A hybrid approach to functional dependency discovery. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, 2016.

[22] Alexandra Rostin, Oliver Albrecht, Jana Bauckmann, Felix Naumann, and Ulf Leser. A machine learning approach to foreign key discovery. In *Proceedings of the ACM SIGMOD Workshop on the Web and Databases (WebDB)*, 2009.

[23] Xiaoyan Yang, Cecilia M. Procopiuc, and Divesh Srivastava. Summarizing relational databases. *Proceedings of the VLDB Endowment*, 2(1):634–645, 2009.

[24] Xiaoyan Yang, Cecilia M. Procopiuc, and Divesh Srivastava. Summary graphs for relational database schemas. *Proceedings of the VLDB Endowment*, 4(11):899–910, 2011.

[25] Meihui Zhang, Marios Hadjieleftheriou, Beng Chin Ooi, Cecilia M. Procopiuc, and Divesh Srivastava. On multi-column foreign key discovery. *Proceedings of the VLDB Endowment*, 3(1):805–814, 2010.