Coverage-based Rewriting for Data Preparation

Chiara Accinelli
University of Genoa, Italy
chiara.accinelli@dibris.unige.it

Simone Minisi
University of Genoa, Italy
simone.minisi@dibris.unige.it

Barbara Catania
University of Genoa, Italy
barbara.catania@dibris.unige.it

ABSTRACT

The development of technological solutions satisfying non discriminating requirements is currently one of the main challenges for data processing. Concepts like *fairness*, i.e., lack of bias, and *diversity*, i.e., the degree to which different kinds of objects are represented in a dataset, have been recently taken into account in designing non-discriminating set selection, ranking, and OLAP approaches. Information extraction is however also at the basis of back-end data processing, for preparing, e.g., extracting and transforming data, usually based on SQL queries, before loading them inside a data warehouse for further front-end processing. The impact of an unfair data preparation process might have a relevant impact on front-end analysis. As an example, an under-represented category in the warehouse might lead to an under-representation of that category in most of the following processes. This kind of guarantee is known as *coverage*. In this paper, we start from this consideration and we propose an approach for automatically rewriting back-end queries, whose results do not guarantee some coverage constraints, into the "closest" queries satisfying those constraints. Through rewriting, coverage-based modifications of data preparation steps are traced for further processing. We also present some preliminary experimental results and we identify some directions for future works.

1 INTRODUCTION

Background. Nowadays, large-scale technologies for the management and the analysis of big data have a relevant and positive impact: they can improve people's lives and enhance scientific exploration. At the same time, it becomes increasingly important to understand the nature of these impacts at the social level and to take responsibility for them, especially when they deal with human-related data [7].

The development of technological solutions satisfying non-discriminating requirements is currently one of the main challenges in data processing and, in particular, in data management. Due to the above-mentioned social relevance, themes such as diversity, non-discrimination, fairness, protection of minorities, and transparency are becoming increasingly crucial when dealing with any data processing step. More concretely, consider a population upon which a data processing (either operational or analytical) task has to be applied. Suppose that a subset of our population shares some characteristics that should not be employed for discrimination (e.g., race, gender, disability status). It is important to guarantee that the result of the processing task is not discriminating with respect to the considered sensitive attributes. This may include ensuring a fair probability of selection, not giving undue relevance to individuals sharing these properties, or other related constraints. This type of requirements is made desirable by the ethical need to take responsibility; however, it is also made mandatory by the recent General Data Protection Regulation of the European Union [5]. The latter imposes that this type of guarantee is provided by design, i.e., intrinsically embedded in the mechanisms of the data processing workflow.

Among the relevant themes from a social point of view, various non-discriminating constraints, like *fairness*, i.e., lack of bias, and *diversity*, i.e., the degree to which different kinds of objects are represented in a dataset, have recently received attention from the data management community. As pointed out in [6, 8], an important direction is developing a holistic treatment of non-discriminating constraints through different stages of the data management and analysis life-cycle, from data cleaning, integration, and preprocessing, through selection and ranking, to result interpretation. In this context, a key issue concerns enforcing non-discriminating constraints incrementally through individual independent choices, rather than as a constraint on the set of final results.

Problem. Recently, various approaches to cope with non-discriminating constraints during front-end information extraction stages, aiming at presenting the users with data organized as to satisfy their needs, have been proposed, in the context of set selection [20], ranking [2, 3, 22], and OLAP [14, 15, 17].

Information extraction is however also at the basis of back-end processing in analytical systems, in which source data have to be prepared, namely extracted, transformed, and loaded, into an analytical repository - a data warehouse - for further front-end processing. This kind of processes is probably simpler from the point of view of data manipulation with respect to relevant front-end search functionalities and can often be represented in terms of Select-Project-Join (SPJ) queries. However, an unfair data preparation process might have a huge impact on many front-end processes: as an example, an under-represented category in the warehouse might lead to an under-representation of that category in any following process. As pointed out in [16], this motivates the study of fairness-aware data transformations, where the idea is to minimally rewrite the transformation query so that certain fairness constraints are guaranteed to be satisfied in the result of the transformation.

In this paper, we start from this considerations and we consider fairness constraints guaranteeing that there are enough entries in the dataset for each object category, before starting front-end processes. Such property has been called *coverage* in [4] and has relationships with both fairness and diversity issues [22]. More precisely, consider a (source) dataset I in which some items are characterized by a sensitive attribute (e.g, gender, race, age) and an SPJ query Q over I, including the sensitive attribute in the projection list. Now suppose that some coverage constraints are given with respect to the result of Q when executed over I (e.g., we would like the number of females is greater than a given threshold). Suppose Q, when executed over I, does not satisfy the constraints. We are interested in *rewriting* Q into a new query Q' such that Q' *relaxes* predicates in Q, thus, $Q \subseteq Q'$, and Q' is the *smallest* query containing Q satisfying the coverage constraints when executed over I.

Notice that we are interested in rewriting Q into Q' in order to achieve *transparency*: through rewriting, the coverage-aware

relaxation process is traced for further analysis or processing. In this respect, the approach we propose can be seen as a revision of existing query refinement techniques, addressing the empty or few answer problem (see, e.g., [12]), to take care of coverage and fairness issues.

Contributions. The main contributions of the paper can be summarized as follows:

- We formally define a query rewriting problem based on coverage constraints.
- We present a baseline algorithm for query rewriting, relaxing the query with the aim of satisfying the constraints at hand and we propose two optimizations based on pruning and incremental refinement.
- We present some preliminary experimental results showing that the proposed approach is effective and efficient when applied over some real datasets.

Organization. The remainder of this paper is organized as follows. Section 2 discusses related work about non-discrimination in information extraction and query relaxation. The problem we consider is then presented in Section 3. The proposed grid-based algorithm is introduced in Section 4 while experimental results are presented in Section 5. Finally, Section 6 outlines future works and presents some concluding remarks.

2 RELATED WORK

There are several areas of research that are related to our work, which we discuss below.

Non-discrimination in information extraction. Information extraction aims at presenting the users with data organized as to satisfy their needs. Non-discriminating information extraction takes into account non-discriminating properties, like fairness and diversity, in generating results to be returned to the user. Fairness and diversity constraints have been taken into account for designing set selection [20] and ranking [2, 3, 22] approaches. Fairness has also been considered to provide fairness-aware rewriting of OLAP, i.e., aggregation-based, queries [14, 15], and database repair approaches [17, 18] that provide provable fairness guarantees about classifiers trained on database training labels. The notion of causal fairness and the design of data management techniques for it have been investigated in [16]. Coverage over multiple categorical attributes has been introduced in [4], where efficient techniques for determining the least amount of additional data that must be obtained to guarantee coverage satisfaction have been proposed. A web application made up of a collection of visual widgets that implement most latest research results on fairness, stability, and transparency for rankings, and that communicate details of the applied methodology to the end-user, is presented in [19, 21]. In [1], an interaction model for explain-and-repair data transformation systems, in which users interactively define constraints for transformation code and the resultant data, is proposed. The system satisfies these constraints as much as possible and provides an explanation for any encountered problem. In this paper, we consider coverage constraints, as proposed in [4], over a single sensitive attribute. The problem we address is closely related to [1]. However, differently from [1], we consider transformations corresponding to SPJ queries with the aim of satisfying coverage constraints through an approach that can be easily integrated inside an SQL engine.

Query relaxation. Query relaxation is a well-studied problem in database literature, in which the goal is to help users in revising query specification or execution so that generated results better match users' desired results. Previous work has exploited query logs [9] and data being queried [23]. In order to address the empty or few answer problem, a skyline-based refined query execution has been proposed in [10] while an interactive framework for query relaxation through rewriting has been presented in [12]. The approach we propose can be seen as a reinterpretation of such last query relaxation framework to take care of coverage and fairness issues without any required user interaction.

3 PROBLEM DEFINITION

Let $D = \{R_1, ..., R_n\}$ be a relational database schema, let $S \in R_k$, $k \in \{1, ..., n\}$ be an attribute called *sensitive attribute* with domain $D_S = \{s_1, ...s_h\}$. Let Q be an arbitrary SQL query. Let $sel_1, ..., sel_d$ be the selection conditions in Q, $sel_i \equiv S_i \theta v_i$, attribute S_i has domain D_{S_i} and belongs to the schema of a relation R_j in D, $v_i \in D_{S_i}$. When needed, we denote Q by $Q\langle v_1, ..., v_d\rangle$ or $Q\langle \overline{v}\rangle$, $\overline{v} \equiv (v_1, ..., v_d)$. For the sake of simplicity, we assume that S_i, $i = 1, ..., d$, is a numeric attribute; the proposed approach can however be easily extended to deal with any ordered domain.

We restrict our attention to SQL queries Q in which: (i) the sensitive attribute S belongs to the projection list; (ii) $Q\langle \overline{v}\rangle \subseteq Q\langle \overline{u}\rangle$ when $u_i \geq v_i, i = 1, ..., d$ (denoted by $\overline{u} \geq \overline{v}$). Such queries are called *sensitive selection monotone* queries.

We then denote with $Q \downarrow_{s_j}$ the query $\sigma_{S=s_j}(Q)$, $j \in \{1, ..., h\}$, and with $|Q(I)|$ the cardinality of the result of Q when executed over a database instance I of D. For each value s_i of the sensitive attribute S, a constraint $CC_j \equiv |Q \downarrow_{s_i}| \geq k_i$, is called *coverage constraint with respect to* s_i or simply *coverage constraint.*

The problem we want to address is the following: given a database instance I, a sensitive selection monotone query $Q\langle \overline{v}\rangle$, and a set of coverage constraints $CC = \{CC_1, ..., CC_t\}$, $t \leq h$, not satisfied by $Q(I)$, rewrite $Q\langle \overline{v}\rangle$ into a new relaxed query $Q' \equiv Q\langle \overline{u}\rangle$ so that: (i) $Q \subseteq Q'$; (ii) $\forall j \in \{1, ..., t\}$, CC_j is satisfied by $Q'(I)$; (iii) there is no other query Q'' satisfying conditions (i) and (ii) such that $Q'' \subset Q'$ (thus, Q' is the smallest query satisfying (i) and (ii)); (iv) $Q' \equiv Q\langle \overline{u}\rangle$ is the closest query to $Q\langle \overline{v}\rangle$ according to the Euclidean distance between $\overline{v}$ and $\overline{u}$. Query Q' is called a *coverage-based rewriting of Q with respect to CC and I.*

Coverage constraints can be provided together with query Q or they can already be available in the system. This could be useful when they represent generally valid non-discrimination rules that must be satisfied by any query execution. Since the user might not be aware of the available constraints, we relax only selection predicates appearing in Q with the aim of generating relaxed queries that are syntactically close to Q (thus, potentially increasing user satisfaction). Additionally, differently from [12], we assume the user is not involved in the relaxation process to make the rewriting process "lighter" from the user point of view.

Consider a numeric selection predicate $sel_i \equiv S_i < v_i$. A *relaxation* of sel_i is any predicate $sel_i' \equiv S_i < v_i'$ s.t. $v_i' \geq v_i$. We can convert any predicate on a numeric domain to a predicate of the form $S_i < v_i$. For instance, a predicate $S_i > v_i$ can be transformed into $-S_i < -v_i$; range predicates of the form $S_i^l < v_i < S_i^u$ can be transformed into two separate predicates $-S_i < -v_i^l$ and $S_i < v_i^u$. In the rest of the paper, for ease of exposition, we assume that numeric predicates have been appropriately transformed into predicates of the form $S_i < v_i$. We rewrite the equality operator $S_i = v_i$ as a conjunction of $S_i \geq v_i$ and $S_i \leq v_i$.

4 A GRID BASED APPROACH FOR COVERAGE-BASED REWRITING

Let $Q\langle\overline{v}\rangle$ be a sensitive selection monotone query and $CC \equiv \{CC_1, ..., CC_t\}$ be a set of coverage constraints. Relaxed queries generated through coverage-based rewriting starting from $Q\langle\overline{v}\rangle$ and CC have the form $Q\langle\overline{u}\rangle$, with $\overline{u} \geq \overline{v}$, and can be represented as points $\overline{u}$ in a d-dimensional space.

Let I be an instance of D and suppose that each selection attribute S_j assumes in I values in $\overline{D}_{S_j} \equiv [a_j^{min}, ..., a_j^{max}]$, $j = 1, ..., d$. In order to navigate the d-dimensional space in a discrete way, we assume to discretize $\overline{D}_{S_j}$ into a fixed number of bins n, that we assume to be expressed as 2^m, for some integer number m. The space can thus be represented as a multidimensional $n \times d$ matrix, called *cumulative multi-dimensional grid for Q and CC in I* ($CMG_{Q,CC,I}$ or CMG for short when no ambiguity arises). Each cell of the CMG with index $(i_1, ..., i_d)$, $i_j \in \{0, ..., n-1\}$, corresponds to query $Q\langle v_1 + (a_j^{max} - v_1)/n * i_1, ..., v_d + (a_j^{max} - v_d)/n * i_d\rangle$. For the sake of simplicity, we denote such query with $Q^{(i_1,...,i_d)}$. Notice that $Q^{(0,...,0)} \equiv Q\langle\overline{v}\rangle$.

In order to detect a coverage-based rewriting for Q with respect to CC and I, we store in each cell $(i_1, ..., i_d)$ of the CMG the estimated cardinality $|Q^{(i_1,...,i_d)}(I)|$ (in the following denoted by $CMG((i_1, ..., i_d)).1$) and a list of estimated cardinalities, one for each value of the sensitive attribute for which a coverage constraint has been specified, namely $(|Q^{(i_1,...,i_d)} \downarrow_{s_1} (I)|, ..., |Q^{(i_1,...,i_d)} \downarrow_{s_t} (I)|)$ (denoted by $CMG((i_1, ..., i_d)).2$).

Given an index $\overline{a}$, we denote with $ur(\overline{a})$ (upper right elements of $\overline{a}$), the set $\{\overline{b} | \overline{b} \geq \overline{a}\}$, and with $ll(\overline{a})$ (lower left elements of $\overline{a}$), the set $\{\overline{b} | \overline{b} \leq \overline{a}\}$. Since we deal with sensitive selection monotone queries, it is easy to show that the following properties hold:

(a) $CMG(\overline{a}).1 \geq CMG(\overline{b}).1$ and $CMG(\overline{a}).2 \geq CMG(\overline{b}).2$ when $\overline{a} \geq \overline{b}$.
(b) $Q^{\overline{a}} \subseteq Q^{\overline{b}}$ for all $\overline{b} \in ur(\overline{a})$;
(c) $Q^{\overline{a}} \supseteq Q^{\overline{b}}$ for all $\overline{b} \in ll(\overline{a})$.

As an example, Figure 1(b) shows matrix CMG for the dataset in Figure 1(a), when only one coverage constraint is provided. For the sake of simplicity, each cell $\overline{a}$ reports only $CMG(\overline{a}).2$. Grey cells correspond to $ur((1,1))$ while light grey cells correspond to $ll((1,1))$. Cell $(1,1)$ is included in both $ur((1,1))$ and $ll((1,1))$.

In order to identify a coverage-based rewriting for Q, we propose three algorithms. The first is a baseline algorithm that identifies the solution by navigating the CMG, one cell after the other, at increasing distance from Q. The second algorithm is obtained from the first one by pruning the space, thus reducing the number of visited cells with a limited overhead; finally, the third algorithm visits the space, after pruning, by iteratively refining the size and the number of matrix cells to be visited, thus converging to the solution faster.

4.1 The Baseline Approach

Algorithm *CRBase* (Coverage-based Rewriting Baseline algorithm) visits the CMG according to a Space Filling Curve (SFC) in which cells are visited at increasing distance from $CMG(\overline{0})$ (see Figure 1(c)), by relying on a function $Next()$. During the visit, we look for the cell corresponding to the query with the minimum cardinality that satisfies coverage-based constraints CC. In particular, let $Space$ be the set containing all the CMG cell indexes. We initialize $\overline{min}$ with the index of the more distant query from Q, corresponding to cell $(n-1, ..., n-1)$. Notice that in $Q^{\overline{min}}$ each selection condition contains the maximum value for the selection attribute, thus all conditions are always satisfied in I and $Q^{\overline{min}}(I)$ returns all the input tuples. As a consequence, if $Q^{\overline{min}}(I)$ does not satisfy CC (line 7), no coverage rewriting can be generated and a void result is returned. Otherwise, we navigate the space one cell after the other according to the SFC (line 10). For each visited cell with index $\overline{a}$, coverage-based constraints CC are checked over $Q^{\overline{a}}(I)$ through predicate $Check(Q^{\overline{a}}(I), CC)$. If they are satisfied ($Check(Q^{\overline{a}}(I), CC)$ is true) (line 11) or they are not satisfied but the cardinality of $Q^{\overline{a}}(I)$ is higher than the current minimum cardinality (line 16), we remove $ur(\overline{a})$ from $Space$ (lines 12 and 17), since all cells in $ur(\overline{a})$ correspond to queries with higher cardinalities with respect to the currently identified minimum. When constraints are satisfied, the index of the currently minimum query is updated accordingly (line 13).

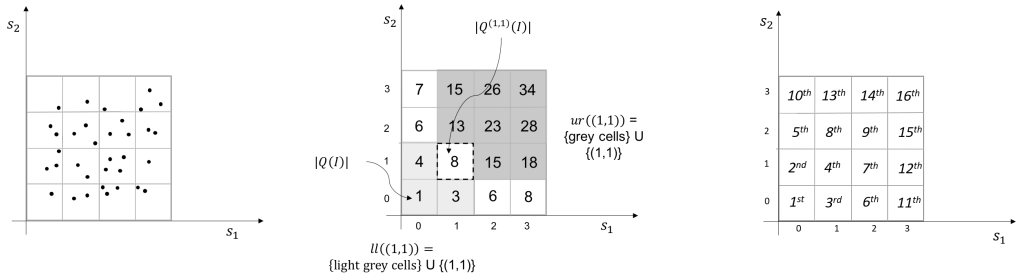
Algorithm 1 CRBase

```
1:  function CRBase(Q, CC, I)
2:  Input Q: a sensitive selection monotone query; CC: a set of coverage-based constraints; I: a database instance
3:  Output min: index of a coverage-based rewriting of Q with respect to CC and I
4:      Space = {all the cell indexes in the CMG}
5:      min = (n − 1, ...., n − 1)
6:      a = (0, ..., 0)
7:      if not Check(Q^min(I), CC) then return ⊥
8:      else
9:          while Space ≠ ∅ do
10:             a = Next(a, Space)
11:             if Check(Q^a(I), CC) then
12:                 Space = Space \ ur(a)
13:                 if |Q^a(I)| < |Q^min(I)| then min = a
14:                 end if
15:             else
16:                 if |Q^min(I)| < |Q^a(I)| then
17:                     Space = Space \ ur(a)
18:                 end if
19:             end if
20:         end while
21:         return min
22:     end if
23: end function
```

4.2 Adding Pruning

Algorithm *CRBase* can be optimized, leading to algorithm *CRBaseP* (*CRBase* with Pruning), by adding some pruning rules. Such rules generalize the pruning already applied by algorithm *CRBase* at lines 12 and 16 with the aim of further reducing the space to be visited and coping with the well known curse of dimensionality problem. Each pruning rule generates two sets of cells:

a) *Locking set* (LS). For each $\overline{a} \in LS$, query $Q^{\overline{a}}(I)$ satisfies coverage constraints. Based on properties (a) and (b), all the queries in $ur(\overline{a})$ can therefore be removed from $Space$ since they satisfy coverage constraints but they are not the closest to the initial query and their cardinality might not be the minimum one. We denote with LS^* the set $\bigcup_{\overline{a} \in LS} ur(\overline{a})$.
b) *No Solution set* (NS). For each $\overline{a} \in NS$, query $Q^{\overline{a}}(I)$ does not satisfy coverage constraints. Based on properties (a)



(a) Data distribution (b) $CMG_{Q,CC,I}$ (c) SFC

Figure 1: Data representation

and (c), this means that all the queries in $ll(\overline{a})$ can be removed from *Space* since they cannot satisfy coverage constraints. We denote with NS^* the set $\bigcup_{\overline{a} \in NS} ll(\overline{a})$.

We consider two pruning rules:

- *Diagonal pruning*. Using binary search, we look for the cell $(l, ..., l)$, $l \in \{0, ..., n-1\}$, closest to $(0, ..., 0)$ such that $Q^{(l,....,l)}(I)$ satisfies coverage constraints. We add $Q^{(l,....,l)}$ into *LS* and $Q^{(l-1,....,l-1)}$ into *NS*. The number of cardinality estimations performed by diagonal pruning is $O(\log n)$, where n is the number of bins.
- *Dimensional pruning*. For the sake of simplicity, we explain dimensional pruning in a 2-dimensional space. The pruning can however be easily extended to any d-dimensional matrix. Suppose that the horizontal axis corresponds to the selection attribute S_1 and the vertical one to the selection attribute S_2. For each position i, $i = 0, ..., n-1$ in the horizontal axis, we look for the first cell $(i, \overline{j})$, $\overline{j} \in \{0, ..., n-1\}$, that satisfies coverage constraints. It is easy to show that $(i, \overline{j}) \in LS$ and $(i, z) \in NS$, for all $z \leq \overline{j} - 1$. A similar computation is then applied to the vertical axis.
  Assuming to use binary search to locate the first cell satisfying the constraints for each direction, the number of cardinality estimations performed by dimensional pruning is $O(dn \log n)$, where n is the number of bins and d the number of axis, i.e., the number of selection conditions in the input query Q.

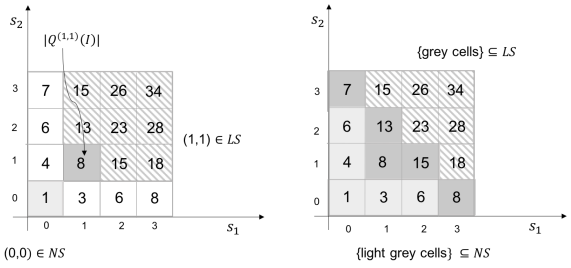
Figure 2 shows an example of the considered pruning rules, with a single coverage constraint set to $Q \downarrow_{s_i} \geq 7$.

Pruning rules can be easily integrated in *CRBase* algorithm as follows:

- sets *LS* and *NS* are computed before line 7 according to diagonal and dimensional pruning;
- at line 4, *Space* is initialized with all the cell indexes in the CMG minus those that do not satisfy coverage constraints, identified by the pruning phase. Thus, *Space* = { all the cell indexes in the CMG } \ ($LS^* \cup NS^*$).

4.3 Adding Iteration

CRBaseP can be further optimized by iteratively increasing the number of bins during the search up to a given maximum $n = 2^m$. The new algorithm is denoted by *CRBasePI* (*CRBaseP* with Iteration). Each iteration, by increasing the number of bins, increases the number of matrix cells to be visited for pruning and search. As a consequence, each iteration increases the precision by which we refine the query and we compute cardinalities. More precisely,



(a) Diagonal pruning (b) Dimensional pruning

Figure 2: Pruning in 2D with *Locking* and *NoSolution* sets

at each iteration $i \geq 1$, the number of bins is set to 2^i and Algorithm *CRBaseP* is used to prune the space, navigate it, and update index $\overline{min}$. Only when we reach the maximum precision i.e., when $i = m$ and the number of bins is equal to n, the computed $\overline{min}$ is returned as result.

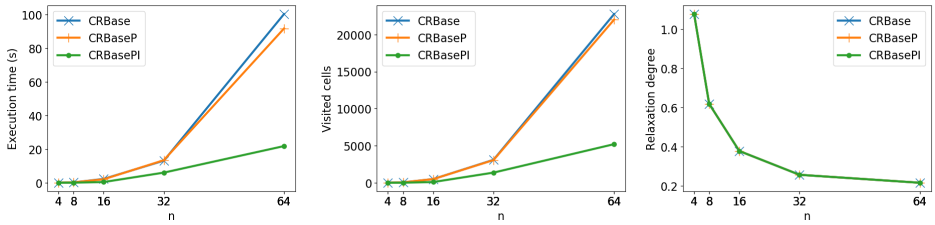
4.4 Cardinality Estimation

Our query refinement framework requires fast and accurate cardinality estimates for queries corresponding to visited cells of the CMG. These estimates could be obtained from the cardinality estimation component of the database system. However, such estimates are often incorrect, especially for queries with multiple joins and selection predicates. Since the accuracy of the proposed coverage-based rewriting depends on the used cardinality estimation approach, similarly to [12], we rely on sampling based estimators for cardinality estimation. In order to avoid sampling repeatedly for each visited cell, we compute an ϵ-approximation of the base tables and we rely on the approach in [13] for generating the sample of joined tables. We compute just one random sample for each query at hand (uniformly, independently, and without replacement). According to [13] such sample can be reused for all *structurally* equivalent queries (i.e., queries containing the same number of selection conditions and the same number of joins).

5 EXPERIMENTAL EVALUATION

5.1 Experimental Setup

All experiments were conducted on a PC with an Intel Core i5-8300H 2.30 GHz CPU and 16GB main memory, running Microsoft Windows 10. All programs were coded in Python 3.8.



(a) Execution time (b) Visited cells (c) Relaxation degree

Figure 3: Comparison of coverage rewriting algorithms

We considered two real datasets with different sizes, stored in PostgreSQL: *Adult dataset* [11] and *US Forbes Richest People dataset*.[1] The *Adult* dataset was originally extracted from the 1994 Census database and has been used in assessing fairness-aware data management tecniques (see, e.g., [16]). It contains 48842 individuals that are described by six numerical attributes (*age, fnlwgt, education-num, capital-gain, capital-loss, hours-per-week*) and eight categorical attributes (*workclass, education, marital-status, occupation, relationship, race, sex, native-country*). For our experiments, we use *sex* as sensitive attribute; it presents an unbalanced proportion of males (67%) and females (33%). For this dataset, we generated a random sample with 9604 tuples, guaranteeing query cardinality estimation with 0.1% error and 99% confidence. The sample is small enough to be stored in memory. The *US Forbes Richest People* dataset contains information about the richest people in US from 2016 (about 400 individuals), described by six numerical attributes (*worthchange, age, realtime-worth, realtimerank, timestamp, realtimeposition*) and nine categorical attributes (*name, lastname, uri, imageuri, source, industry, gender, headquarters, state*). For our experiments we use *gender* as sensitive attribute; it presents an unbalanced proportion of males (87%) and females (13%).

To measure the performance of the algorithms, we consider both the execution time and the number of query cardinality estimations, corresponding to the number of visited cells. To measure the accuracy of the algorithms, similarly to [12], given an initial query Q over a database instance I, rewritten into a new query Q', we compute the *relaxation degree* as the ratio

$$\frac{|Q'(I)| - |Q(I)|}{|Q(I)|}.$$

We then performed four groups of experiments aiming at: (i) comparing the three proposed algorithms; (ii) analyzing their performance and accuracy with respect to the number of bins and a single coverage constraint; (iii) analyzing their performance while changing the reference dataset; (iv) analyzing their performance and accuracy while changing the number of coverage constraints. Those preliminary experimental results refer to selection queries, we leave to future work the experimental analysis of join queries.

5.2 Experimental Results

Impact of pruning and iteration. In the first group of experiments, we compare the three proposed algorithms to understand the impact of pruning and iteration on performance and relaxation degree. To this aim, we consider a query $Q4$ with four selection conditions, selecting individuals who are over 20 years-old, have education number equal or greater than 13, work more than 20 hours per week and have a capital gain higher than 5500:

```
Q4: SELECT * FROM adult_data
WHERE age > 20 AND education_num ≥ 13
AND hours_per_week > 20 AND capital_gain > 5500
```

Consider the coverage constraint $Q \downarrow_{Female} \geq 250$. Query Q returns 1242 tuples (selectivity equal to 2.5%), out of which 200 are females (about 16%), thus the constraint is not satisfied.

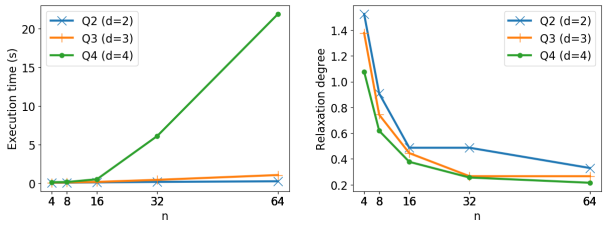
Figure 3 shows the execution time (in seconds) of the three algorithms by varying the number of bins n. As expected, the number of visited cells increases (and, as a consequence, performance decreases) while increasing n. *CRBasePI* exhibits the best performance, showing the benefits of relying on pruning and iteration. On the other hand, all techniques return the same coverage-based rewritten query, whose relaxation degree decreases while increasing the number of bins (see Figure 3 (c)). Similar results have been obtained with other queries. Therefore, in the following, we present experimental results only for *CRBasePI*. Additionally, since the number of visited cells and the execution time are correlated, in the following we report only execution time results.

Impact of the number of selection conditions. In the second group of experiments, we analyze in more details execution time and relaxation degree achieved with *CRBasePI* by varying the number of bins n and the threshold of the coverage constraint. The considered queries ($Q2$ and $Q3$ listed below and $Q4$ presented above) contain a different number of selection conditions but all them have a selectivity between 2-4% and the percentage of returned females is about 16% of the result. In this way, we can analyze *CRBasePI* performance independently from differences in query selectivity (similar results have been obtained with queries with higher selectivities).

```
Q2: SELECT * FROM adult_data
WHERE hours_per_week > 20 AND
capital_gain > 5500

Q3: SELECT * FROM adult_data
WHERE age > 20 AND hours_per_week > 20
AND capital_gain > 5500
```

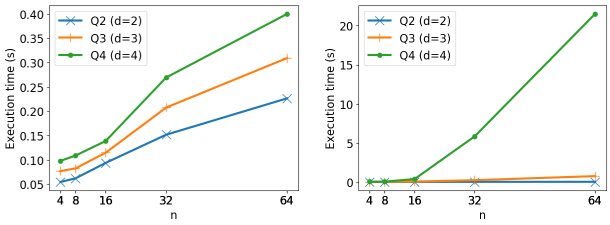
For each query, we consider a coverage constraint $Q \downarrow_{Female} \geq qf + 25\%$, where qf denotes the number of females returned by the initial query. From Figure 4 (a), we observe that, by varying the number of selection conditions, and therefore the dimensionality of the CMG, execution time rapidly increases starting from 32 bins. This is a well known problem, due to the curve of dimensionality. From Figure 4 (b), we notice that, starting from 16 bins, we get a relaxation degree lower than 0.4. The relaxation

[1] https://www.forbes.com/forbes-400/#ec679fa7e2ff



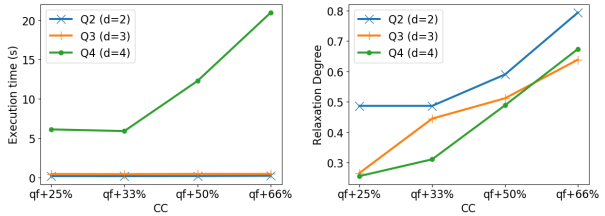
(a) Execution time (b) Relaxation degree

Figure 4: *CRBasePI* performance and accuracy, with respect to the number of bins



(a) Pruning time (b) Search time

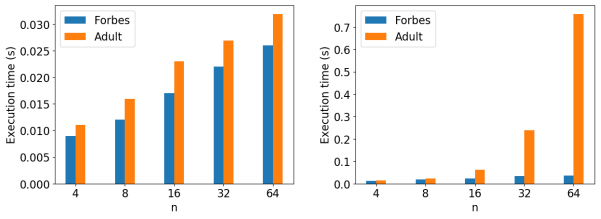
Figure 5: *CRBasePI* pruning and search time, with respect to the number of bins



(a) Execution time (b) Relaxation degree

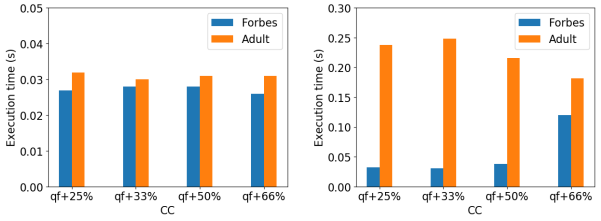
Figure 6: *CRBasePI* performance and accuracy, with respect to coverage threshold

degree is almost stable starting from 32 bins. By combining together results shown in Figures 4 (a) and (b), we can consider a number of bins equal to 32 a reasonable choice for getting a good compromise between performance and accuracy. From Figure 5(a), we notice that the pruning time is quite low (lower than half a second) even with $d = 4$, thus it does not deeply suffer from the curse of dimensionality while the search time does (Figure 5(b)). As a final experiment, we analyze *CRBasePI* performance by fixing the number of bins to 32 and varying the coverage constraint, setting the threshold to qf plus a given percentage (25%, 33%, 50%, 66%). From Figure 6(a), we observe that, by increasing the female coverage threshold, the execution time increases (quite rapidly for $d = 4$ due to the curve of dimensionality problem) since more cells are visited by the algorithm (the distance between the original query and the final one is higher). On the other hand, the relaxation degree almost linearly increases (see Figure 6(b)) with respect to the additional percentage of females required by the coverage constraint since, by increasing females, males increase as well.



(a) Execution time, $d = 2$ (b) Execution time, $d = 3$

Figure 7: Performance comparison between *Adult* and *Forbes* datasets, with respect to the number of bins



(a) Execution time, $d = 2$ (b) Execution time, $d = 3$

Figure 8: Performance comparison between *Adult* and *Forbes* datasets, with respect to coverage threshold

Impact of sampling. In our next experiment, we examine performance when varying the dataset size. To this aim, we consider the *Forbes* dataset. Since it is quite small (about 400 individuals), we do not generate any sample and we execute queries directly on the input dataset. Due to space constraints, we consider only queries with two and three selection predicates, keeping query selectivity in the range considered for the *Adult* dataset (about 3%):

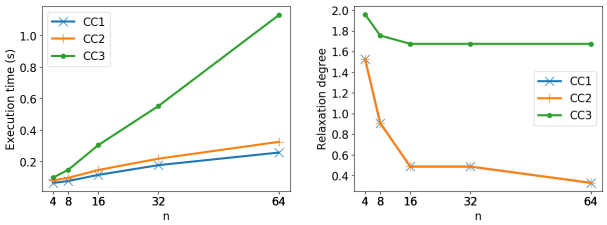
```
Q2_f: SELECT * FROM forbes
WHERE realtimeworth > 10000 AND age < 65

Q3_f: SELECT * FROM forbes
WHERE realtimeworth > 10000 AND age < 65
AND age > 45
```

Figure 7 compares execution time of corresponding queries executed over the *Adult* and *Forbes* datasets. As expected, the dataset size influences the cardinality query estimation time and, as a consequence, the total execution time. A similar result holds while increasing the female threshold (Figure 8). In both cases, differences in times are more evident for higher number of selection conditions ($d = 3$).

Impact of the number of coverage constraints. In order to analyze the impact of the number of coverage constraints, we consider query $Q2$ and the following sets of coverage constraints:

$CC_1 = Q2 \downarrow_{Female} \geq 456$
$CC_2 = Q2 \downarrow_{Female} \geq 456$ AND $Q2 \downarrow_{Male} \geq 1800$
$CC_3 = Q2 \downarrow_{Female} \geq 456$ AND $Q2 \downarrow_{Male} \geq 2400$

CC_1 is the constraint we considered in the previous experiments. CC_2 specifies an additional coverage constraint on males, which is however satisfied by the coverage-based rewriting generated when considering CC_1. Thus, we expect that the solution returned with CC_1 is the same than that returned with CC_2. On the other hand, with CC_3, we expect a different rewriting for increasing the number of males, as required by the constraint.



(a) Execution time (b) Relaxation degree

Figure 9: *CRBasePI* Performance and accuracy, with respect to different coverage constraints

Figure 9(a) shows execution time of query $Q2$ with respect to the considered coverage thresholds. When considering CC_2, the execution time slightly increases since, for each visited cell, one more query cardinality estimation is needed. However, since, by construction, when CC_1 is satisfied CC_2 is satisfied as well, the number of visited cells in the two cases coincides. On the other hand, when considering CC_3 more cells are visited and, as a consequence, the execution time increases as well. From Figure 9(b) we see that, as expected, the relaxation degrees for CC_1 and CC_2 coincide since in both cases the returned relaxed queries coincide. Relaxation is higher for CC_3 since a higher coverage threshold is considered.

6 CONCLUDING REMARKS

In this paper, we have presented a preliminary approach for coverage-based rewriting of SQL queries, suitable for addressing non-discrimination in early data processing stages. The proposed techniques revise and extend existing query refinement approaches, defined for addressing the empty or few answer problem, to take care of coverage issues, without the need of any user interaction. Experimental results show that the proposed approach is effective and efficient. Future works include: (i) further optimizations of the *CRBAsePI* algorithm to cope with the curve of dimensionality issue, by exploiting CMG sparsity and additional heuristics; (ii) considering more than one sensitive attribute and automating the identification of the sensitive attribute to analyze first; (iii) dealing with categorical attributes in the selection conditions and more complex fairness constraints; (iv) taking into account data summaries of the input dataset, thus making the rewriting reusable to datasets characterized by the same synopses; (v) data preparation is an iterative process, coverage-based rewriting can be considered as a new relational operation to take into account during the whole query processing steps, including query optimization.

REFERENCES

[1] Dolan Antenucci and Michael J. Cafarella. 2018. Constraint-based Explanation and Repair of Filter-Based Transformations. *PVLDB* 11, 9 (2018), 947–960.

[2] Abolfazl Asudeh, H. V. Jagadish, Gerome Miklau, and Julia Stoyanovich. 2018. On Obtaining Stable Rankings. *PVLDB* 12, 3 (2018), 237–250.

[3] Abolfazl Asudeh, H. V. Jagadish, Julia Stoyanovich, and Gautam Das. 2019. Designing Fair Ranking Schemes. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019.* 1259–1276.

[4] Abolfazl Asudeh, Zhongjun Jin, and H. V. Jagadish. 2019. Assessing and Remedying Coverage for a Given Dataset. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019.* 554–565.

[5] Piero A. Bonatti and Sabrina Kirrane. 2019. Big Data and Analytics in the Age of the GDPR. In *2019 IEEE International Congress on Big Data, BigData Congress 2019, Milan, Italy, July 8-13, 2019.* 7–16.

[6] Marina Drosou, H. V. Jagadish, Evaggelia Pitoura, and Julia Stoyanovich. 2017. Diversity in Big Data: A Review. *Big Data* 5, 2 (2017), 73–84.

[7] Serge Abiteboul et Al. 2018. Research Directions for Principles of Data Management (Dagstuhl Perspectives Workshop 16151). *Dagstuhl Manifestos* 7, 1 (2018), 1–29.

[8] Donatella Firmani, Letizia Tanca, and Riccardo Torlone. 2019. Data Processing: Reflections on Ethics. In *Proceedings of the 1st International Workshop on Processing Information Ethically co-located with 31st International Conference on Advanced Information Systems Engineering, PIE@CAiSE 2019, Rome, Italy, June 4, 2019.*

[9] Nodira Khoussainova, YongChul Kwon, Magdalena Balazinska, and Dan Suciu. 2010. SnipSuggest: Context-Aware Autocompletion for SQL. *PVLDB* 4, 1 (2010), 22–33.

[10] Nick Koudas, Chen Li, Anthony K. H. Tung, and Rares Vernica. 2006. Relaxing Join and Selection Queries. In *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006.* 199–210.

[11] Moshe Lichman. 2013. *UCI Machine Learning Repository.* Technical Report.

[12] Chaitanya Mishra and Nick Koudas. 2009. Interactive query refinement. In *EDBT 2009, 12th International Conference on Extending Database Technology, Saint Petersburg, Russia, March 24-26, 2009, Proceedings.* 862–873.

[13] Matteo Riondato, Mert Akdere, Ugur Çetintemel, Stanley B. Zdonik, and Eli Upfal. 2011. The VC-Dimension of SQL Queries and Selectivity Estimation through Sampling. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011, Proceedings, Part II.* 661–676.

[14] Babak Salimi, Corey Cole, Peter Li, Johannes Gehrke, and Dan Suciu. 2018. HypDB: A Demonstration of Detecting, Explaining and Resolving Bias in OLAP queries. *PVLDB* 11, 12 (2018), 2062–2065.

[15] Babak Salimi, Johannes Gehrke, and Dan Suciu. 2018. Bias in OLAP Queries: Detection, Explanation, and Removal. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018.* 1021–1035.

[16] Babak Salimi, Bill Howe, and Dan Suciu. 2019. Data Management for Causal Algorithmic Fairness. *IEEE Data Eng. Bull.* 42, 3 (2019), 24–35.

[17] Babak Salimi, Luke Rodriguez, Bill Howe, and Dan Suciu. 2019. Capuchin: Causal Database Repair for Algorithmic Fairness. *CoRR* abs/1902.08283 (2019).

[18] Babak Salimi, Luke Rodriguez, Bill Howe, and Dan Suciu. 2019. Interventional Fairness: Causal Database Repair for Algorithmic Fairness. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019.* 793–810.

[19] Julia Stoyanovich and Bill Howe. 2019. Nutritional Labels for Data and Models. *IEEE Data Eng. Bull.* 42, 3 (2019), 13–23.

[20] Julia Stoyanovich, Ke Yang, and H. V. Jagadish. 2018. Online Set Selection with Fairness and Diversity Constraints. In *Proceedings of the 21th International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, March 26-29, 2018.* 241–252.

[21] Chenkai Sun, Abolfazl Asudeh, H. V. Jagadish, Bill Howe, and Julia Stoyanovich. 2019. MithraLabel: Flexible Dataset Nutritional Labels for Responsible Data Science. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019.* 2893–2896.

[22] Ke Yang, Vasilis Gkatzelis, and Julia Stoyanovich. 2019. Balanced Ranking with Diversity Constraints. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019.* 6035–6042.

[23] Junjie Yao, Bin Cui, Liansheng Hua, and Yuxin Huang. 2012. Keyword Query Reformulation on Structured Data. In *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012.* 953–964.