Detecting Attribute Dependencies from Query Feedback

Peter J. Haas
IBM Almaden Research Center
San Jose, CA, USA
phaas@us.ibm.com

Fabian Hueske
University of Ulm
Ulm, Germany
fhueske@gmail.com

Volker Markl
IBM Almaden Research Center
San Jose, CA, USA
marklv@us.ibm.com

ABSTRACT

Real-world datasets exhibit a complex dependency structure among the data attributes. Learning this structure is a key task in automatic statistics configuration for query optimizers, as well as in data mining, metadata discovery, and system management. In this paper, we provide a new method for discovering dependent attribute pairs based on query feedback. Our approach avoids the problem of searching through a combinatorially large space of candidate attribute pairs, automatically focusing system resources on those pairs of demonstrable interest to users. Unlike previous methods, our technique combines all of the pertinent feedback for a specified pair of attributes in a principled and robust manner, while being simple and fast enough to be incorporated into current commercial products. The method is similar in spirit to the CORDS algorithm, which proactively collects frequencies of data values and computes a chi-squared statistic from the resulting contingency table. In the reactive query-feedback setting, many entries of the contingency table are missing, and a key contribution of this paper is a variant of classical chi-squared theory that handles this situation. Because we typically discover a large number of dependent attribute pairs, we provide novel methods for ranking the pairs based on degree of dependency. Such ranking information, e.g., enables a database system to avoid exceeding the space budget for the system catalog by storing only the currently most important multivariate statistics. Experiments indicate that our dependency rankings are stable even in the presence of relatively few feedback records.

1. INTRODUCTION

Real-world datasets typically exhibit strong and complex dependencies between data attributes. Discovering this dependency structure is important in a variety of settings. For query optimization in a relational database management system (DBMS), discovery of highly dependent columns is crucial in order to avoid erroneous independence assumptions during selectivity estimation. Such assumptions can lead to selectivity estimates that are too low by orders of magnitude, resulting in highly suboptimal query execution plans. Commercial database systems currently use knowledge about dependent attributes for *statistics configuration*, that is, for determining which sets of multivariate statistics to maintain in the system catalog. Although automatic statistics configuration is the primary motivation behind our research, the potential applications of methods for dependence discovery are much broader. In data integration problems, discovering the dependency structure is a key step when developing mappings between different schemas. As another example, system monitoring products often maintain a database of joint observations, over time, of various system metrics, for purposes of post-mortem problem analysis. Discovery of highly dependent metrics can play a key role in both root-cause analysis of system failures, as well as in the design of system monitors [14].

One approach to testing for dependency between attributes A and B is to proactively examine attribute-value pairs (a,b) from the dataset, and use the observed frequencies of the pairs to test for a dependency. The CORDS method for dependency detection [16] follows this approach, using essentially a chi-squared test to decide whether a dependency exists. CORDS uses random sampling techniques to make the method scale to large tables, and heuristic pruning methods to limit the number of attribute pairs considered. Proactive methods such as CORDS, although very useful, do not by themselves provide a complete solution to the dependency-detection problem, especially in very large databases. Specifically, the space of candidate attribute-pairs can be combinatorially huge, overwhelming any pruning strategy, so that this space cannot be thoroughly explored.

Reactive methods—in which detection of dependencies is based on query feedback—complement proactive methods by focusing system resources on attributes that are of demonstrable interest to the user, without the need for explicit workload profiling. Reactive methods have the additional desirable property that dependence testing is based on exact, rather than sampled, frequencies. As discussed in [23], the feedback information can be collected with minimal overhead; even if some dependent attributes escape detection, it would certainly be wasteful not to opportunistically exploit this readily available information for purposes of statistics configuration. For these reasons, products such as DB2 have begun to exploit feedback information to automate statistics collection [1].

An effective solution to the dependency-detection problem can be obtained by combining the reactive and proactive approaches. The idea is to use a reactive feedback-based method for reasons of efficiency, and then supplement this method with a proactive method such as CORDS, run either periodically or in throttled background mode [1], to help provide robust selectivity estimates during the reactive method's learning period, or when the workload suddenly changes. The focus of this paper is on providing the first ingredient of this hybrid approach: an effective and practical method for feedback-based dependency detection.

There is much room for improvement in the simple "correlation analyzer" (CA) method, described in [1], that has hitherto been used by DB2 for feedback-based dependency detection. The CA method examines each available feedback observation for a given pair of attributes. For example, a "feedback observation" for the attribute pair (`Color`, `Year`) might comprise the number of tuples that satisfy each of the predicates `(Color='red')`, `(Year='2005')`, and `(Color='red' AND Year = '2005')`. If the relationship between these three cardinalities deviates enough from what would be expected under the independence assumption, then this feedback observation would be considered as providing evidence for a dependency between `Year` and `Color`. Other feedback observations would correspond to other possible specific (`Color`, `Year`) combinations; if "enough" indicative feedback observations are observed, then a dependency is declared. Although this method is simple and fast enough to be used in a commercial product, it suffers from several drawbacks. One apparent difficulty with the CA method is that the individual cardinalities on `Year` and `Color` may not both be available. The solution to this problem is not described in [1]; we present a solution in Section 5. Another issue is that CA method for determining the critical degree of "deviation from independence" is completely ad hoc, as is the number of non-independent feedback observations required to declare a dependency between an attribute pair. The resulting behavior of the method can be unstable, with the outcome being unduly influenced by one or two "outlier" feedback records. In contrast, our new method considers all of the feedback records pertaining to `Year` and `Color` in a principled manner.

Another approach to detecting dependencies from feedback, lying at the other end of the complexity spectrum, is the SASH algorithm [18]. The "restructuring phase" of SASH uses feedback to build a graphical statistical model (a "junction tree") that represents the dependency relationships. This graphical model serves to decompose the set of all attributes into mutually disjoint partitions. A multivariate, feedback-based histogram is maintained on the attributes within each partition, and an independence assumption is used to relate attributes that lie in different partitions. The structure of the graphical model is chosen so as to minimize a scoring function that measures the discrepancy between observed and estimated selectivities over a query workload; the algorithm performs essentially a steepest descent search through the space of candidate model structures. Although SASH can be very accurate, and allows for a very flexible representation of dependency structure, the scoring operations required when searching for the best model can be prohibitively expensive. The cost of scoring is exacerbated even more when more sophisticated histograms, such as the self-tuning histograms in [22], are used. The scoring costs—together with the tight coupling between determination of the best graph model and selection of the number of buckets in the histograms—make this approach difficult to incorporate into commercial DBMSs. Moreover, in some applications, such as system monitoring, it suffices to identify dependent attributes, and the additional cost and complexity of maintaining histograms (unavoidable in SASH), is not needed by the user.

This paper proposes a new approach to feedback-based dependency detection that overcomes the deficiencies of the CA approach while being simple and fast enough to be used in commercial products. Our method is intermediate in complexity between CA and SASH. We roughly follow the approach in CORDS, in that, for each candidate attribute pair, we obtain a "contingency table" of joint and marginal frequencies, and use this table to test for a dependency. The key technical challenge is that, because our contingency table is based on feedback, there are many missing entries, e.g., values of `Year` and `Color` that the queries did not reference. This means that we cannot, as in [16], simply appeal to the classical statistical theory surrounding the chi-squared test. A major contribution of this paper is a new variant of chi-squared theory (see Theorem 1 below) that is tailored to the feedback setting. The resulting algorithm is well suited to the task of automated statistics configuration. Because the method is not tied to any particular type of multivariate statistic, it can easily be coded into an existing DBMS, and our experiments indicate that the method is fast enough to yield acceptable performance for off-line analysis of the feedback warehouse.

The technique can also be used in other situations where dependency structure needs to be discovered, such as the mining and monitoring applications mentioned above. Indeed, if the feedback observations do not need to be "completed" (in the sense of Section 5 below), then our method does not require that the feedback warehouse contain optimizer estimates of selectivities, but merely the observed selectivity values. This relaxed requirement contrasts with feedback-based methods such as LEO [23] and SASH [18], which are specifically targeted at query optimization.

Because dependence between attributes is so prevalent in the real world, any dependency-detection method is likely to discover a large number of dependent attribute pairs. Thus the real task is often to rank discovered pairs in decreasing order of dependency. In the context of statistics configuration, we can then maintain multivariate statistics on the "most dependent" attributes, subject to memory and processing constraints. The CA method uses an ad hoc dependency measure comprising the sum of errors over all feedback observations. As shown experimentally in Section 9, this approach can lead to erratic dependency rankings that are highly sensitive to the number and nature of the available feedback records. For CORDS, the test statistic for the dependency analysis can easily be normalized into a classical dependency measure called the "mean squared contingency" [7] that can be used for ranking. In our setting of incomplete contingency tables, normalization is more complicated. We provide a theoretically rigorous normalization method, as well a number of ad hoc normalizations that are easier to implement and numerically more stable. Fortunately, our experiments indicate that all of the resulting rankings are fairly consistent, so that, among a collection of "reasonable" normalizations, the particular choice of normalization is not all that critical. Our experiments also show that the resulting rankings are extremely stable with respect to the number and processing order of feedback observations. Finally, we also show that the new ranking measures lead to a promising method of robustly pruning away independent or almost-independent attribute pairs, so that attention can be focused on the detailed analysis of the remaining pairs.

The rest of the paper is organized as follows. After reviewing related work in Section 2, we give some background and introduce our technical notation in Section 3. In Section 4 we present the new methodology. Sections 5 and 6 address issues related to missing and inconsistent feedback. The dependency detection algorithm is given in Section 7, and our ranking methods are given in Section 8. Section 9 describes the results of an empirical study, and Section 10 contains our conclusions. An appendix contains the proof of our key statistical result.

2. RELATED WORK

The problem of discovering dependency structure has been studied by the database community in the context of automatic statistics configuration and maintenance. Proactive approaches for determining sets of attributes on which to maintain multivariate statistics include CORDS [16]—as discussed in the introduction—as well as the methods in [9, 10]. The latter methods build statistical models that represent the dependency structure, and are based on a full off-line scan of the data (unlike CORDS, which is based on sampling). The methods in [9, 10] are primarily oriented toward a specific type of multivariate statistic, namely histograms; CORDS is agnostic about the specific statistic used, as is our current approach.[1] Proactive algorithms have also been proposed for specific kinds of dependencies, such as exact and approximate functional dependencies [15] and fuzzy algebraic constraints [13]. The data mining community has also looked at methods for discovering correlated attributes, typically based on a full scan of the data; see, for example, [6, 25].

As discussed in the introduction, the two main methods for discovering dependencies using feedback are the CA method in [1] and the SASH method [18]. SASH, as with the proactive methods in [9, 10], is oriented towards histograms, although the framework presented in [18] could potentially encompass other multivariate statistics. There has also been recent work on maintaining multivariate histograms based on query feedback [4, 22]; these methods require, however, that the sets of attributes on which to maintain the histograms be chosen a priori.

An intermediate approach along the proactive/reactive spectrum is described in [3, 5]. The given methods determine the set of columns on which to create statistics by analyzing the sensitivity of query plan selection to certain statistical parameters. The input to these methods comprises essentially a specified query workload and a set of catalog statistics that have been gathered previously. Neither actual nor observed selectivities are used in the calculations.

Recent work in system monitoring [14] indicates another potential application of the results in this paper. The idea is to mine a historical database of tuples of the form $\langle t, m_1(t), m_2(t), \ldots, m_k(t)\rangle$, where t is a measurement time and $m_i(t)$ is the value of the ith system metric at time t. Here $m_i(t)$ might be the utilization of a disk or the number of active threads in a software application. Many system anomalies manifest themselves as departures from the usual dependency structure, e.g., when $m_1 =$ "packets sent by process 1," $m_2 =$ "packets received by process 2," and there is a problem with the network connection between the two processes. Such anomalies can only be detected by setting up a monitor that jointly tracks the two metric streams. The work in [14] uses a proactive approach to identify candidate sets of streams for joint monitoring; the current technology can be used to identify important sets of metric streams to monitor, based on analytical queries that users execute over the historical database during the course of problem analysis. Such a reactive approach might help address the scalability challenges inherent in the proactive approach.

3. PRELIMINARIES

In this section, we motivate our results by outlining the statistics-configuration setting in which our problem arises, give a precise problem definition, outline our assumptions, and describe both the CA method and CORDS in more detail.

3.1 Problem Context: Statistics Configuration

As is well known, even the best query optimizers can produce selectivity estimates that are off by orders of magnitude, often due to erroneous assumptions of independence between attributes. The LEO (LEarning Optimizer) approach [23] was originally proposed

[1]We note that there are many types of multivariate statistics besides histograms, ranging from simple "column-group statistics" as described in [16] to more sophisticated statistics such as wavelets [24] and sketches [11].

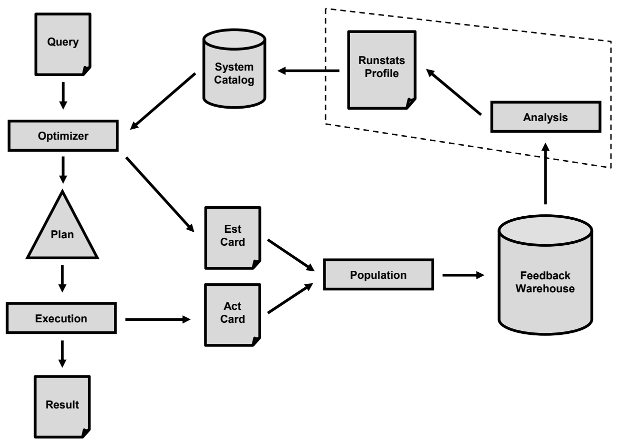

Figure 1: System architecture for statistics configuration

to combat poor optimizer estimates by exploiting query feedback. An evolved version of this methodology has been incorporated into DB2 LUW, and is outlined in [1]. A simplified[2] version of the system architecture is shown in Figure 1. As queries are processed, the system records the optimizer's cardinality estimates, as well as the actual observed cardinalities, and populates a *query feedback warehouse* with these (estimate, actual) pairs. The warehouse is periodically mined (off line) using the CA analysis technique mentioned previously. Based on the analysis, the RUNSTATS statistics-collection utility may be instructed to collect multivariate "column group" statistics; the instructions are transmitted to RUNSTATS via its "profile" (i.e., its configuration file). The work described in this paper was originally motivated by the need to improve the mining portion of the feedback loop (indicated by a dashed line in Figure 1), specifically, the analysis component.

3.2 Problem Definition

Our goal is to detect dependencies between attributes. By "dependency," we mean a departure from the "independence" condition

$$f_{\alpha\beta} = f_{\alpha\cdot} f_{\cdot\beta} \tag{1}$$

for $\alpha \in D_A$ and $\beta \in D_B$. Here D_X denotes the set of the distinct values for attribute X that appear in the dataset, $f_{\alpha\beta}$ denotes the fraction of elements t in the dataset such that $t.A = \alpha$ and $t.B = \beta$, and the quantities $f_{\alpha\cdot}$ and $f_{\cdot\beta}$ are marginal sums: $f_{\alpha\cdot} = \sum_{\beta\in D_B} f_{\alpha\beta}$ and $f_{\cdot\beta} = \sum_{\alpha\in D_A} f_{\alpha\beta}$. Thus $f_{\alpha\cdot}$ is the fraction of elements t in the dataset such that $t.A = \alpha$, and similarly for $f_{\cdot\beta}$. Note that the relation in (1) is, strictly speaking, simply an assertion about a factorization property of the frequency distribution of values in the dataset; however, if an element X is selected randomly and uniformly and (1) holds, then $P\{X.A = \alpha \text{ and } X.B = \beta\} = P\{X.A = \alpha\}P\{X.B = \beta\}$, so that (1) can be interpreted as implying statistical independence in an appropriate sense. Moreover, if one views the data in the dataset as having been generated by a probabilistic mechanism that is characterized by independence of attributes A and B, then one would expect (1) to hold, at least ap-

[2]The full system uses additional techniques, such as update-insert-delete counters and goodness-of-fit tests, to determine when to update statistics, and how to prioritize various types of updates. The results described in this paper comprise "recommendations" to the prioritization scheme; see [1] for details.

proximately. As discussed below, we use this latter "superpopulation" point of view to help us characterize dependency in a practical sense.

Our method takes as input a set of observations $\mathscr{O} = \{\mathscr{O}_1, \mathscr{O}_2, \ldots, \mathscr{O}_n\}$, where $1 \le n < |D_A \times D_B|$ and each observation $\mathscr{O}_i$ concerns a conjunctive predicate of the form "$A = \alpha_i$ AND $B = \beta_i$" with $\alpha_i \in D_A$ and $\beta_i \in D_B$. Each sub-predicate that appears in the conjunctive predicate, e.g., "$A = \alpha_i$" in the above example, is called a *simple* predicate. We assume that $(\alpha_i, \beta_i) \ne (\alpha_j, \beta_j)$ for $j \ne i$. Each observation $\mathscr{O}_i$ is a set having one of the forms (i) $\mathscr{O}_i = \{f_{\alpha_i\beta_i}, f_{\alpha_i\cdot}, f_{\cdot\beta_i}\}$, (ii) $\mathscr{O}_i = \{f_{\alpha_i\beta_i}, f_{\alpha_i\cdot}\}$, (iii) $\mathscr{O}_i = \{f_{\alpha_i\beta_i}, f_{\cdot\beta_i}\}$, or (iv) $\mathscr{O}_i = \{f_{\alpha_i\beta_i}\}$, depending on the feedback that is available. In the context of statistics configuration, the "dataset" may comprise a base table or, more generally, a view computed from one or more base tables,[3] and the observations in $\mathscr{O}$ are derived from the query feedback records (QFRs) in the feedback warehouse. As discussed in Section 3.1, a QFR contains the observed cardinality for a (simple or conjunctive) predicate, together with the optimizer's estimated cardinality. Whereas the quantity $f_{\alpha_i\beta_i}$ is guaranteed to be available from the feedback warehouse—otherwise we would not be using this observation for the dependency analysis—the quantities $f_{\alpha_i\cdot}$ and $f_{\cdot\beta_i}$ may or may not be available, depending on the query plans that were chosen by the optimizer. Suppose, for example, that attributes A and B correspond to columns in a table T. If the simple predicate p_2: "$T.B = \beta_i$" was applied over the result of the simple predicate p_1: "$T.A = \alpha_i$" in a given query, then only $f_{\alpha_i\cdot}$ and $f_{\alpha_i\beta_i}$ would be available. If, on the other hand, (A, B) is a prefix of an index on T and both p_1 and p_2 were applied simultaneously, then only $f_{\alpha_i\beta_i}$ would be available. As a final example, $f_{\alpha_i\cdot}$ and $f_{\alpha_i\beta_i}$ might have been obtained as described above from an execution of some query and $f_{\cdot\beta_i}$ might also be available based on the execution of a different query in the workload.

We assume throughout that the dataset size M is known, or at least can be estimated to within an error that is small relative to other errors. In the DBMS setting, M typically corresponds to the cardinality of a table T, and might be obtained from the system catalog; the error can be assumed small because any reasonable feedback-based statistics-configuration system would ensure that the basic statistics for T are up to date before proceeding to a dependency analysis. We use our knowledge of M to estimate relative frequencies such as $f_{\alpha\beta}$, $f_{\alpha\cdot}$, and $f_{\cdot\beta}$ from the absolute frequencies (i.e., cardinalities) in the QFRs.

For the purpose of "completing" feedback observations as described in Section 5 below, we also assume that estimates of marginal relative frequencies are available, i.e., estimates of the form $\hat{f}_{\alpha_i\cdot}$ and $\hat{f}_{\cdot\beta_i}$; in the setting of statistics configuration, these quantities correspond to the optimizer's selectivity estimates for simple predicates. Finally, we assume knowledge of an upper bound δ on the relative error of these estimates. In particular, we have

$$\left|\frac{\hat{f}_{\alpha\cdot} - f_{\alpha\cdot}}{f_{\alpha\cdot}}\right| \le \delta \qquad \text{and} \qquad \left|\frac{\hat{f}_{\cdot\beta} - f_{\cdot\beta}}{f_{\cdot\beta}}\right| \le \delta$$

for all $\alpha \in D_A$ and $\beta \in D_B$. For practical purposes, all that is needed is a reasonable approximate bound; it even suffices that the bound holds merely with high probability. In the setting of statistics configuration, a value of δ can be obtained by scanning all of the records in the feedback warehouse corresponding to simple predicates and computing relative errors by comparing actual to estimated selectivities. We then compute δ by aggregating these observed relative errors, e.g., we take δ as the mean, median, or, more conservatively, an upper percentile of the errors. In this setting, δ is typically small: detection of large selectivity errors on simple predicates will automatically trigger statistics collection, often with automatic adjustments to parameters such as the number of histogram buckets used by the optimizer to estimate selectivities.

[3]Note that we analyze dependency for a pair of attributes with respect to the table or view in which they jointly appear. In principle, for a given attribute pair, we could obtain different results for different views.

3.3 The CA Method

Next, we describe the CA method [1] for detecting a dependency between attributes A and B. Each feedback observation pertaining to these attributes is examined. For a particular observation $\mathscr{O}_i = \{f_{\alpha_i\beta_i}, f_{\alpha_i\cdot}, f_{\cdot\beta_i}\}$, an observational dependency is declared if the relationship

$$1 - \Theta \le \frac{f_{\alpha_i\beta_i}}{f_{\alpha_i\cdot} f_{\cdot\beta_i}} \le 1 + \Theta$$

fails to hold, where Θ is "a small pre-specified" positive parameter less than 1. If a pair of attributes has at least two observational dependencies, then the attributes are considered dependent. The number of observational dependencies is used as a measure of the degree of dependency when ranking attribute pairs. The description in [1] does not specify what to do if $f_{\alpha_i\cdot}$ or $f_{\cdot\beta_i}$ is unavailable; we give a solution to this problem in Section 5. As discussed in Section 1, the value of Θ is completely ad hoc. Moreover, the CA method does not combine all feedback observations pertinent to attributes A and B in a principled manner, which can lead to unstable behavior of the detection algorithm.

The CA method computes the dependency measure for the attribute pair (A, B) essentially as $\sum_i |f_{\alpha_i\beta_i} - f_{\alpha_i\cdot} f_{\cdot\beta_i}|$. As shown in Section 9, this measure can lead to unstable, erratic rankings.

3.4 CORDS

We conclude Section 3 by briefly describing the dependency-analysis method used by CORDS, since this method is closely related to our new technique. In the proactive setting of CORDS, we essentially have available (an estimate of) the marginal and joint frequencies corresponding to every attribute value in D_A and D_B. An appropriate test statistic for dependence is the chi-squared statistic, defined for M observations by

$$\chi^2 = M \sum_{\alpha \in D_A} \sum_{\beta \in D_B} \frac{(f_{\alpha\beta} - f_{\alpha\cdot} f_{\cdot\beta})^2}{f_{\alpha\cdot} f_{\cdot\beta}}.$$

The dependence test rests on the fact that standard result [7] that χ^2 has, asymptotically, a chi-squared distribution with r degrees of freedom as M becomes large, where $r = (|D_A| - 1)(|D_B| - 1)$. We can normalize χ^2 to obtain a measure of the "distance from independence." This distance is called the *mean-square contingency distance* (MSCD), and is defined in [7] as $\phi^2 = \chi^2 / \big(M(d-1)\big)$, where $d = \min(|D_A|, |D_B|)$. It can be shown that $0 \le \phi^2 \le 1$. Moreover, $\phi^2 = 0$ if and only if (1) holds, and $\phi^2 = 1$ if and only if the value of one of the attributes is completely determined by the value of the other, i.e., a functional dependency exists between the attributes. The MSCD can be used to rank attribute pairs by their degree of dependency.

A note on terminology: in the CORDS scenario, we have a complete *contingency table*, where this statistical terminology refers to a display of the (absolute) attribute-value frequencies in a $D_A \times D_B$ array, with table entries of the form $n_{\alpha\beta} = M f_{\alpha\beta}$; moreover, marginal absolute frequencies of the form $n_{\alpha\cdot} = M f_{\alpha\cdot}$ and $n_{\cdot\beta} = M f_{\cdot\beta}$ are displayed as row and column totals, respectively.

A key technical challenge in our reactive setting is that we do not have frequency information for all attribute-value pairs $(\alpha, \beta) \in$

$D_A \times D_B$, as required for chi-squared theory. We only have information for those pairs $\{(\alpha_i, \beta_i): 1 \le i \le n\}$ that correspond to the available feedback observations; i.e., we have an incomplete contingency table. In the following section, we modify classical chi-squared theory to obtain an analog (Theorem 1) to the classical limit theorem alluded to above. The new limit theorem leads to a more principled test for dependence than the CA method. We discuss the related issue of how to rank attribute pairs in Section 8.

4. DEPENDENCY DETECTION

We first assume that each observation $\mathscr{O}_i$ is of the form (i) given in Section 3.2, so that the relative frequencies $f_{\alpha_i\beta_i}$, $f_{\alpha_i\cdot}$, and $f_{\cdot\beta_i}$ are all available; we discuss the issue of incomplete feedback observations in the next section. Our approach is to compute a statistic from $\{\mathscr{O}_1, \mathscr{O}_2, \dots, \mathscr{O}_n\}$ that measures the departure from the independence assumption. The statistic is of the form

$$H_M = M\mathbf{x}^t\mathbf{Q}\mathbf{x}, \tag{2}$$

where the superscript "t" denotes matrix transpose, the vector[4] $\mathbf{x} = (x_1, x_2, \dots, x_n)$ is defined by

$$x_i = \frac{f_{\alpha_i\beta_i} - f_{\alpha_i\cdot}f_{\cdot\beta_i}}{f_{\alpha_i\cdot}f_{\cdot\beta_i}} \tag{3}$$

for $1 \le i \le n$ (where we take $0/0 = 1$), and the matrix $\mathbf{Q}$ is specified as follows. First define an $n \times n$ matrix $\mathbf{\Sigma} = \|\sigma_{ij}\|$ by setting

$$\sigma_{ij} = \begin{cases} \frac{(1-f_{\alpha_i\cdot})(1-f_{\cdot\beta_i})}{f_{\alpha_i\cdot}f_{\cdot\beta_i}} & \text{if } i = j; \\ -\frac{1-f_{\alpha_i\cdot}}{f_{\alpha_i\cdot}} & \text{if } i \ne j,\ \alpha_i = \alpha_j, \text{ and } \beta_i \ne \beta_j; \\ -\frac{1-f_{\cdot\beta_i}}{f_{\cdot\beta_i}} & \text{if } i \ne j,\ \alpha_i \ne \alpha_j, \text{ and } \beta_i = \beta_j; \\ 1 & \text{if } i \ne j,\ \alpha_i \ne \alpha_j, \text{ and } \beta_i \ne \beta_j. \end{cases} \tag{4}$$

for $1 \le i, j \le n$. Clearly $\mathbf{\Sigma}$ is symmetric, and there exists a real orthogonal matrix $\mathbf{G}$ and a diagonal matrix $\mathbf{D} = \mathrm{diag}(d_1, d_2, \dots, d_n)$ such that $\mathbf{\Sigma} = \mathbf{G}^t\mathbf{D}\mathbf{G}$ or, equivalently, $\mathbf{G}\mathbf{\Sigma}\mathbf{G}^t = \mathbf{D}$; see [12, p. 410]. This representation of $\mathbf{\Sigma}$ is called the *(symmetric) Schur decomposition*; in its canonical form, the rows of $\mathbf{G}$ are the eigenvectors of $\mathbf{\Sigma}$ and the diagonal entries of $\mathbf{D}$ are the eigenvalues. We show in the Appendix that each d_i is nonnegative. Denote by $r = r(\mathbf{Q})$ the number of strictly positive diagonal entries of $\mathbf{D}$. Set $\tilde{\mathbf{D}} = \mathrm{diag}(\tilde{d}_1, \tilde{d}_2, \dots, \tilde{d}_n)$, where

$$\tilde{d}_i = \begin{cases} 1/d_i & \text{if } d_i \ne 0; \\ 0 & \text{if } d_i = 0 \end{cases} \tag{5}$$

for $1 \le i \le n$.[5] We then define the matrix $\mathbf{Q}$ in (2) as $\mathbf{Q} = \mathbf{G}^t\tilde{\mathbf{D}}\mathbf{G}$.

Observe that $H_M \ge 0$, and it can be shown that $H_M = 0$ if and only if (1) holds. Thus a naive criterion for dependency would be that $H_M > 0$. Such a criterion, however, is too stringent. Even when it is intuitively clear that two attributes are statistically independent of each other, the formal independence assumption in (1) seldom holds exactly, due both to data errors and to idiosyncrasies in the mechanism by which the data is collected. Thus we consider attributes A and B to be dependent only if $H_M > \theta$, where θ is a positive threshold parameter.

We can derive a reasonable value of θ using a "superpopulation" model of the kind commonly used in the theory of survey sampling [21, p. 22]. For simplicity, suppose that A and B are the only attributes in the dataset. The idea is to view the elements $t_1, t_2, \dots, t_M$ in the dataset as independent and identically distributed samples from a joint probability distribution p on (D_A, D_B), where $p_{\alpha\beta} = P\{t.A = \alpha \text{ and } t.B = \beta\} = f_{\alpha\cdot}f_{\cdot\beta}$ for each $\alpha \in D_A$ and $\beta \in D_B$; here $f_{\alpha\cdot}, f_{\cdot\beta}$ are defined as before. In other words, we suppose that $t.A$ and $t.B$ are "truly" statistically independent with marginal probabilities equal to the observed marginal probabilities $f_{\alpha\cdot}$ and $f_{\cdot\beta}$. Under this model, the statistic H_M is a random variable; we choose θ such that, under our "null hypothesis" of independence, the probability that H_M exceeds θ is small. To this end, we can use the large-sample result given in Theorem 1 below. In the theorem, χ^2_d denotes the cumulative distribution function for a χ^2 random variable with d degrees of freedom.

THEOREM 1. *Under the superpopulation model, and setting $H_M = M\mathbf{x}^t\mathbf{Q}\mathbf{x}$, we have $P\{H_M \le x\} \to \chi^2_r(x)$ as $M \to \infty$ for all $x \ge 0$, where $r = r(\mathbf{Q})$.*

Theorem 1 asserts that if the number of rows M in the dataset is large, then H_M has approximately a χ^2_r distribution; see the Appendix for a proof. Using the theorem, we can choose θ as the $(1-p)$-quantile of χ^2_r to ensure that $P\{H_M > \theta\} \approx p$. For example, if $r = 10$ and we want to ensure that $P\{H_M > \theta\} \approx 0.005$, then we choose $\theta = 25.2$.

Although the foregoing procedure superficially resembles the classical χ^2 test for independence, there is a key difference. The classical χ^2 statistic corresponds to a sum over all $|D_A \times D_B|$ possible attribute-value combinations, whereas our statistic is a sum over the d arbitrary combinations in the feedback, and often $d \ll |D_A \times D_B|$.

We note that our new approach is not directly driven by specific queries in the workload as with [3, 5] or the SASH algorithm, but rather by observed selectivities induced by the query processing. Consequently, in the query-optimization setting, our technique is not quite as efficient in focusing system resources on user queries, but is more robust in the presence of previously-unseen queries.

5. INCOMPLETE OBSERVATIONS

We have assumed up until now that each observation $\mathscr{O}_i$ is as of the form (i) given in Section 3.2, so that the joint selectivity $f_{\alpha_i\beta_i}$ and the two marginal selectivities $f_{\alpha_i\cdot}$ and $f_{\cdot\beta_i}$ are available. Often, however, at least one marginal selectivity is missing, so that we are dealing with an observation of form (ii), (iii), or (iv). In this section, we describe a simple method for completing a feedback observation by estimating the missing selectivities.

In practice—and especially in the context of statistics configuration—we need not be concerned with observations of the form (iv), in which only the joint selectivity is available. As explained previously, such observations typically result from the application of a multi-column index. Under this scenario, the presence of a dependency can be determined from the index itself, with no feedback analysis required.

We therefore focus on observations of the form (ii) or (iii), in which the joint selectivity and one of the two marginal selectivities are available. Our approach is to estimate the missing marginal selectivity for each incomplete observation and then apply the methodology for an observation of the form (i). An overall goal in estimating a missing marginal selectivity is to be conservative, that is, to be reluctant to declare the presence of a dependency, because each such declaration leads to increased processing and storage requirements. We therefore choose the selectivity value that is "most consistent" with the independence assumption, subject to our knowledge about the range of possible values for this selectivity.

[4]All vectors are assumed to be column vectors.

[5]In practice, we would fix a small number ε—roughly equal to the machine precision—and set $\tilde{d}_i = 1/d_i$ if $d_i > \varepsilon$ and $\tilde{d}_i = 0$ otherwise.

Specifically, consider the case in which we know $f_{\alpha_i\beta_i}$ and $f_{\alpha_i\cdot}$ for a given value of i, but do not know $f_{\cdot\beta_i}$; we handle the case in which $f_{\alpha_i\cdot}$ is unknown in an analogous manner. By assumption, we have available an estimate $\hat{f}_{\cdot\beta_i}$—e.g., from the optimizer—and we have assumed a known upper bound δ on the magnitude of the relative error of this estimate. It follows that $l_i \le f_{\cdot\beta_i} \le u_i$, where $l_i = \hat{f}_{\cdot\beta_i}/(1+\delta)$ and

$$u_i = \begin{cases} \min\bigl(\hat{f}_{\cdot\beta_i}/(1-\delta), 1\bigr) & \text{if } 0 \le \delta \le 1; \\ 1 & \text{if } \delta > 1. \end{cases}$$

The goal is to choose the estimate $f^*_{\cdot\beta_i}$ of $f_{\cdot\beta_i}$ so as to make the ratio $f_{\alpha_i\beta_i}/(f_{\alpha_i\cdot}f^*_{\cdot\beta_i})$ as close to 1 as possible. Equivalently, we want to make $r_i f^*_{\cdot\beta_i}$ as close to 1 as possible, where $r_i = f_{\alpha_i\cdot}/f_{\alpha_i\beta_i}$. Complicating the situation is the fact that, for some observation $\mathscr{O}_j = \{ f_{\alpha_j\beta_j}, f_{\alpha_j\cdot} \}$ with $j \ne i$, we may have $\beta_j = \beta_i$, which implies that we also want to make $r_j f^*_{\cdot\beta_j} = r_j f^*_{\cdot\beta_i}$ close to 1. In general, setting $J = \{ j:\ \beta_j = \beta_i \}$, we want to find the value y such that $r_j y$ is close to 1 for all $j \in J$. We then use this value as our estimate of $f_{\cdot\beta_i}$, and hence also of $f_{\cdot\beta_j}$ for $j \in J$. The notion of "close" may be captured via Euclidean distance: choose y to minimize $f(y) = \sum_{j\in J}(r_j y - 1)^2$ subject to the constraint that $l \le y \le u$, where $l = l_i$ and $u = u_i$ as defined above. Since $y_0 = \sum_{j\in J} r_j / \sum_{j\in J} r_j^2$ solves the equation $f'(y) = 0$, we take our estimate y^* as either l, u, or y_0, depending on which of $f(l)$, $f(y_0)$, and $f(u)$ is smallest.

The foregoing discussion assumes that $f_{\alpha_i\beta_i} > 0$. If $f_{\alpha_i\beta_i} = 0$ and $f_{\alpha_i\cdot} > 0$, then clearly the value of $f_{\cdot\beta_i}$ most consistent with the independence assumption is $f_{\cdot\beta_i} = 0$. (Similar reasoning shows that $f_{\cdot\beta_j} = 0$ for $j \in J$.) If $f_{\alpha_i\beta_i} = f_{\alpha_i\cdot} = 0$, then $\mathscr{O}_i$ cannot be used to estimate $f_{\cdot\beta_i}$ and the above procedure is applied using the observations $\{ \mathscr{O}_j:\ j \in J - \{i\} \}$. If none of the $\mathscr{O}_j$ observations can be used to estimate $f_{\cdot\beta_i}$, then we simply use the estimate $\hat{f}_{\cdot\beta_i}$.

6. CONSISTENCY ISSUES

We have assumed so far that the feedback observations are mutually consistent, i.e., there exists at least one frequency distribution on $D_A \times D_B$ with marginal and joint frequencies that coincide with those specified in $\mathscr{O}$. Because feedback observations are obtained at different time points, however, and because updates, deletions from, and insertions to the data set may occur in between observations, inconsistencies can occur in practice. This problem is endemic to any feedback-based method, not just ours. A variety of solutions have been proposed to handle the problem, and most of these can be applied in our setting. We therefore give a brief description of some pertinent methods for dealing with inconsistencies, and do not pursue this issue in further detail here.

Since QFRs typically have a timestamp, we can resolve simple problems, such as the presence of two different observations of $f_{\alpha_i\cdot}$, by simply discarding the observation with the older timestamp. Another technique is to monitor a UDI (update-insert-delete) counter, and purge the warehouse periodically when the UDI counter exceeds a threshold; the UDI counter is reset to zero at each purge. This approach limits the extent of possible inconsistencies. If these measure do not suffice, then a solution along the lines of the linear-programming method described in [17, 22] can be applied. Roughly speaking, this method constructs a linear program in which the feedback observations are treated as constraints, and in which there are other constraints that embody basic probability axioms, e.g., constraints of the form $f_{\alpha_i\cdot} \ge f_{\alpha_i\beta_i}$, and $\sum_i f_{\alpha_i\cdot} \le 1$. A pair of "slack variables" is added to each constraint. The slack variables represent the adjustments (positive or negative) to the constraints that are required in order for the complete set of constraints to admit a feasible solution, i.e., for a consistent frequency distribution to exist. The objective function to be minimized is the sum of the slack variables, which corresponds to the sum of the absolute values of the adjustments. Solving the linear program, e.g., using the Simplex Method, yields the minimal adjustments to the observed frequencies needed to resolve any inconsistencies. As in [22], the terms in the objective function can be weighted so as to favor adjustments to older feedback observations.

Algorithm 1 Detecting a dependency from feedback

```
 1: $F$: collection of QFRs
 2: $(A,B)$: attribute-pair of interest
 3: $M$: number of elements in dataset
 4: $n$: number of feedback observations
 5: $p$: max. allowable probability of falsely detecting dependency
 6:
 7: Complete the feedback observations; see Section 5
 8: Adjust feedback observations for consistency; see Section 6
 9: Compute $\mathbf{x}$, $\mathbf{\Sigma}$ for $(A,B)$ from $F$ and $M$, using (3) and (4)
10: DECOMP($\mathbf{\Sigma}, \mathbf{G}, \mathbf{D}, n, r$)          // Schur decomposition of $\mathbf{\Sigma}$
11: for $i = 1$ to $n$ do                        // compute $\mathbf{y} = M^{1/2}\mathbf{Gx}$
12:     $y_i = M^{1/2}\sum_{j=1}^{n} g_{ij}x_j$
13: end for
14: $H_M = 0$
15: for $i = 1$ to $n$ do                        // compute $H_M$ as $\mathbf{y}^t\tilde{\mathbf{D}}\mathbf{y}$
16:     $H_M = H_M + y_i^2\tilde{d}_i$                  // $\tilde{d}_i$ is computed using (5)
17: end for
18: $\theta = $ CHI2INV$(1-p, r)$                // compute the threshold value
19: if $H_M > \theta$ then                       // apply the dependency test
20:     detect dependency
21: else
22:     do not detect dependency
23: end if
```

7. THE DETECTION ALGORITHM

Algorithm 1 gives a high-level description of our off-line method for detecting a dependency. The function DECOMP that is called in Step 10 computes the Schur decomposition of $\mathbf{\Sigma}$, and returns the matrices $\mathbf{G}$ and $\mathbf{D}$, as well as r, the number of nonzero diagonal elements of $\mathbf{D}$. The Schur decomposition can be obtained by computing the eigenvalues and eigenvectors of $\mathbf{\Sigma}$ using algorithms as in [20, Sec. 11.2–11.3]; then $\mathbf{G}$ is the matrix whose rows are the eigenvectors, and the diagonal elements of $\mathbf{D}$ are the corresponding eigenvalues. The function CHI2INV(q,r) called in Step 18 computes the q-quantile of χ^2_r, i.e., the function evaluates the inverse of the χ^2_r cumulative distribution function at the point q; see, for example, [20, p. 221] for an implementation of this function.

The overall complexity of Algorithm 1 is $O(n^3)$, where n is the number of feedback observations [20]. The most expensive step is the Schur decomposition, which has complexity $O(n^3)$; all other steps have complexity at most $O(n^2)$. Although the $O(n^3)$ cost may seem expensive, we show in Section 9.6 that sampling and incremental-maintenance techniques can be used to obtain performance that is quite acceptable in practice.

8. RANKING ATTRIBUTE PAIRS

As discussed in the introduction, most real-world data sets have a large set of dependent attribute pairs that will be identified by our testing procedure. Thus we often need to rank the attribute pairs in order of decreasing dependency. In the case when complete feedback—i.e., complete frequency information and hence the full contingency table—is available, we can simply compute the MSCD as described in Section 3.4 by appropriately normalizing our test statistic. Such complete information is essentially never available in our setting, and normalization becomes nontrivial.

It is theoretically possible to normalize our test statistic $H_M = M\mathbf{x}^t\mathbf{Q}\mathbf{x}$ to lie between 0 and 1, as follows. Suppose that the Schur decomposition used in Section 4 to compute $\mathbf{Q}$ is the canonical version, so that the diagonal elements of $\mathbf{D}$ are the eigenvalues of $\mathbf{\Sigma}$. Let d^* be the smallest of the nonzero diagonal elements and let $\tilde{d}^* = 1/d^*$. It is not hard to see that $\tilde{d}^*$ is the largest eigenvalue of $\mathbf{Q}$. The Courant-Fischer Minimax Theorem (see, e.g., [12, p. 411]) then implies that $\tilde{d}^* = \max_{\mathbf{u}\in\Re^n} \mathbf{u}^t\mathbf{Q}\mathbf{u}/\|\mathbf{u}\|^2$, where $\|\mathbf{u}\| = (\mathbf{u}^t\mathbf{u})^{1/2}$. Hence $H_M/\eta(\mathbf{x}) \in [0,1]$, where $\eta(\mathbf{x}) = M\tilde{d}^*\|\mathbf{x}\|^2$. This method can be numerically unstable, however, because d^* can be close to 0.

We therefore seek some heuristic dependency measures of the form H_M/z that are easy to compute, numerically stable, and yield intuitively satisfying rankings. Some potential choices of z include:

1. *Table cardinality*: $z = M$. Observe that, when comparing attribute pairs in the same dataset, this normalization is equivalent to using the "raw" value of H_M.

2. *Minimum number of distinct values in the full dataset*: $z = \min(|D_A|, |D_B|)$. This normalization is essentially the normalization used when defining the MSCD.

3. *Minimum number of distinct values in the feedback warehouse*: $z = \min(|D'_A|, |D'_B|)$. Here D'_A $(\subseteq D_A)$ is the number of distinct values of attribute A appearing in the feedback records, and similarly for D'_B. This normalization is essentially the "feedback version" of the MSCD normalization.

4. *Minimum number of distinct values used to compute H_M*: $z = \min(n_A, n_B)$. Here n_A is the number of distinct α_i values actually used in computing H_M, and similarly for n_B.[6]

5. *Degrees of freedom*: $z = r$. This normalization can also be viewed as a feedback version of the MSCD normalization.

6. *Threshold θ at $p = 0.05$*: $z = 0.95$-quantile of χ^2_r. This normalization can be viewed as a rough approximation of the theoretical normalization factor $\eta(\mathbf{x})$; whereas $\eta(\mathbf{x})$ represents an upper bound on H_M, the quantity θ represents an approximate "probabilistic" upper bound that is exceeded with low probability. Note that this normalization is a monotone transformation of the previous normalization.

7. *Threshold θ at $p = 0.005$*: $z = 0.995$-quantile of χ^2_r. distribution.

Section 9 contains an empirical evaluation of the resulting dependency measures.

9. EXPERIMENTAL RESULTS

We prototyped the new and CA methods as Java programs running outside DB2; DB2 created dump files during query optimization and processing, which were then reformatted to form a feedback warehouse file. We mined the warehouse using the new dependency-detection and ranking algorithms, as well as the CA algorithm in [1]. Our goals were to (i) compare the various dependency rankings resulting from different normalizations, (ii) explore, as the number of feedback observations varied, both the degree to which the new algorithm detects dependencies and the stability of the dependency rankings, (iii) compare the new algorithm to the CA algorithm, and (iv) examine the scalability of the new detection algorithm as the number of feedback observations grows.

[6]In general, $n_a \le D'_A$ because, e.g., some values of attribute A might not appear in any conjunctive predicate that also involves attribute B. Similarly, $n_B \le D'_B$.

Attribute	MAKE	MODEL	COLOR	YEAR	RANDOM
#DVs	20	40	100	80	200
α	1.1	1.1	1.05	1.09	1.0

Table 1: Attribute characteristics for default SYNTH dataset

9.1 Experimental Setup

All experiments were run on a Pentium III 1.2GHz CPU with 1GB Ram and a 40GB hard disk. We ran our experiments on three datasets, one real and two synthetic. The first dataset comprised real-world data from a Department of Motor Vehicles (DMV), We focused on two tables: the CAR table, which contains information on the make, model, color and year of registration for each car, and the OWNER table, which contains information on the city, state and country in which each car owner lives. Our experiments showed that, by almost any measure, all attribute pairs within each table were dependent to some degree. Some pairs, such as MAKE-MODEL, obey an almost perfect functional dependency, whereas pairs such as COLOR-YEAR are far less dependent (though still not completely independent). As a sanity check, we generated a couple of columns of random data, to generate some pairs that are "truly" independent. Our second database was the TPC-DS database.[7] Our third database was a synthetic database (SYNTH) loosely modeled after the CAR table, for which we were able to control the frequency distributions and degree of dependency, and for which we were able to compute complete contingency tables, and hence the "true" MSCD-based dependency rankings—i.e., the rankings based on complete feedback—for all of the attribute pairs. In more detail, the SYNTH dataset comprises attributes MAKE, MODEL, COLOR, YEAR, and RANDOM (again as a sanity check), with the relative number of distinct values for the attribute matching the pattern in the CAR table. For each attribute, the attribute-value frequencies were generated according to a generalized Zipf distribution with parameter $\alpha \ge 0$; i.e., the relative frequency of the ith most frequent attribute value was proportional to $i^{-\alpha}$. For our "default" SYNTH dataset, the Zipf parameter was chosen to match the frequency distribution seen in the CAR table; Table 1 displays the number of distinct values and Zipf parameter for the default dataset. To control the dependence between a given attribute pair (i.e., column pair), we first sorted each of the two columns by attribute value, so that there was an almost perfect dependency between the attributes. Then we scanned through the second column and, with probability $\rho \in [0,1]$, we swapped the current (ith) row with the Jth row, where J was uniformly distributed on $\{1,2,\ldots,i\}$. Thus if $\rho = 0$, no swaps would occur and the two attributes would remain highly dependent; if $\rho = 1$, then the second column would be randomly shuffled, so that there would be no dependence between the attributes. By varying ρ we could control the dependency of each attribute pair. Table 2 displays the values of ρ used in our experiments. When experimenting with the MAKE-MODEL pair, we adjusted the data slightly so as to obtain a true functional dependency, as in the CAR table.

9.2 Detecting Dependencies

In initial experiments, we found that our algorithm, using significance levels $p = 0.05$ and $p = 0.005$, detected dependencies

[7]See www.tpc.org/tpcds.

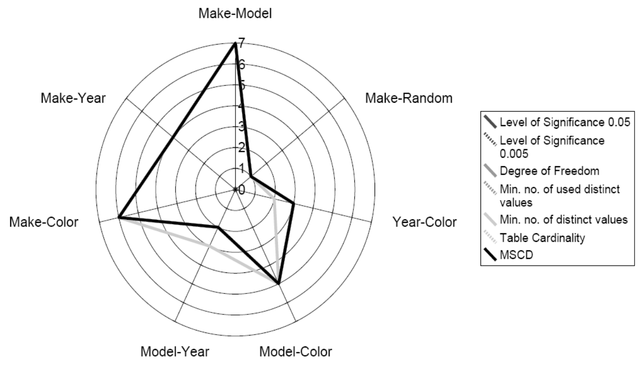

Figure 2: Rankings for various dependency measures (SYNTH)

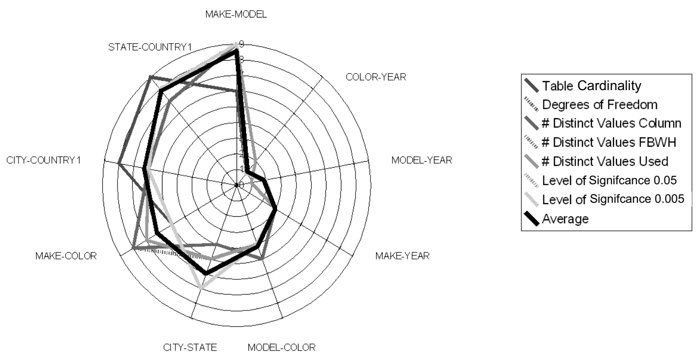

Figure 3: Rankings for various dependency measures (DMV)

Pair	ρ
MAKE-MODEL	0.000
MAKE-YEAR	0.450
MAKE-COLOR	0.005
MODEL-YEAR	0.550
MODEL-COLOR	0.025
YEAR-COLOR	0.700
MAKE-RANDOM	1.000

Table 2: Dependency settings for SYNTH dataset

between virtually every pair of (non-random) attributes, regardless of the number of feedback observations. To see whether this was due to the vagaries of using feedback observations, we also ran several standard chi-squared independence tests (at the same significance levels) using all of the available data. In general, the results for the new procedure were fairly close to that of the standard chi-squared test, with the feedback-based procedure slightly more likely to detect dependencies. More specifically, the results for the pairs of non-random attributes were identical for the standard and feedback-based tests. On the other hand, the standard chi-squared test identified all "truly" independent pairs as independent, whereas the feedback-based procedure, while correctly identifying most of these pairs as independent, misclassified a couple of the pairs when H_M slightly exceeded the threshold θ.

It follows from these results that, with respect to dependency detection, the new feedback-based test reasonably approximates the standard chi-squared test based on complete feedback. The use of either test, however, can result in the identification of many dependent pairs, due to the ubiquity of dependency in real data. Fortunately, our test statistic H_M exhibited intuitively reasonable behavior, taking on large values for highly dependent pairs such as MAKE-MODEL and much lower values for quasi-independent pairs such as COLOR-YEAR. For this reason we shifted attention to the problem of ranking the attribute pairs by degree of dependence. The remaining sections focus on this issue.

9.3 Ranking Measures

We computed the various heuristic ranking measures introduced in Section 8, using both the SYNTH and the DMV datasets. Figures 2 and 3 display these rankings in a radar plot; the degree of dependency is lowest at the center and highest at the outer perimeter. As can be seen, the different normalizations produce rankings that are almost identical for the SYNTH dataset (where they are ordered correctly) and are quite similar for the DMV dataset (where they agree with intuition). Thus, as long as we use a reasonable normalization, we will get reasonable results, and the particular choice of normalization is not that critical.

Normalizations 1 (table cardinality) and 2 (minimum number of distinct values in full dataset) yield the worst rankings. In Figure 3, the three normalizations most closely related to the theoretical ranking $\eta(\mathbf{x})$—namely, normalizations 5,6, and 7—coincide, are the closest to the average ranking, and agree most closely with intuition. (In this radar plot, the curve for normalizations 5–7 has, e.g., rank 7 for CITY-STATE.) We use normalization 7 as our dependency measure for the remainder of our experiments. We also

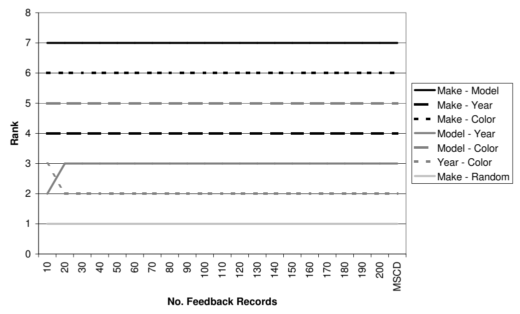

Figure 4: Rank vs. feedback size (SYNTH)

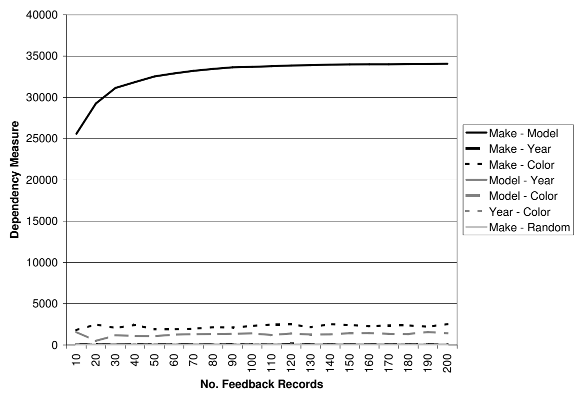

Figure 6: Dependency measure vs. feedback size (SYNTH)

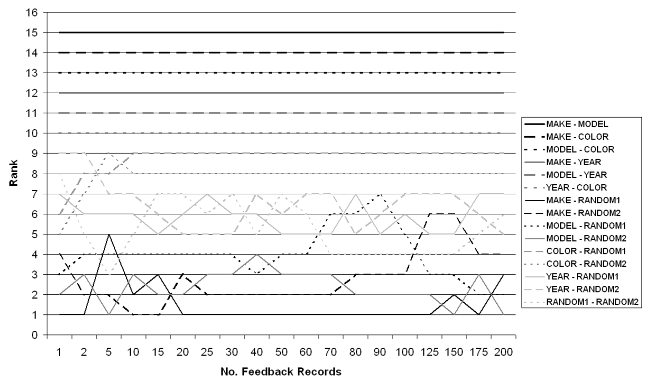

Figure 5: Rank vs. feedback size (DMV)

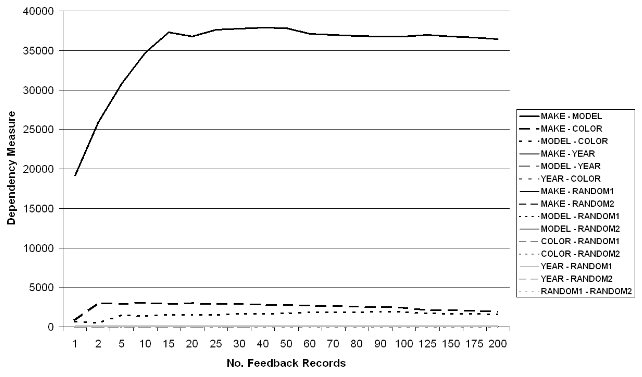

Figure 7: Dependency measure vs. feedback size (DMV)

focus henceforth on the CAR table in the DMV database; results for other tables in the DMV database are similar.

9.4 Stability of Rankings

For our next experiments, we observed the relative ranking of the attribute pairs, as well as the value of the ranking measure (normalized H_M) as the number of feedback observations[8] changed. As can be seen from Figure 4, the relative rankings attribute pairs in the SYNTH database table are extremely stable as the number of feedback observations increases. The rightmost position on the horizontal axis, labeled MSCD, corresponds to the MSCD ranking based on complete feedback. Thus the normalized H_M ranking agrees with the complete MSCD ranking, and both rankings are consistent with the relative degrees of dependency that we induced between the attribute pairs. Observe that the correct ranking is obtained using as few as 20 feedback records. Very similar results were seen to hold when we varied characteristics of the SYNTH dataset. For example, when we decreased each Zipf parameter by 50%, we obtained results almost identical to Figure 4, except that the correct ranking was obtained for all numbers of feedback records.

Figure 5 shows analogous results for the real DMV dataset. As with the synthetic data, the relative rankings for the non-random attribute pairs in the CAR table are extremely stable as the number of feedback observations increases. The noise (oscillating rankings) in the lower half of the chart is not surprising, as those combinations include at least one random attribute and therefore have the same, negligible degree of dependency. Indeed, the figure suggests a robust method for pruning out independent pairs: compute the ranking using different numbers of feedback observations, and eliminate a pair from consideration if there is appreciable variability in the resulting ranking numbers.

[8]We use the terms “feedback observation” and “feedback record” interchangeably in this section.

Figures 6 and 7 show the dependency measure as a function of the number of feedback records for the SYNTH and DMV datasets. In both plots, we can only distinguish clearly three separate curves, for MAKE-MODEL, MAKE-COLOR and MODEL-COLOR. The dependency measures of the remaining pairs are too close to the horizontal axis to be clearly visible. Note that the dependency measure of the functionally dependent attributes MAKE and MODEL exceeds the measure for next most dependent pair MAKE-COLOR by an order of magnitude, as would be expected intuitively. All pairs whose curves are not visible in the diagram have a very small degree of dependency.

Figures 8 and 9 show ranking and dependency-measure results for the table of customer addresses in the TPC-DS Again, the rankings are stable, and correspond to intuition.

9.5 Comparison to CA Method

For purposes of comparison, we ran the CA method on the DMV data, completing missing feedback observations using the method of Section 5. The results for the ranking and for the dependency measure are given in Figures 10 and 11, respectively. Comparing, e.g., Figures 5 and 10, we see that, unlike for our new method,

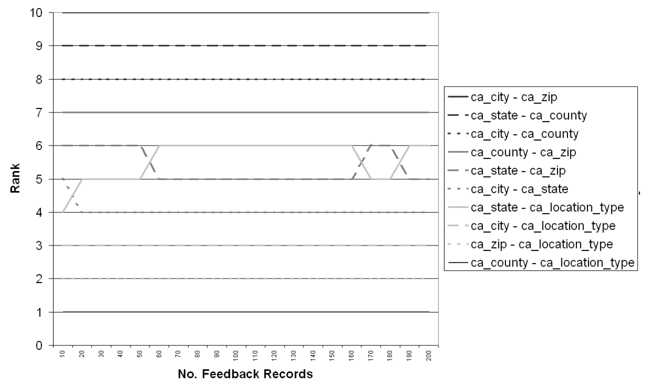

Figure 8: Rank vs. feedback size (TPC-DS)

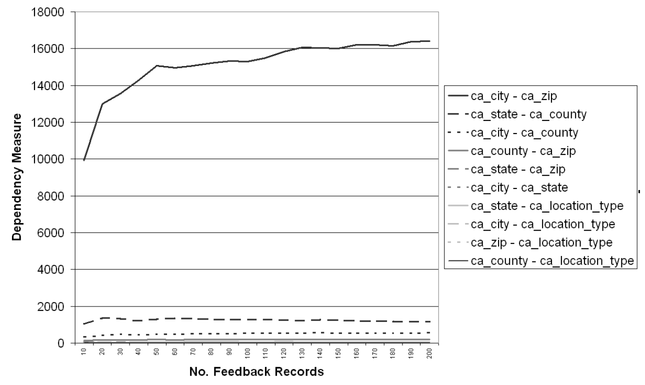

Figure 9: Dependency measure vs. feedback size (TPC-DS)

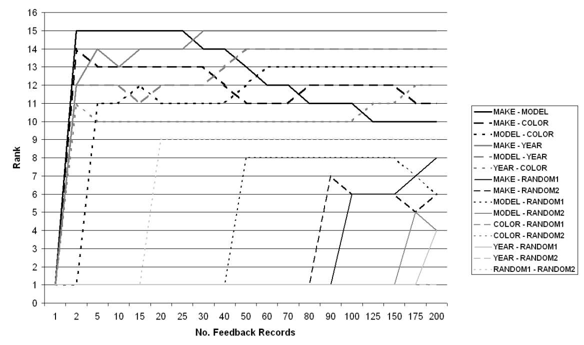

Figure 10: Rank vs. feedback size (CA method)

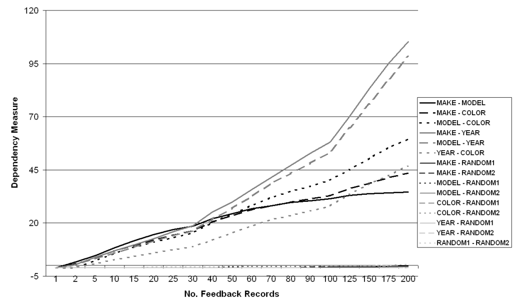

Figure 11: Dependency measure vs. feedback size (CA method)

the ranking produced by the CA method keeps changing as more and more feedback observations are observed. Besides being unstable, the ranking is erratic; for example, given 200 feedback observations, the CA method gives the most highly dependent pair MAKE-MODEL a rank of 6, the lowest rank for a pair that does not contain a RANDOM attribute.

Similar results hold for the dependency measure. Whereas the curves for our new dependency measure are concave, with the dependency measure stabilizing as the number of feedback observations increases, the curves in Figure 11 are convex, again indicating instability. Figure 11 also helps explain why MAKE-MODEL has a rank of 6 in Figure 10. The pair MAKE-MODEL has only 36 distinct-value combinations (which is also the distinct number of models). Therefore, the dependency measure (which is the count of dependent feedback observations) cannot exceed 36, whereas the other pairs have much higher dependency measures because of the higher number of analyzed feedback observations.

In general, the ranking measure for the CA method is especially inaccurate if highly dependent attribute-pairs are seldom queried or if the queries only cover a small set of distinct-value combinations, which is always the case for columns with few such combinations (e.g. MAKE-MODEL). Our new method appears to avoid these drawbacks.

9.6 Execution Time

Figure 12 shows the execution time of the dependency detection algorithm as a function of the number of feedback observations. The empirical results are consistent with the $O(n^3)$ complexity result given in Section 7. As long as there are fewer than a hundred or so feedback records for a given attribute pair, the algorithm can achieve subsecond response times, which is acceptably fast for the off-line warehouse mining task. One way to achieve acceptable practical performance is to run the mining algorithm on a random sample of feedback records. Indeed, in plots such as Figure 5, the feedback records were processed in random order, so that these plots can also be viewed as displaying the relative rankings as a function of sample size. As can be seen, accurate rankings can be obtained with small samples (less than 100 records), thereby keeping the execution cost acceptably low.

If, for some reason, the sampling approach does not suffice, then it may be possible to reduce the processing cost by incrementally and approximately maintaining the statistic H_M. The key observation is that the symmetric Schur decomposition $\mathbf{\Sigma} = \mathbf{G}^t\mathbf{D}\mathbf{G}$ is a special case of the more general "singular value decomposition" (SVD)—see [12, Sec. 2.5.3]—and a large literature exists on low-cost methods for updating the SVD. For illustrative purposes, we briefly outline a relatively simple scheme for approximately updating $\mathbf{G}$ and $\mathbf{D}$, and hence H_M. Suppose that the dimension of $\mathbf{\Sigma}$ is currently $n \times n$. If we obtain a new feedback record, then we effectively need to expand $\mathbf{\Sigma}$ by padding it with $2n+1$ elements computed as in (3). We can view this process as appending an $n \times 1$ column vector $\mathbf{y}$ and then a $1 \times (n+1)$ row vector $\mathbf{z}$. Recall that $r = r(\mathbf{Q})$ is the number of positive[9] diagonal entries of $\mathbf{D}$, and fix a positive integer $k \leq r$. By appropriately renumbering the feedback records (and hence permuting the rows and columns of $\mathbf{\Sigma}$), we can

[9]As discussed before, "positive," in practice, means larger than a small cutoff value ε.

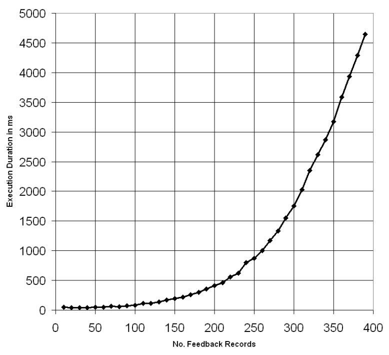

Figure 12: Execution time vs. feedback size

assume that the diagonal elements of $\mathbf{D}$ appear in descending order from upper left to lower right. Denote by $\mathbf{D}_k$ the square diagonal submatrix comprising the first k rows and columns of $\mathbf{D}$; observe that $\mathbf{D}_k$ is nonsingular. Let $\mathbf{G}_k$ denote the submatrix obtained from $\mathbf{G}$ by dropping all but the first k rows of $\mathbf{G}$. Then the simple "folding-in" method of [8]—adapted to our setting—proceeds by appending first the column vector $\mathbf{y}^t\mathbf{G}_k^t\mathbf{D}_k^{-1}$ and then the row vector $\mathbf{z}\mathbf{G}_k^t\mathbf{D}_k^{-1}$ to $\mathbf{G}_k$; the matrix $\mathbf{D}_k$ remains unchanged. The cost of this update is $O(nk)$. We then have $H_M \approx M\mathbf{x}_k^t\mathbf{G}_k^t\mathbf{D}_k^{-1}\mathbf{G}_k\mathbf{x}_k$; computing H_M is an $O(k^2)$ operation. When $k = r$ and there are no numerical roundoff errors, this process is exact; otherwise, error accrues. Lin [19] proposes a metric for the accrued error; when the metric value exceeds a threshold, $\mathbf{G}$ and $\mathbf{D}$ are recomputed from scratch. The updates can also be batched into blocks of m feedback records, i.e., the vectors $\mathbf{y}$ and $\mathbf{z}$ can be replaced by $n \times m$ and $m \times (n+m)$ matrices, respectively. If desired, more accurate (and expensive) updating schemes are available [26].

We conclude by noting that incremental updating and sampling can be combined. For example, we can flip a coin to decide whether to use an incoming feedback record to update H_M. Because we have so far obtained performance that is acceptable in practice using sampling alone, we leave a detailed study of incremental maintenance methods to future work.

10. CONCLUSIONS

We have developed a novel feedback-based method for detecting dependencies between attributes, as well as for ranking attribute pairs according to degree of dependency. Our approach yields stable dependency rankings, in the sense that the rankings are relatively insensitive to the number or order of feedback observations. Especially in combination with sampling of feedback records, our method is fast and simple enough to be practical for commercial database systems. Besides being useful in the context of query optimization, the method is potentially useful in other settings such as data mining, data integration, and system monitoring.

A key open research question is how to extend the methodology to gracefully deal with higher-order dependencies. The results in [16] indicate that, for purposes of query optimization, the benefits of identifying kth-order dependencies diminish sharply as k increases beyond 2. Third-order dependencies may be useful, however, and even higher-order dependencies may be of interest in settings outside of query optimization. We are currently trying to apply some classical methods of contingency-table analysis to this problem.

11. APPENDIX: PROOF OF THEOREM 1

In the following, we write "$X_n \Rightarrow Y$" to denote convergence in distribution [2, p. 338] of the sequence of random variables $\{X_n : n \geq 1\}$ to the limiting random variable Y. That is, if we denote by F_n and F the cumulative distribution functions of X_n and Y, then $X_n \Rightarrow Y$ if and only if $F_n(x) \to F(x)$ for each x at which F is continuous. We assume for convenience that $D_A = \{1,2,\ldots,L_A\}$ and $D_B = \{1,2,\ldots,L_B\}$, and write $L = L_AL_B$, $\mathbf{f} = (f_{11}, f_{12}, \ldots, f_{L_AL_B}) \in [0,1]^L$, and $\mathbf{p} = (p_{11}, p_{12}, \ldots, p_{L_AL_B}) \in [0,1]^L$.

Under our superpopulation model, the strong law of large numbers [2, p. 290] and the multivariate central limit theorem [2, p. 398] imply that $\mathbf{f} \to \mathbf{p}$ with probability 1 and $M^{1/2}(\mathbf{f}-\mathbf{p}) \Rightarrow \mathbf{N}(\mathbf{0},\mathbf{\Lambda})$ as $M \to \infty$. Here $\mathbf{N}$ denotes a multivariate normal random vector, $\mathbf{0} = (0,0,\ldots,0)$ and the $L \times L$ covariance matrix $\mathbf{\Lambda} = \|\lambda_{ij,kl}\|$ is given by

$$\lambda_{ij,kl} = \begin{cases} -p_{ij}p_{kl} & \text{if } (i,j) \neq (k,l) \\ p_{ij}(1-p_{ij}) & \text{if } (i,j) = (k,l). \end{cases} \tag{6}$$

As before, $(\alpha_1,\beta_1),(\alpha_2,\beta_2),\ldots,(\alpha_n,\beta_n)$ are the values that appear in the n feedback observations $\{\mathscr{O}_1,\ldots,\mathscr{O}_n\}$. For $\mathbf{u} = (u_{11}, u_{12}, \ldots, u_{L_AL_B}) \in [0,1]^L$ and $1 \leq i \leq n$, set

$$g_i(\mathbf{u}) = \frac{u_{\alpha_i\beta_i}}{u_{\alpha_i\cdot}u_{\cdot\beta_i}} = \frac{u_{\alpha_i\beta_i}}{\left(\sum_{j=1}^{L_B} u_{\alpha_i j}\right)\left(\sum_{j=1}^{L_A} u_{j\beta_i}\right)}.$$

Thus, using (3) and noting that $g_i(\mathbf{p}) = 1$, we have $x_i = g_i(\mathbf{f}) - g_i(\mathbf{p})$ for $1 \leq i \leq n$. Let $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ be an arbitrary but fixed vector of real numbers, and set $g(\mathbf{u}) = c_1g_1(\mathbf{u}) + c_2g_2(\mathbf{u}) + \cdots + c_ng_n(\mathbf{u})$. A "delta-method" argument [2, p. 402] shows that

$$M^{1/2}\big(g(\mathbf{f}) - g(\mathbf{p})\big) \Rightarrow N(0, \mathbf{h}^t\mathbf{\Lambda}\mathbf{h})$$

as $M \to \infty$, where $\mathbf{h} = \nabla g(\mathbf{p})$ and ∇ denotes the gradient operator. (Note that the limit is a univariate normal random variable.) Using the definition of g, the linearity of the gradient operator, and standard properties of the normal distribution, we can rewrite the above result as $M^{1/2}\mathbf{c}^t\mathbf{x} \Rightarrow \mathbf{c}^t\mathbf{N}(\mathbf{0},\mathbf{\Sigma})$. Here $\mathbf{\Sigma} = \|\sigma_{ij}\|$ is defined by setting

$$\sigma_{ij} = \mathbf{h}_i^t\mathbf{\Lambda}\mathbf{h}_j \tag{7}$$

for $1 \leq i,j \leq L$ with $\mathbf{h}_i = \nabla g_i(\mathbf{p})$. Because the vector $\mathbf{c}$ is arbitrary, the Cramér-Wold Theorem [2, p. 397] implies that

$$M^{1/2}\mathbf{x} \Rightarrow \mathbf{N}(\mathbf{0},\mathbf{\Sigma}) \tag{8}$$

as $M \to \infty$.

Note that $\mathbf{\Sigma}$ is the covariance matrix of a normal random vector $\mathbf{Z}$. Thus $\mathbf{D}$ is the covariance matrix of the normal random vector $\mathbf{G}^t\mathbf{Z}$, and therefore must have nonnegative diagonal entries, as asserted earlier.

We now give a more explicit representation of $\mathbf{\Sigma}$. Taking derivatives, we find that

$$\frac{\partial g_i(\mathbf{u})}{\partial u_{kl}} = \begin{cases} -g_i(\mathbf{u})/u_{\alpha_i\cdot} & \text{if } k = \alpha_i \text{ and } l \neq \beta_i; \\ -g_i(\mathbf{u})/u_{\cdot\beta_i} & \text{if } k \neq \alpha_i \text{ and } l = \beta_i; \\ w_i(\mathbf{u})/(u_{\alpha_i\cdot}^2 u_{\cdot\beta_i}^2) & \text{if } k = \alpha_i \text{ and } l = \beta_i; \\ 0 & \text{if } k \neq \alpha_i \text{ and } l \neq \beta_i, \end{cases}$$

where $w_i(\mathbf{u}) = u_{\alpha_i\cdot}u_{\cdot\beta_i} - u_{\alpha_i\beta_i}(u_{\alpha_i\cdot} + u_{\cdot\beta_i})$. Because $g_i(\mathbf{p}) = 1$ for $1 \leq i \leq n$ and $p_{kl} = p_{k\cdot}p_{\cdot l}$ for all $1 \leq k,l \leq L$ by definition, it follows that

$$\mathbf{h}_{i;kl} = \begin{cases} -1/p_{\alpha_i\cdot} & \text{if } k = \alpha_i \text{ and } l \neq \beta_i; \\ -1/p_{\cdot\beta_i} & \text{if } k \neq \alpha_i \text{ and } l = \beta_i; \\ \gamma_i & \text{if } k = \alpha_i \text{ and } l = \beta_i; \\ 0 & \text{if } k \neq \alpha_i \text{ and } l \neq \beta_i, \end{cases} \tag{9}$$

where $\mathbf{h}_{i;kl}$ denotes the kl component of $\mathbf{h}_i$ and

$$\gamma_i = \bigl(1 - (p_{\alpha_i \cdot} + p_{\cdot \beta_i})\bigr) / (p_{\alpha_i \cdot} p_{\cdot \beta_i}).$$

Using (6), (7), and (9), a tedious but straightforward calculation shows that

$$\sigma_{ij} = \begin{cases} \frac{(1-p_{\alpha_i \cdot})(1-p_{\cdot \beta_i})}{p_{\alpha_i \cdot} p_{\cdot \beta_i}} & \text{if } i = j; \\ -\frac{1-p_{\alpha_i \cdot}}{p_{\alpha_i \cdot}} & \text{if } i \neq j,\ \alpha_i = \alpha_j, \text{ and } \beta_i \neq \beta_j; \\ -\frac{1-p_{\cdot \beta_i}}{p_{\cdot \beta_i}} & \text{if } i \neq j,\ \alpha_i \neq \alpha_j, \text{ and } \beta_i = \beta_j; \\ 1 & \text{if } i \neq j,\ \alpha_i \neq \alpha_j, \text{ and } \beta_i \neq \beta_j. \end{cases} \tag{10}$$

Observe that the definitions of $\mathbf{\Sigma}$ in (4) and (10) coincide, so that the matrix $\mathbf{G}$ defined in Section 4 satisfies $\mathbf{G\Sigma G}^t = \mathbf{D}$. Let

$$\tilde{\mathbf{D}}_0 = \operatorname{diag}\Bigl(\sqrt{\tilde{d}_1}, \sqrt{\tilde{d}_2}, \ldots, \sqrt{\tilde{d}_n}\Bigr),$$

so that $\tilde{\mathbf{D}}_0 \mathbf{G\Sigma G}^t \tilde{\mathbf{D}}_0^t = \tilde{\mathbf{D}}_0 \mathbf{D} \tilde{\mathbf{D}}_0^t = \mathbf{I}_r$, where $\mathbf{I}_r$ is an $n \times n$ diagonal matrix having r diagonal elements equal to 1 and $n - r$ diagonal elements equal to 0. Thus $\tilde{\mathbf{D}}_0 \mathbf{GN}(\mathbf{0}, \mathbf{\Sigma}) = \mathbf{N}(\mathbf{0}, \tilde{\mathbf{D}}_0 \mathbf{G\Sigma G}^t \tilde{\mathbf{D}}_0^t) = \mathbf{N}(\mathbf{0}, \mathbf{I}_r)$ and, using (8) together with the Continuous Mapping Theorem [2, p. 343], we have

$$\begin{aligned} H_M &= (M^{1/2} \tilde{\mathbf{D}}_0 \mathbf{Gx})^t (M^{1/2} \tilde{\mathbf{D}}_0 \mathbf{Gx}) \\ &\Rightarrow \bigl(\tilde{\mathbf{D}}_0 \mathbf{GN}(\mathbf{0}, \mathbf{\Sigma})\bigr)^t \bigl(\tilde{\mathbf{D}}_0 \mathbf{GN}(\mathbf{0}, \mathbf{\Sigma})\bigr) = \mathbf{N}(\mathbf{0}, \mathbf{I}_r)^t \mathbf{N}(\mathbf{0}, \mathbf{I}_r) \end{aligned}$$

as $M \to \infty$. The rightmost term, being distributed as the sum of r independent $N(0,1)^2$ random variables, is well known to have a χ_r^2 distribution, and the desired result follows.

12. REFERENCES

[1] A. Aboulnaga, P. J. Haas, S. Lightstone, G. M. Lohman, V. Markl, I. Popivanov, and V. Raman. Automated statistics collection in DB2 UDB. In *Proc. VLDB*, pages 647–658, 2004.

[2] P. Billingsley. *Probability and Measure*. Wiley, second edition, 1986.

[3] N. Bruno and S. Chaudhuri. Exploiting statistics on query expressions for optimization. In *Proc. ACM SIGMOD*, pages 263–274, 2002.

[4] N. Bruno, S. Chaudhuri, and L. Gravano. STHoles: A multidimensional workload-aware histogram. In *Proc. ACM SIGMOD*, pages 211–222, 2001.

[5] S. Chaudhuri and V. Narasayya. Automating statistics management for query optimizers. *IEEE Trans. Knowledge Data Engrg.*, 13(1):7–20, 2001.

[6] C. E. H. Chua, R. H. L. Chiang, and E.-P. Lim. An intelligent middleware for linear correlation discovery. *Decision Support Sys.*, 32(4):313–326, 2002.

[7] H. Cramér. *Mathematical Methods of Statistics*. Princeton, 1948.

[8] S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *J. Soc. Inform. Sci.*, 41:391–407, 1990.

[9] A. Deshpande, M. Garofalakis, and R. Rastogi. Independence is good: dependency-based histogram synopses for high-dimensional data. In *Proc. ACM SIGMOD*, pages 199–210, 2001.

[10] L. Getoor, B. Taskar, and D. Koller. Selectivity estimation using probabilistic models. In *Proc. ACM SIGMOD*, pages 461–472, 2001.

[11] A. C. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *Proc. STOC*, pages 389–398, 2002.

[12] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins, second edition, 1989.

[13] P. J. Haas and P. G. Brown. BHUNT: Automatic discovery of fuzzy algebraic constraints in relational data. In *Proc. VLDB*, pages 668–679, 2003.

[14] P. J. Haas, J. M. Lake, G. M. Lohman, and T. Syeda-Mahmood. Problem detection via monitoring of metric-stream dependency structure. Submitted for publication, 2006.

[15] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Tiovonen. TANE: An efficient algorithm for discovering functional and approximate dependencies. *Comput. J.*, 42(2):100–111, 1999.

[16] I. F. Ilyas, V. Markl, P. J. Haas, P. Brown, and A. Aboulnaga. CORDS: Automatic discovery of correlations and soft functional dependencies. In *Proc. ACM SIGMOD*, pages 647–658, 2004.

[17] M. Kutsch, P. J. Haas, V. Markl, N. Megiddo, and T. M. Tran. Integrating a maximum-entropy cardinality estimator into DB2 UDB. In *Proc. EDBT*, pages 1092–1096, 2006.

[18] L. Lim, M. Wang, and J. Vitter. SASH: A self-adaptive histogram set for dynamically changing workloads. In *Proc. VLDB*, pages 369–380, 2003.

[19] M.-H. Lin. Out-of-core singular vlaue decomposition. Technical report, Computer Science Department, State University of New York at Stony Brook, 2000.

[20] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, second edition, 1992.

[21] C.-E. Särndal, B. Swensson, and J. Wretman. *Model Assisted Survey Sampling*. Springer-Verlag, 1992.

[22] U. Srivastava, P. J. Haas, V. Markl, and N. Megiddo. ISOMER: Consistent histogram construction using query feedback. In *Proc. ICDE*, 2006.

[23] M. Stillger, G. Lohman, V. Markl, and M. Kandil. LEO - DB2's learning optimizer. In *Proc. VLDB*, pages 19–28, 2001.

[24] J. S. Vitter and M. Wang. Approximate computation of multidimensional aggregates of sparse data using wavelets. In *Proc. ACM SIGMOD*, pages 193–204, 1999.

[25] H. Xiong, P.-N. Tan, and V. Kumar. Exploiting a support-based upper bound of Pearson' correlation coefficient for efficiently identifying strongly correlated pairs. In *Proc. KDD*, pages 334–343, 2004.

[26] H. Zha and H. D. Simon. On updating problems in latent semantic indexing. *SIAM J. Sci. Comput.*, 21(2):782–791, 1999.