Visualizing Redundancies caused by Multivalued Dependencies in Relational Data

Christoph Köhnen[1]

[1]*Chair of Scalable Database Systems, University of Passau, Germany*

Abstract

Based on a novel visualization of redundancies caused by functional dependencies in relational data, this work motivates an extension to multivalued dependencies. The earlier work is inspired by the visualization of dental plaque and colors cells in relational data with hues depending on their grade of redundancy, which corresponds to a lower information content. Lost values in a relation instance can be restored using given functional dependencies and other tuples. This effect also holds for multivalued dependencies. However, restoring data using this dependency tends to require more data from the rest of the relation. Thus, the redundancies caused by multivalued dependencies are much lower than the ones caused by functional dependencies. The optimizations to reduce the computation effort from the original work generally do not hold for multivalued dependencies, so a future challenge is to adapt these conditions to these dependencies. Our vision is to offer the plaque test for more general kinds of dependencies as an efficient tool to visualize redundant data.

Keywords

Functional Dependencies, Multivalued Dependencies, Relational Model, Information Theory, Entropy

1. Introduction

Visualization in relational data is an important topic. There exist several models to visualize functional dependencies, a cornerstone of database normalization, e.g., in sunburst diagrams or graph-based visualizations [1]. These approaches can be helpful for data exploration. However, these models visualize the functional dependencies stand-alone, without taking specific relation instances into account. Hence, redundancies in databases can hardly be identified in such a visualization.

Database schemas in sufficiently high normal form avoid redundancies and prevent update anomalies. Arenas and Libkin [2] define a schema as *well-designed* if and only if there is no possible instance containing redundancies, which, in the case of functional dependencies, equivalently means that the schema is in Boyce-Codd normal form. Such redundancies caused by functional dependencies are visualized by a novel principle, which is presented in [3] and called "plaque test". In this approach, each cell in a relation instance is colored according to its value of information content (or entropy), which depends on a chosen set of fulfilled functional dependencies. Such a visualization can be helpful to assist database normalization and even for teaching normal forms. The theory of computing this information content originates from [2] and applies not only to functional dependencies but even to generalized dependencies. In this work, we take the obvious next step by extending the "plaque test" to multivalued dependencies and discuss their contribution to the entropy values in relation instances.

ID	AlbumTitle	Band	BY	RY	TR	TrackTitle
1	Not That…	Anastacia	1999	2000	1	Not Th…
1	Not That…	Anastacia	1999	2000	2	I'm Out…
1	Not That…	Anastacia	1999	2000	3	Cowboy…
2	Wish You…	Pink Floyd	1965	1975	1	Shine O…
3	Freak of…	Anastacia	1999	2001	1	Paid my…

1
0.9
0.8
0.7
0.6

Figure 1: Plaque test for the given relation with the functional dependencies from (⋆). Cell hues correspond to entropy values, defined in the color scale. Figure adapted from [3].

The concept of "plaque test" for functional dependencies [3] is illustrated in Figure 1. The relation[1] shown manages a CD collection. Each CD has an identifier (ID), a title (AlbumTitle), and was released in a given year (RYear/RY) by a band (Band), which was founded in a given year (BYear/BY). Each CD has tracks (Track/TR) and each track has a title (TrackTitle).

We consider the following functional dependencies:

$$\begin{gathered} \text{ID} \rightarrow \text{AlbumTitle, Band, BYear, RYear} \\ \text{ID, Track} \rightarrow \text{TrackTitle} \quad \text{and} \quad \text{Band} \rightarrow \text{BYear} \end{gathered} \tag{$\star$}$$

The result of the "plaque test" for the functional dependencies in (⋆) is shown in the original instance in Figure 1. The color hues are chosen corresponding to the cell's entropy, as defined in the legend on the right: the more redundant the value, the smaller the entropy and the deeper the blue. For white cells with entropy 1 we say that this cell has full information content.

35th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), May 22-24, 2024, Herdecke, Germany.

christoph.koehnen@uni-passau.de (C. Köhnen)

[1]Example from the German Wikipedia site on database normalization, at https://de.wikipedia.org/wiki/Normalisierung_(Datenbank), last accessed April 2024.

If we consider the attribute “BYear”, lost values can be recovered by checking the values in other tuples together with two functional dependencies, while the values in “AlbumTitle”, “Band” and “RYear” can only be recovered using one functional dependency. This is a reason for the lower entropy values in “BYear”. Additionally, this column has differently high redundancies. This is due to the fact that the founding year in the row with ID 1 can be restored either with the functional dependency “ID $\rightarrow$ BYear” or with “Band $\rightarrow$ BYear”, while for the row with ID 3 this can only be done with the latter one.

As stated above, the entropies depend on the choice of functional dependencies. Adding the dependency “BYear $\rightarrow$ Band”, which is fulfilled in this instance, causes lower entropies in “Band”. In general, adding dependencies to the chosen set might reduce the entropies but can never increase them. Choosing the set of all fulfilled functional dependencies would cause the lowest possible entropies. This case is considered as an example in [3].

A logical next step is the extension of the “plaque test” to other dependencies, since the theoretical framework from [2] applies to general constraints. We will consider a relation with multivalued dependencies and compute the entropies with these dependencies as input. If the constraint set consists only of functional dependencies, there are two rules presented in [3]: immediately identify cells with an entropy of 1 and delete irrelevant rows and columns to execute the computation on a smaller table. However, these rules cannot be applied in the same way if the set of constraints contains multivalued dependencies.

Contributions. We build on a visualization approach for redundancies in relational databases caused by functional dependencies, which relies on an existing theoretical framework. We discuss the generalization of this approach to multivalued dependencies, which is in the scope of the theoretical framework, but has never been implemented or algorithmically optimized for implementation. We identify challenges in transferring simplifications for the redundancy computation, which are valid for functional dependencies, to multivalued dependencies.

Structure. In Section 2 we define the preliminaries, such as functional and multivalued dependencies and information content. In Section 3 we consider related work on functional dependencies and information theory. In Section 4 we present results for schemas with functional dependencies and a possible extension to multivalued dependencies. In Section 5 we discuss the next steps.

2. Preliminaries

We first introduce relations and functional dependencies as in [3], that is, we take into account the order of the tuples. We also use this principle for the definition of multivalued dependencies. Due to the definition of the entropy-related information content in [2] we need to identify each tuple unambiguously. We introduce the computation of the information content and present algorithmic optimizations and an approximation approach.

2.1. Relations, Positions and Variables

In the following, the set of positive integers is denoted by $\mathbb{N} := \{1, 2, \dots\}$. We define a partial map, representing an order-preserving relation, allowing duplicate tuples.

Definition 2.1. A *relation* R of arity m is specified by a finite set $\mathrm{sort}(R) := \{A_1, \dots, A_m\}$ of attributes A_i. The domain of an attribute A is denoted by $\mathrm{Dom}(A)$. Then an *instance* I of the relation R is defined as a partial map $I \colon \mathbb{N} \rightharpoonup \mathrm{Dom}(A_1) \times \dots \times \mathrm{Dom}(A_m)$ with finite domain of definition $\mathrm{Def}(I) := \{j \in \mathbb{N} \mid I(j) \text{ is defined}\}$.

For simplification, we assume $\mathrm{Dom}(A) = \mathbb{N}$ for all $A \in \mathrm{sort}(R)$, i.e., each tuple consists of positive integers.

Definition 2.2. Let R be a relation and I an instance of R. For a subset $\alpha \subseteq \mathrm{sort}(R)$ of attributes, we have the projection $\pi_\alpha(I)$ of the tuples in I to the attributes in α.

Given an instance I of a relation R we denote by $t_j[A_k]$ the value of the attribute A_k in the j-th tuple $t_j := I(j)$ of I. Similarly, for a subset $\alpha = \{A_1, \dots, A_s\} \subseteq \mathrm{sort}(R)$ of attributes, we denote by $t_j[A_1 \dots A_s]$ or $t_j[\alpha]$ the tuple of values $t_j[A_1], \dots, t_j[A_s]$, that is, the j-th tuple $\pi_\alpha(I(j))$ of the projection $\pi_\alpha(I)$. For subsets $\alpha, \beta \subseteq \mathrm{sort}(R)$ we define $t_j[\alpha\beta] := t_j[\alpha \cup \beta]$.

Definition 2.3. Let I be an instance of a relation R with attributes $\mathrm{sort}(R) = \{A_1, \dots, A_m\}$. A *position* in I is a pair (j, A_k) with $j \in \mathrm{Def}(I)$ and $k \in \{1, \dots, m\}$; it represents the cell of the k-th attribute of the j-th tuple, with value $t_j[A_k]$. The instance obtained by putting the value v at position $p = (j, A_k)$ is denoted by $I_{p \leftarrow v}$.

Let $Q = \{q_1, \dots, q_k\}$ be positions, $X = (x_1, \dots, x_k)$ distinct variables, and $V = (v_1, \dots, v_k)$ values in $\mathbb{N}$. The instance obtained by replacing each position q_i by the variable x_i resp. by value v_i for $1 \leqslant i \leqslant k$ is denoted by $I_{Q \leftarrow X}$ resp. $I_{Q \leftarrow V}$. Hence, the instance obtained by replacing the positions from Q by variables, and then position p by the value v, is $(I_{Q \leftarrow X})_{p \leftarrow v}$.

We choose to define Q as a set, but X and V as tuples. The reason is that the information content defined in [2], which we will introduce in Section 2.3, considers sets of lost values without duplicates and a fixed order, while variables and values are determined for a fixed position and, in the case of values, duplicates are possible.

2.2. Dependencies

In this section, we define syntax and semantics of functional and multivalued dependencies. We start by defining functional dependencies as in [4].

Definition 2.4. A *functional dependency* in a relation R is an expression of the form $f := \alpha \to \beta$, where $\alpha, \beta \subseteq \mathrm{sort}(R)$ are sets of attributes. It is called *simple*, if β consists of exactly one attribute.

An instance I (without variables) of a relation R is said to *fulfill* the functional dependency f (write $I \models f$) if for all $j_1, j_2 \in \mathrm{Def}(I)$ it holds

$$t_{j_1}[\alpha] = t_{j_2}[\alpha] \;\Rightarrow\; t_{j_1}[\beta] = t_{j_2}[\beta].$$

Next, we define multivalued dependencies as in [5].

Definition 2.5. A *multivalued dependency* in a relation R is an expression of the form $f := \alpha \twoheadrightarrow \beta$, where $\alpha, \beta \subseteq \mathrm{sort}(R)$ are sets of attributes.

An instance I (without variables) of a relation R is said to *fulfill* the multivalued dependency f (write $I \models f$) if for all $j_1, j_2 \in \mathrm{Def}(I)$ with $t_{j_1}[\alpha] = t_{j_2}[\alpha]$ there exists $j \in \mathrm{Def}(I)$ such that

$$t_{j_1}[\alpha\beta] = t_j[\alpha\beta] \quad \text{and} \quad t_{j_2}[\alpha\gamma] = t_j[\alpha\gamma],$$

where $\gamma := \mathrm{sort}(R)\backslash(\alpha \cup \beta)$ is the set of all attributes not covered by α and β.

From here we consider F as a set of functional and multivalued dependencies.

Definition 2.6. Let F and I be as above. The instance I (without variables) *fulfills* F (write $I \models F$) if $I \models f$ for all $f \in F$.

An instance I containing distinct variables at positions $Q = \{q_1, \ldots, q_k\}$ *fulfills* f resp. F if there exist values $V = (v_1, \ldots, v_k)$ such that it holds $I_{Q \leftarrow V} \models f$ resp. $I_{Q \leftarrow V} \models F$.

To check $I \models F$ in the implementation, we apply the equivalence with $I \models f \; \forall f \in F$, which indeed holds for instances I without variables. If I contains variables this equivalence can be secured under additional conditions. It clearly holds if F is closed under logical implications. If F only contains functional dependencies, it suffices to require the transitive closeness of F. In the sense of the theoretical framework [2] the computation leads to the same results regardless if F is a canonical cover, closed under logical implications or in between. However, applying the above equivalence can lead to incorrect results for non-closed F. Therefore, we offer an option to compute the closure prior to entropy calculations, which can be disabled if the closeness of the given set F is clear to reduce the computation time.

2.3. Entropy and Information Content

We now introduce the information content using the (information-theoretic) entropy. Given two probability spaces, the conditional entropy is introduced in [2] and [3]. In an instance I with a set $\mathrm{Pos} := \mathrm{Def}(I) \times \{A_1, \ldots, A_m\}$ of positions and a set F of functional and multivalued dependencies, for a considered position $p \in \mathrm{Pos}$, we define two probability spaces as follows.

First, let $\mathcal{B}(I,p) := (\mathcal{P}(\mathrm{Pos}\backslash\{p\}), P_{\mathcal{B}})$, where $P_{\mathcal{B}}$ is the uniform distribution on the set of all subsets of $\mathrm{Pos}\backslash\{p\}$. This space models the possible cases of a lost set of possible values from the instance I on positions other than the considered one p.

Then, for $k \in \mathbb{N}$ we let $\mathcal{A}_F^k(I,p) := (\{1, \ldots, k\}, P_{\mathcal{A}})$, where the conditional probability of $v \in \{1, \ldots, k\}$ given $Q \subseteq \mathrm{Pos}\backslash\{p\}$ is equally distributed over $V_Q := \{v \in \{1, \ldots, k\} \mid (I_{Q \leftarrow X})_{p \leftarrow v} \models F\}$ and zero otherwise. This space models the possible values in $\{1, \ldots, k\}$ at p, given that Q is the set of lost value positions in I.

With these two probability spaces, we can use the conditional entropy of $\mathcal{B}(I,p)$ given $\mathcal{A}_F^k(I,p)$, which is denoted as $H(\mathcal{A}_F^k(I,p) \mid \mathcal{B}(I,p))$. This conditional entropy measures some kind of uncertainty at position p, if the value is lost and putting in natural numbers up to k is possible. From another perspective, one can say that the conditional entropy is a quantification of the information content at a position, if the value is present. Intuitively, uncertainty, if a value is lost, corresponds to information content, if the value is present.

We now present the information content of position $p \in \mathrm{Pos}$ in instance I of relation R with respect to a set F of functional and multivalued dependencies with $I \models F$. While in [3] the special case, where F consists only of functional dependencies, is used, Arenas and Libkin [2] define the information content for general dependencies. Thus, the definition also holds for sets F containing functional and multivalued dependencies.

Definition 2.7. Let F, I and p be as introduced above. The *information content* of position p with respect to F in instance I is given as

$$\mathrm{INF}_I(p \mid F) := \lim_{k \to \infty} \frac{\mathrm{INF}_I^k(p \mid F)}{\log k},$$

where $\mathrm{INF}_I^k(p \mid F) := H(\mathcal{A}_F^k(I,p) \mid \mathcal{B}(I,p))$ is the conditional entropy of $\mathcal{A}_F^k(I,p)$ given $\mathcal{B}(I,p)$.

The information content for possible values limited to k is normalized by $\log k$ and the limit of k to infinity is taken to cover the entire set of natural numbers.

Since the space $\mathcal{B}(I,p)$ consists of all subsets of positions other than p, its size grows exponentially with the size of the relation. In the remainder of the section, we introduce optimizations and an approximate approach from [3] to overcome this performance problem.

2.3.1. Optimizations

We illustrate the optimizations from [3] with an example. Consider instance I given in Figure 2 and the (singleton) set $F := \{\mathrm{A} \to \mathrm{C}\}$ of functional dependencies. The two ways to optimize the computation are to directly identify positions with entropy 1 and to reduce the size of the problem. Note that it is still an open question whether these optimizations hold for a set containing multivalued dependencies.

A	B	C	D
7	2	8	4
5	2	8	6
7	2	8	6

Figure 2: Instance I.

*Identify ones.*A position $p := (j, A_k)$ has an entropy of 1 if and only if for all functional dependencies $f = \alpha \to \beta \in F$ with $A_k \in \beta$ we have that $t_j[\alpha]$ is unique in the (duplicate allowing) projection $\pi_\alpha(I)$. This means that a lost value at position p cannot be restored by other tuples and thus has an information content of 1. Hence, if a column does not appear on the right-hand side of any functional dependency from F, the positions in the whole column have full information content. This optimization allows to skip the computation for all the positions with the introduced property. In the example considered, all positions (j, A_k) with $A_k \neq C$ and position $(2, C)$ have full information content and we only have two positions left to compute.

Reduce problem size. Rows and columns can be deleted under conditions and the computation can be performed on the resulting subtable. Afterwards, the result is embedded in the original table. A column can be deleted if the attribute does not appear in any functional dependency from the given set F. A row can be deleted if all its positions have entropy 1, which can be checked with the previous optimization. In the example, the columns B and D can be deleted, as well as row 2. The resulting subtable has only four positions, and thus the number of subsets to be considered is reduced by a factor 2^{12}.

2.3.2. Approximation

For large tables, the entropy computation can be expensive. The number of subsets $Q \subseteq \mathrm{Pos} \setminus \{p\}$ for a considered position p grows exponentially with the table size. Hence, even small tables can have a long computation time. The aforementioned optimizations may possibly be insufficient, especially when the rows and columns to be deleted are rare. In a randomized approach, not all subsets are considered, but only a fixed number of randomly chosen ones. Depending on the number of samples, the result can be sufficiently accurate with a high probability. The approach is called the Monte Carlo method and introduced in [6]. Building on that method, we derived the entropy computation in our earlier work [3]. The method holds for general dependencies; thus our derivation also applies for multivalued dependencies.

Let $\mathrm{INF}_I(p \mid F, Q) := \lim_{k\to\infty} \frac{\log \#V_Q}{\log k}$ represent the information content of position p in instance I with respect to a set F of functional or multivalued dependencies with $I \models F$, given a fixed subset $Q \subseteq \mathrm{Pos} \setminus \{p\}$ of lost values. We define random variables

$$X_i : \mathcal{P}(\mathrm{Pos} \setminus \{p\}) \to [0, 1], \quad Q \mapsto \mathrm{INF}_I(p \mid F, Q).$$

Then $\mathrm{E}[X_i] = \mathrm{INF}_I(p \mid F)$ and for independent identically distributed $X_1, \ldots, X_n$ as above and their average $X := \frac{1}{n} \sum_{i=1}^{n} X_i$, for all $\varepsilon, \delta > 0$ it holds

$$\Pr\big(|X - \mathrm{INF}_I(p \mid F)| \geqslant \varepsilon\big) \leqslant \delta$$

provided that $n \geqslant 2\ln(2/\delta)/\varepsilon^2$. Here, ε stands for the accuracy of the approximation, δ for an error probability, thus the accuracy ε holds with a confidence of $1 - \delta$ for a number of samples n as required in the inequality.

We show an example in which the number of sufficient iterations is computed such that the entropies have a fixed accuracy under a specified confidence.

Example 2.8. An accuracy of 0.01 with a confidence of 99% can be achieved with at least $2\ln(2/0.01)/0.01^2 \approx 100,000$ iterations. For an accuracy of 0.04 with a confidence of 99.9% the sufficient sample size is $2\ln(2/0.001)/0.04^2 \approx 10,000$.

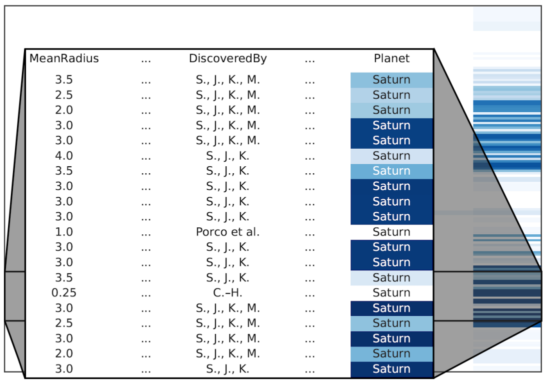
3. Related Work

Functional dependencies are a well-studied and established research field. An approach to visualize functional dependencies is a sunburst diagram [1]. This visualization can help humans process a large number of functional dependencies, but it is only on the schema level.

Conversely, Lee [5] proposes an information-theoretic approach to analyze relational databases. In contrast to the “plaque test”, given a concrete instance, the self-information of a set of attributes using entropies is computed, but without considering the entropies of individual cells. We are not aware of a visualization of this framework, however, this only considers the instance level.

The “plaque test” (see our earlier work in [3]) takes schema and instance into account. For a given instance and a set of functional dependencies, the information content of each cell is calculated and visualized using the term of entropy. This information content is directly connected with the redundancy of the cell, and each cell becomes a shade of blue depending on its degree of redundancy, like a heat map. Intuitively, this means that depending on the given functional dependencies and other tuples in the relation, lost values in a cell can be restored using these dependencies and the other tuples. The more available tuples or functional dependencies we have, the higher is the chance to restore a lost value, and thus the higher is the redundancy in this considered cell.

The theoretical framework for the “plaque test” is provided in [2]. This work introduces the term information content generally for all kinds of constraints in relational databases. The computations for the “plaque test” are simplified in our earlier work [3] and implemented only for functional dependencies, and the next obvious step is to extend this implementation for multivalued dependencies. While the theoretical considerations hold for



MeanRadius	...	DiscoveredBy	...	Planet
3.5	...	S., J., K., M.	...	Saturn
2.5	...	S., J., K., M.	...	Saturn
2.0	...	S., J., K., M.	...	Saturn
3.0	...	S., J., K., M.	...	Saturn
3.0	...	S., J., K., M.	...	Saturn
4.0	...	S., J., K.	...	Saturn
3.5	...	S., J., K.	...	Saturn
3.0	...	S., J., K.	...	Saturn
3.0	...	S., J., K.	...	Saturn
3.0	...	S., J., K.	...	Saturn
1.0	...	Porco et al.	...	Saturn
3.0	...	S., J., K.	...	Saturn
3.0	...	S., J., K.	...	Saturn
3.5	...	S., J., K.	...	Saturn
0.25	...	C.–H.	...	Saturn
3.0	...	S., J., K., M.	...	Saturn
2.5	...	S., J., K., M.	...	Saturn
3.0	...	S., J., K., M.	...	Saturn
2.0	...	S., J., K., M.	...	Saturn
3.0	...	S., J., K.	...	Saturn

Figure 3: "Plaque test" applied to a real-world dataset with 150 rows and a minimum entropy of 0.61 (rounded). The color scale is normalized with respect to the minimum entropy value. The zoom-in highlights a subset of rows and hides white cells only. Figure adapted from [3].

multivalued dependencies, the "plaque test" optimizations [3] are only proven for functional dependencies.

4. Preliminary Results

In this section, we present the results of the "plaque test" on a larger real-world dataset. Details on the implementation and the experimental setup can be found in [3]. The informativeness, the optimizations, and the scalability are discussed. Then we show an extension to a given set of multivalued dependencies on a small example and make considerations about the results and possible adjustments of the optimizations and the Monte Carlo method.

4.1. Results for Functional Dependencies

As a real-world example, we used the first 150 rows of a dataset that describes natural satellites and originates from the WDC Web Table Corpus[2] with 35 valid simple functional dependencies. We analyzed the informativeness of the "plaque test", the scalability of the exact computations, and the improvement of the runtimes with the Monte Carlo method from Section 2.3.2.

Informativeness. Figure 3 shows the "plaque test" for the dataset with entropies calculated using the Monte Carlo method with 100,000 iterations, thus an accuracy of about 0.01 with a confidence of 99% (see Example 2.8). White cells have entropies of exactly 1, i.e., no redundancy, while blue cells have values below 1, where darker hues have lower entropies. Redundancies are concentrated in the last column "Planet". It appears that there are several functional dependencies with "Planet" on the right-hand side, which cause the redundancies in the last column. This consideration is a good indicator for the informativeness of the "plaque test".

Exact computation. The following results are stated in the earlier work [3]. Without the optimizations from Section 2.3.1, only the entropies of the relation containing the first three rows can be computed within 24 hours. With the optimizations, two more rows are manageable in this dataset. The optimizations reduce the runtime for four rows from more than 24 hours to less than one second, which shows their efficiency. The unoptimized case shows exponential complexity. In the optimized case, the runtime stays small for up to four rows since the relevant subtables are small in this concrete instance. Adding the fifth line causes more redundancies, which increases the relevant table, and thus the computation time significantly. Hence, the optimizations are only practicable for small tables with a small number of redundancies, such that the table size can be efficiently reduced. The reduction potential depends on the concrete given table.

Monte Carlo method. The runtime of the Monte Carlo method grows asymptotically linear with the number of iterations. The number of rows mathematically causes a quadratic growth of the runtime, since for the verification of functional dependencies, pairs of rows have to be considered. The numbers in [3] verify this assumption. For the first 150 rows of the satellite dataset, 100,000 iterations can be computed in less than 30 minutes, which is a significant improvement over the exact computation. This approximation costs little accuracy, but for 100,000 iterations, the error is within 0.01 around the exact value with a probability of 99%. If an accuracy of 0.04 with a confidence of 99.9% suffices, the number of iterations can be reduced to 10,000 (see Example 2.8), reducing the runtime for the first 150 rows to less than three minutes.

Entropy values are mainly used to scale the hues for visualization, thus small errors compared with the range of values are not exactly noticeable by a human eye. Reducing accuracy requirements opens possibilities to handle datasets beyond 150 rows. Finding sufficient accuracies for satisfactory visualizations motivates further research.

4.2. First Steps for Multivalued Dependencies

We now consider entropies in instances with multivalued dependencies. The relation[3] in Figure 4a manages university courses and attributes "Course", "Book" and "Lecturer". The following multivalued dependencies hold:

$$\text{Course} \twoheadrightarrow \text{Book} \quad \text{and} \quad \text{Course} \twoheadrightarrow \text{Lecturer} \qquad (\star\star)$$

[2] http://webdatacommons.org/webtables/index.html#results-2015

[3] Example adapted from English Wikipedia site on multivalued dependencies, at https://en.wikipedia.org/wiki/Multivalued_dependency, last accessed April 2024.

Course	Book	Lecturer
AHA	Silber…	John…
AHA	Neder…	John…
AHA	Silber…	Will…
AHA	Neder…	Will…
AHA	Silber…	Chris…
AHA	Neder…	Chris…
OSO	Silber…	John…
OSO	Silber…	Will…
SQL	Introd…	Raym…

(a) The relational input data.

Course	Book	Lecturer
0.9999	0.9968	0.9958
0.9986	0.9992	0.9957
0.9992	0.9975	0.9996
0.9999	0.9989	0.9994
0.9997	0.9993	0.9998
0.9999	0.9996	0.9998
1	0.9998	1
1	0.9998	1
1	1	1

(b) Entropies for (⋆⋆).

Figure 4: Plaque test for the given relation with multivalued dependencies from (⋆⋆). Hues correspond to entropy values.

Figure 4 shows the plaque test, with these two multivalued dependencies as input. It contains the relation in Figure 4a and a table with entropies and colored cells, analogously to entropies for functional dependencies, in Figure 4b. While full entropies are exact, the other values are rounded to four digits after the dot.

The resulting table shows small redundancies compared to those caused by functional dependencies in the introductory example. This indicates that the subsets of lost values such that a value in a considered position can be restored using the given multivalued dependencies and the rest of the relation instance are very rare.

Applying the Monte Carlo method can lead to false positive cases of full information content, most probably if the chosen number of iterations is small. This error appears when there is no randomly chosen subset such that the value in a considered position can be restored. This consideration and the small computed redundancies indicate that the Monte Carlo method in instances with multivalued dependencies should be applied with a better accuracy than with functional dependencies.

Finding reasons for the small differences in the entropies over the relation can be a challenge. Another opportunity for future work will be to find optimizations analogously to the case of functional dependencies. It involves finding a necessary and sufficient condition for cells with full information content, so that false positive entropies of 1 can be excluded. It also contains finding conditions for reducing the problem size to make bigger problems practically computable.

5. Open Research Challenges

We see several future challenges in our visualization approach of redundancies in relational data. We discuss the problems of optimizing the computation effort for constraint sets with multivalued dependencies, the scalability of the computation when applying the Monte Carlo method and a user study to elaborate the potential of using the “plaque test” for teaching purposes.

Optimizations for multivalued dependencies. The optimizations from Section 2.3.1 hold in relations with functional dependencies, however, cannot be immediately transferred to multivalued dependencies. Hence, the reduction of the problem size seems to be more complicated with multivalued dependencies involved. Even if only a part of the attributes appears in a multivalued dependency, the value of all attributes can be relevant for its validation. This reduces the possibility to delete columns in the original relation. Hence, finding optimizations holding also for multivalued dependencies, which are backed by formal proofs, requires theoretical research.

Scalability. Even with the prior simplifications, there are many scenarios where the entropy computation is practically out of reach. The Monte Carlo method avoids exponential runtime. For functional dependencies, 150 tuples are manageable. One goal is to execute computations for even larger datasets, possible approaches to address this are parallelization and dynamic programming.

Due to many entropies, caused by multivalued dependencies, close to 1, false positive entropies of 1 are likely when applying the Monte Carlo method only. Without an optimization to find cells with full information content immediately, these false positives cannot be identified. Hence, finding a combination of optimizations and approximation, as for functional dependencies but adapted to multivalued dependencies, is another challenge.

User study. We are planning to use the “plaque test” for an experiment involving students. Our assumption is that a visualization of redundancies can be helpful in finding normalization approaches. The question is whether students using such a visualization tool can easily suspect ways for a suitable decomposition of a schema containing several redundancies and the user study shall give us an idea if the “plaque” is helpful. If the results of the study confirm our assumption, we see potential for the “plaque test” as a tool to support the teaching of normal forms. The experimental schemas shall contain functional as well as multivalued dependencies. The results can be helpful to understand the interpretability of the plaque.

Summarizing all the possible challenges, a combination of theory, algorithmic, and user study requires skills in different disciplines and allows us to view the problem of redundancies from several perspectives.

Acknowledgments

I thank Stefanie Scherzinger, Jens Zumbrägel, and Stefan Klessinger for their valuable discussions, suggestions, and support on this topic.

References

[1] S. Kruse, D. Hahn, M. Walter, F. Naumann, Metacrate: Organize and Analyze Millions of Data Profiles, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017.

[2] M. Arenas, L. Libkin, An Information-theoretic Approach to Normal Forms for Relational and XML Data, in: Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, 2003.

[3] C. Köhnen, S. Klessinger, J. Zumbrägel, S. Scherzinger, A Plaque Test for Redundancies in Relational Data, in: Joint Proceedings of Workshops at the 49th International Conference on Very Large Data Bases, 2023.

[4] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995.

[5] T. Lee, An Information-Theoretic Analysis of Relational Databases—Part I: Data Dependencies and Information Metric, IEEE Transactions on Software Engineering SE-13 (1987) 1049–1061.

[6] M. Mitzenmacher, E. Upfal, Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis, Cambridge University Press, 2017.