Dependency-based Query Result Approximation
(extended abstract)

Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese

Department of Computer Science, University of Salerno,
via Giovanni Paolo II n.132, 84084 Fisciano (SA), Italy
{lcaruccio,deufemia,gpolese}@unisa.it

Abstract. Failing queries are database queries returning few o no results. It might be useful reformulating them in order to retrieve results that are close to those intended with original queries. In this paper, we introduce an approach for rewriting failing queries that are in the disjunctive normal form. In particular, the approach prescribes to replace some of the attributes of the failing queries with attributes semantically related to them by means of Relaxed Functional Dependencies (RFDs), which can be automatically discovered from data. The semantics of automatically discovered RFDs allow us to rank them in a way to provide an application order during the query rewriting process. Experiments show that such application order of RFDs yields a ranking of the approximate query answers meeting the expectations of the user.

Keywords: query rewriting, query relaxation, relaxed functional dependencies

1 Introduction

A common problem in query processing is coping with failing queries, that is, queries returning few or no answers. To this end, several techniques have been proposed to relax queries in order to make them return also approximate answers, so as to broaden the answer set. Manually relaxing failing queries is a time-consuming task, and if the query is over-relaxed, prohibitive costs are paid in terms of bandwidth per returned tuple. Thus, researchers have proposed automated approaches to query relaxation [15, 17–19], but the existing algorithms have several limitations, mostly due to a poor knowledge about the characteristics of the database instance under examination. To this end, automatic data profiling techniques are becoming available [1], which are capable of providing useful metadata concerning the characteristics of a database instance, including data dependencies, domain cardinalities, data quality constraints, and so on. They can be exploited for several purposes, including optimizations related to

SEBD 2018, June 24-27, 2018, Castellaneta Marina, Italy.

query processing, such as the optimization of query executions, rewriting queries and views upon schema evolutions [8], and so on.

In this paper we describe an approach exploiting Relaxed Functional Dependencies (RFDs) [5] to relax the results of failing queries [7]. In order to explain our approach, in what follows we provide a running example, which will be used throughout the paper.

	Make	Model	Price ($)	Location	Year	Start Production Year	Color	Mileage	Length (mm)	Power (kw)	EngineCC	Torque (Nm)	Consumption (US mpg) Urban/Extra Urban/Combined	Fuel
1	Ford	Focus 1.6tdci	6500	Pittsburgh (PA)	2007	2004	Red	60000	4342	80	1560	260	37.97/58.78/49.04	Diesel
2	Mazda	3 td	8000	Denver (CO)	2009	2009	Gray	45000	4580	80	1560	240	44.38/60.28/53.45	Diesel
3	Ford	Focus 1.6tdci	11000	San Diego (CA)	2010	2010	Blue	15000	4337	85	1560	270	52.29/69.19/61.86	Diesel
4	Ford	Focus 1.6i 16V	3000	Austin (TX)	2005	2004	Blue	48000	4342	73.8	1596	150	21.56/38.55/30.14	Gasoline
5	Volkswagen	Golf 1.6tdi	12000	Las Vegas (NV)	2008	2008	Black	70000	4199	77	1598	250	41.30/60.28/52.29	Diesel
6	Ford	Focus 1.6tdci	12300	Seattle (WA)	2011	2010	Gray	8000	4337	85	1560	270	52.29/69.19/61.86	Diesel
7	Volkswagen	Golf 1.6tdi	14000	Miami (FL)	2009	2008	Gray	50000	4199	77	1598	250	41.30/60.28/52.29	Diesel
8	Toyota	Avensis 1.8 16V SW	6000	New Orleans (LU)	2005	2003	Red	38000	4700	95	1794	170	25.06/40.55/32.64	Gasoline
9	Ford	Focus 1.6tdci	5000	Chicago (IL)	2006	2004	White	66000	4342	80	1560	260	37.97/58.78/49.04	Diesel
10	Ford	Fiesta 1.4 16V	7000	Charleston (SC)	2009	2008	Black	27000	3950	71	1388	125	31.39/51.12/41.30	Gasoline

Table 1. A sample of the car selling database.

Example 1. Let us consider the car selling database `CarDealerDB` of Table 1. There are several usage scenarios for this database. For instance, a user might search cars available for sale or examine detailed characteristics of some specific models.

Suppose the user is looking for a Ford `Focus` with a price around `8000$`. Then s/he might enter the following query:

$Q \equiv$ `Model like '%Focus%' AND Price >= 7500 AND Price <= 8500`

By executing it on the `CarDealerDB` no tuples will be returned. However, some similar cars might have characteristics close to the requested one. Such similarities cannot be taken into account by the query, since we assume that no function is available for evaluating the similarity of categorical attributes. To this end, the approach we propose in this paper exploits automatically discovered RFDs to rewrite queries so as to enable them generate also approximate results. The semantics of RFDs enable us to derive ranking techniques that can be used to decide the application order of RFDs during the query rewriting process. To this end, we also provide experimental results proving that such application order yields approximating query answers meeting the expectations of the final user.

The paper is organized as follows. Section 2 discusses the related work, while Section 3 presents background information on RFDs. Then, we describe our query rewriting approach in Section 4 and discuss its empirical evaluation in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

Approximation in query answering has been extensively studied in the recent years. Most of the early efforts were devoted to reduce response time by seek-

ing approximate query answers. Some techniques performed data summaries, statistics or histograms to compute approximate answers [20]. The Aqua system focusses on the fast generation of approximate results, in order to improve efficiency [2]. The work in [21] introduces the problem of approximate data exchange, aiming to produce fast approximate answers, and successively exact answers.

The AIMQ query relaxation method removes some constraints from the query based on approximate functional dependencies [19]. It learns the attribute importance based on pre-extracted data, ranking the relevant answer tuples by using the similarities between an imprecise query and answer tuples. The AQRR method [15] is similar to AIMQ, but it exploits user preferences for deriving the attribute importance and to evaluate similarities between an imprecise query and answer tuples.

Muslea proposed a method adopting machine learning to learn rules from the database [17]. In particular, for each failed query, it will find the most similar rule for generating alternative queries. The query relaxation method proposed in [18] trains the system beforehand on portion of the data. The focus is on queries with conjunctions or disjunctions of atoms and the relaxation procedure is based on bayesian networks, targeting the approximation of domain knowledge. The approach is data driven, but a similar service can be achieved by exploiting schema information, and in particular integrity constraints, which in turn can be used to inform the user about the query failing conditions [13]. The approach proposed in [11] focuses on how to explain non-answer queries by pinpointing the constraint causing the empty result. The approach proposed in [9] allows to modify the query based on the notion of generalization, identifying the conditions under which a generalization is applicable. Koudas *et al.* [14] suggest alternative queries based on the "minimal" shift from the original one.

ORange is a system automatically assisting the user in the process of query refinement, aiming to satisfy a specific cardinality constraint [3]. The system exploits a similarity-aware a query refinement schema, which is also able to maximize its similarity w.r.t. to the original range query.

Finally, the framework proposed in [16] is able to relax queries in an interactive fashion, based on a process aiming to optimize a wide variety of application-dependent objective functions. In particular, given an initial query returning an empty-answer set, the framework dynamically computes and suggests alternative queries with fewer conditions than those initially requested, in order to help the user derive a query with a non-empty-answer.

3 Relaxed Functional Dependencies

In this section we will review the definition of relaxed functional dependency (RFD).

Consider a relational database schema $\mathcal{R}$ defined over a set of attributes $attr(\mathcal{R})$, derived as the union of attributes from the relation schemas composing $\mathcal{R}$. For an instance r of $\mathcal{R}$ and a tuple $t \in r$, we use $t[A]$ to denote the projection

of t onto A; similarly, for a set X of attributes in $attr(\mathcal{R})$, $t[X]$ denotes the projection of t onto X.

Definition 1. Functional Dependency (FD). *An FD over $\mathcal{R}$ is a statement $X \rightarrow Y$ (X implies Y), with $X, Y \subseteq attr(\mathcal{R})$, such that, given an instance r of $\mathcal{R}$, $X \rightarrow Y$ is satisfied in r if and only if for every pair of tuples (t_1, t_2) in r, whenever $t_1[X] = t_2[X]$, then $t_1[Y] = t_2[Y]$.*

In the last part of the FD definition, we notice that the projections of two tuples over a subset of attributes are compared by means of the equality function. This is one of the two dimensions that have been modified in order to define RFDs, by enabling the use of tuple comparisons based on a *similarity constraint*. The latter, defined as $\phi_{\leq\alpha}$, can be expressed in terms of a similarity metric $\approx$, such that $a \approx b$ is true if a and b are "close" enough w.r.t. a predefined threshold (α). Examples of similarity metrics are the edit or the Jaro distance [10].

Another important characteristic of the FD definition is that it specifies a property of the database schema that must hold on every instance of it. This is the second dimension that has been modified to derive a general definition of RFD, which admits the possibility that the property might hold for a subset rather than all the tuples. The latter can be formally specified by means of a *condition* filtering the tuples on which the dependency applies, or by means a *coverage measure*. Given a database instance r of $\mathcal{R}$, and two sets of attributes $X, Y \subseteq attr(\mathcal{R})$, representing the Left-Hand-Side (LHS) and Right-Hand-Side (RHS), resp., of an RFD φ, a coverage measure Ψ on φ quantifies the amount of tuples in r violating or satisfying φ. It can be defined as a function $\Psi : dom_X \times dom_Y \rightarrow \mathbb{R}$, where dom_A is the domain of attribute A. As an example, the *confidence measure* evaluates the maximum number of tuples $r_1 \subseteq r$ such that φ holds in r_1 [12].

In what follows, we provide a formal definition of RFD.

Definition 2. Relaxed Functional Dependency (RFD). *Consider a relational database schema $\mathcal{R}$, and a relation schema $R = (A_1, \ldots, A_k)$ of $\mathcal{R}$. An RFD φ on $\mathcal{R}$ is denoted by*

$$\mathbb{D}_c : X_{\Phi_1} \xrightarrow{\Psi \geq \epsilon} Y_{\Phi_2} \tag{1}$$

where

- $c = (c_1, \ldots, c_k)$ is the set of conditions constraining the domain $\mathbb{D}$ on which φ applies;
- $X, Y \subseteq attr(R)$ with $X \cap Y = \emptyset$;
- Φ_1 (Φ_2, resp.) is a set of similarity constraints $\phi[X]$ ($\phi[Y]$, resp.).
- Ψ is a coverage measure defined on $\mathbb{D}_c$.
- ϵ is a threshold indicating the bound for the result of the coverage measure.

Given $r \subseteq \mathbb{D}_c$ a relation instance on R, r *satisfies* the RFD φ, denoted by $r \models \varphi$, if and only if: $\forall\ t_1, t_2 \in r$, if $\phi[X]$ indicates true for each constraint $\phi \in \Phi_1$, then *almost always* $\phi[Y]$ indicates true for each constraint $\phi \in \Phi_2$. Here, *almost always* means that $\Psi(\pi_X(r), \pi_Y(r)) \geq \epsilon$.

In other words, if $t_1[X]$ and $t_2[X]$ agree with the constraints specified by Φ_1, then $t_1[Y]$ and $t_2[Y]$ agree with the constraints specified by Φ_2 with a degree of certainty (measured by Ψ) greater than ϵ.

As an example, in a database of scientific publications it is likely to have the same address and affiliation for authors with the same name. Thus, an FD $\{\mathsf{Author}\} \rightarrow \{\mathsf{Address}, \mathsf{Affiliation}\}$ might hold. However, these attributes might have been stored using different abbreviations. Thus, the following RFD might hold:

$$\mathbb{D}_{\text{TRUE}} : \mathsf{Author}_{\approx} \xrightarrow{\Psi_{err(0)}} \{\mathsf{Address}_{\approx}, \mathsf{Affiliation}_{\approx}\}$$

where $\approx$ is a string similarity function, and $\Psi_{err(0)}$ corresponds to the expression $\psi(X, Y) = 0$, where $\psi(X, Y)$ measures the number of tuples violating the RFD. However, authors might change affiliation during their life, or there might be homonimies, possibly caused by first name abbreviations. As a consequence, the previous RFD should tolerate possible exceptions. This can be modeled by introducing a different coverage measure into the RFD, making it conditional:

$$\mathbb{D}_{\text{TRUE}} : \mathsf{Author}_{\approx} \xrightarrow{\psi(Author, Address, Affiliation) \leq 0.02} \{\mathsf{Address}_{\approx}, \mathsf{Affiliation}_{\approx}\}$$

4 Methodology

Let us consider the car selling database (`CarDealerDB`) shown in Table 1. Among several usage scenarios for this database, let us consider one in which a user might search cars available for sale or examine detailed features for some specific car models. In particular, suppose the user is looking for a Ford `Focus` with a price around `8000$`. Then s/he might enter the following query:

$Q \equiv$ `Model like '%Focus%' AND Price >= 7500 AND Price <= 8500`

By executing this query on `CarDealerDB` no tuples will be returned as result. However, some cars close to the query request might be returned, if we were capable of evaluating the similarity of categorical attributes. To this end, we propose an approach exploiting RFDs, in order to rewrite a query so as to enable it generate approximated results. The rewriting process consists in replacing attributes instantiated in the query with those related to them by means of RFDs. As an example, let us assume that the RFD

$$\mathbb{D}_{\text{TRUE}} : \mathsf{Model}_{\approx_t} \xrightarrow{\psi(Model, Length) \leq 0.1} \mathsf{Length}_{\approx_n}$$

holds on `CarDealerDB`, where $\approx_t$ and $\approx_n$ are proper text and numerical similarity functions, respectively. The RFD says that in 90% of cases, cars with similar models (categorical attribute) must have similar length (numeric attribute). We can therefore infer that, cars whose `Length` is similar to the `Focus` one are more suitable to be returned as result.

More generally, let us suppose that a query Q has a condition $X \; OP \; x$, where OP is a comparison operator, and the following RFD holds on the underlying database:

$$\mathbb{D}_{\text{TRUE}} : \mathsf{X}_{\approx_f} \xrightarrow{\psi(X,Y) \leq \epsilon} \mathsf{Y}_{\approx_g}$$

then we might rewrite Q in Q_1 OR ... OR Q_m, where Q_i is obtained from Q by replacing the condition $X\ OP\ x$ with a condition $Y \approx_g y_i$, where y_i is the value of Y for a tuple in which $X\ OP\ x$. Such process can be better formalized by using similarity subsets generated by inference algorithms for RFDs [6].

From the example in Table 1 it follows that cars with `Model like '%Focus%'` satisfy either `Length = 4337` or `Length = 4342`. Thus, Q can be rewritten into Q_1 OR Q_2, where:

$Q_1 \equiv$ `Length`$\geq 4337-\varepsilon_i$ `AND Length` $\leq 4337+\varepsilon_i$ `AND Price` ≥ 7500 `AND Price` ≤ 8500
$Q_2 \equiv$ `Length`$\geq 4342-\varepsilon_i$ `AND Length` $\leq 4342+\varepsilon_i$ `AND Price` ≥ 7500 `AND Price` ≤ 8500

When computing Q_1 and Q_2 for $\varepsilon_i = 250$, we get the tuple t_2 as result, which corresponds to a Mazda 3, since its Length is similar to the one of Focus, and the Price matches the one of the original query.

It is worth to note that the attribute X of the query Q can also be a portion of the LHS of some RFDs, e.g.,

$$\mathbb{D}_{\text{TRUE}} : \mathsf{A}_{\approx_{f_1}} \mathsf{X}_{\approx_{f_2}} \mathsf{B}_{\approx_{f_3}} \xrightarrow{\psi(A,X,B,Y)\leq\epsilon} \mathsf{Y}_{\approx_g}$$

In this case, supposing that A, B, Y are textual attributes, we relax Q by replacing $X\ OP\ x$ with a condition "($Y \approx_g y_1$ AND $A \approx_{f_1} a_1$ AND $B \approx_{f_3} b_1$) OR ... OR ($Y \approx_g y_n$ AND $A \approx_{f_1} a_n$ AND $B \approx_{f_3} b_n$)", where y_i is the value of Y for a tuple in which $X\ OP\ x$, $A\ OP\ a_i$, and $B\ OP\ b_i$. Thus, we do not consider the whole $Y \approx_g y$ category, but its subset consisting of the tuples having the values for A and B similar to one of the tuples where $X\ OP\ x$. For instance, since the following dependency

$$\mathbb{D}_{\text{TRUE}} : \mathsf{Model}_{\approx_{f_1}} \mathsf{Power}_{\approx_{f_2}} \mathsf{EngineCC}_{\approx_{f_3}} \xrightarrow{\psi(X,Y)\leq 0.05} \mathsf{Torque}_{\approx_g}$$

holds on the `CarDealearDB`, we can relax the condition `Model like '%Focus%'` with
`((Torque` ≥ 240 `AND Torque` ≤ 280`) AND (EngineCC` ≥ 1530 `AND EngineCC` ≤ 1590`) AND (Power` ≥ 75 `AND Power` ≤ 85`)) OR`
`((Torque` ≥ 250 `AND Torque` ≤ 290`) AND (EngineCC` ≥ 1530 `AND EngineCC` ≤ 1590`) AND (Power` ≥ 80 `AND Power` ≤ 90`)) OR`
`((Torque` ≥ 130 `AND Torque` ≤ 170`) AND (EngineCC` ≥ 1566 `AND EngineCC` ≤ 1626`) AND (Power` ≥ 68.8 `AND Power` ≤ 78.8`))`.

However, in practice the rewriting process of the query might be accomplished according to different RFDs. Hence, we need to establish a priority in the application of the different RFDs during the relaxation process, so as to meet user preferences. To this end, we found out that current RFD discovery algorithms rank the extracted RFDs based on the coverage and similarity thresholds [6], and only extract non-trivial RFDs, discarding redundant ones. Thus, by following the same ranking order used in RFD discovery algorithms, we naturally meet user preferences, since it is expected that the user would prefer relaxing queries by first using RFDs with high coverage and high similarities.

Datasets	Precision	Recall
Breast-Cancer	0.85	0.92
Hepatitis	0.81	0.89
Lymphography	0.82	0.93

Table 2. Precision and recall obtained for the considered datasets.

5 Experimental Results

We evaluated the proposed approach on three different datasets. In particular, we considered the Breast-cancer, the Hepatitis, and the Lymphography datasets drawn from the UC Irvine Machine Learning repository [4].

In order to evaluate the performances of the proposed approach, we have defined 5 failing queries for each considered dataset. Successively, we have requested a domain expert to analyze the datasets and the queries in order to identify the ten tuples most similar to the target result of each query (ground truth). Finally, we have (*i*) applied the proposed approach to rewrite the five failing queries, (*ii*) run them on the considered dataset, and (*iii*) compared their results to the ground truth.

We measured the effectiveness of the proposed approach with precision and recall. Let A be the set of tuples obtained from the queries generated by the proposed approach and B the ground truth identified by the expert. Then, we computed the *precision* as $|A \cap B|/|A|$, while *recall* as $|A \cap B|/|B|$ [22]. The results of precision and recall for each dataset are shown in Table 2.

As we notice, we obtained high values for both measures, especially the recall. We can observe that the number of RFDs impacts on the quality of the resulting queries. This is due to the fact that ranking strategies used in current RFD discovery algorithms suffer from some noise when the datasets contain many RFDs.

6 Conclusions and Future Work

We have described an approach for rewriting failing queries that are in disjunctive normal form [7]. It relies on RFDs, which capture important constraints among attributes. Automatically extracted RFDs provide parameters enabling us to derive the priority by which they should be applied during the query relaxation process. We have experimentally verified that such priorities meet user expectations in terms of query results.

In the future we plan to define new ranking strategies for RFDs, in order to better refine their application order, and to further improve the performances of the proposed query rewriting technique. Moreover, we are currently investigating further query rewriting rules exploiting additional semantic properties of RFDs.

References

1. Abedjan, Z., Golab, L., Naumann, F.: Profiling relational data: a survey. The VLDB Journal **24**(4), 557–581 (2015)
2. Acharya, S., Gibbons, P.B., Poosala, V., Ramaswamy, S.: The Aqua approximate query answering system. In: SIGMOD. pp. 574–576 (1999)
3. Albarrak, A., Noboa, T., Khan, H.A., Sharaf, M.A., Zhou, X., Sadiq, S.: ORange: Objective-aware range query refinement. In: MDM. pp. 333–336 (2014)
4. Blake, C.L., Merz, C.J.: UCI repository of machine learning databases (1998), `http://www.ics.uci.edu/mlearn/MLRepository.html`
5. Caruccio, L., Deufemia, V., Polese, G.: Relaxed functional dependencies – A survey of approaches. IEEE TKDE **28**(1), 147–165 (2016)
6. Caruccio, L., Deufemia, V., Polese, G.: On the discovery of relaxed functional dependencies. In: IDEAS. pp. 53–61 (2016)
7. Caruccio, L., Deufemia, V., Polese, G.: Learning effective query management strategies from big data. In: ICMLA. pp. 643–648 (2017)
8. Caruccio, L., Polese, G., Tortora, G.: Synchronization of queries and views upon schema evolutions: A survey. ACM TODS **41**(2), 9:1–9:41 (2016)
9. Chaudhuri, S.: Generalization and a framework for query modification. In: ICDE. pp. 138–145 (1990)
10. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE TKDE **19**(1), 1–16 (2007)
11. Huang, J., Chen, T., Doan, A., Naughton, J.F.: On the provenance of non-answers to queries over extracted data. In: PVLDB. pp. 736–747 (2008)
12. Huhtala, Y., Kärkkäinen, J., Porkka, P., Toivonen, H.: TANE: An efficient algorithm for discovering functional and approximate dependencies. The Computer Journal **42**(2), 100–111 (1999)
13. Janas, J.M.: On the feasibility of informative answers. In: Advances in Data Base Theory, pp. 397–414. Springer (1981)
14. Koudas, N., Li, C., Tung, A.K.H., Vernica, R.: Relaxing join and selection queries. In: VLDB. pp. 199–210 (2006)
15. Meng, X., Ma, Z.M., Yan, L.: Answering approximate queries over autonomous web databases. In: WWW. pp. 1021–1030 (2009)
16. Mottin, D., Marascu, A., Roy, S.B., Das, G., Palpanas, T., Velegrakis, Y.: A holistic and principled approach for the empty-answer problem. The VLDB Journal **25**(4), 597–622 (2016)
17. Muslea, I.: Machine learning for online query relaxation. In: KDD. pp. 246–255 (2004)
18. Muslea, I., Lee, T.: Online query relaxation via bayesian causal structures discovery. In: AAAI. pp. 831–836 (2005)
19. Nambiar, U., Kambhampati, S.: Mining approximate functional dependencies and concept similarities to answer imprecise queries. In: WebDB. pp. 73–78 (2004)
20. Poosala, V., Ganti, V.: Fast approximate query answering using precomputed statistics. In: ICDE. p. 252 (1999)
21. de Rougemont, M., Vieilleribiére, A.: Approximate data exchange. In: ICDT. pp. 44–58 (2007)
22. Salton, G.: Introduction to Modern Information Retrieval. McGraw-Hill (1983)