# CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies

Ihab F. Ilyas[1] Volker Markl[2] Peter Haas[2] Paul Brown[2] Ashraf Aboulnaga[2]

[1] Purdue University
250 N. University Street
West Lafayette, Indiana, 47906
ilyas@cs.purdue.edu

[2] IBM Almaden Research Center
650 Harry Road, K55/B1
San Jose, CA, 95139
marklv,phaas,pbrown1,aashraf@us.ibm.com

## ABSTRACT

The rich dependency structure found in the columns of real-world relational databases can be exploited to great advantage, but can also cause query optimizers—which usually assume that columns are statistically independent—to underestimate the selectivities of conjunctive predicates by orders of magnitude. We introduce CORDS, an efficient and scalable tool for automatic discovery of correlations and soft functional dependencies between columns. CORDS searches for column pairs that might have interesting and useful dependency relations by systematically enumerating candidate pairs and simultaneously pruning unpromising candidates using a flexible set of heuristics. A robust chi-squared analysis is applied to a sample of column values in order to identify correlations, and the number of distinct values in the sampled columns is analyzed to detect soft functional dependencies. CORDS can be used as a data mining tool, producing dependency graphs that are of intrinsic interest. We focus primarily on the use of CORDS in query optimization. Specifically, CORDS recommends groups of columns on which to maintain certain simple joint statistics. These "column-group" statistics are then used by the optimizer to avoid naive selectivity estimates based on inappropriate independence assumptions. This approach, because of its simplicity and judicious use of sampling, is relatively easy to implement in existing commercial systems, has very low overhead, and scales well to the large numbers of columns and large table sizes found in real-world databases. Experiments with a prototype implementation show that the use of CORDS in query optimization can speed up query execution times by an order of magnitude. CORDS can be used in tandem with query feedback systems such as the LEO learning optimizer, leveraging the infrastructure of such systems to correct bad selectivity estimates and ameliorating the poor performance of feedback systems during slow learning phases.

## 1. INTRODUCTION

When a query optimizer in a commercial relational DBMS chooses a horrible query plan, the cause of the disaster is usually an inappropriate independence assumption that the optimizer has imposed on two or more columns. Independence assumptions are used in virtually all query optimizers because they greatly simplify selectivity estimation: e.g., if $p_1$ and $p_2$ are predicates on respective columns $C_1$ and $C_2$, then the selectivity of the conjunctive predicate $p_1 \wedge p_2$ is estimated by simply multiplying together the individual selectivities of $p_1$ and $p_2$. This approach, however, ignores the rich dependency structure that is present in most real-world data. Indeed, our experience indicates that dependency between columns is the rule, rather than the exception, in the real world.

This paper introduces CORDS (CORrelation Detection via Sampling), an efficient and scalable tool for automatically discovering statistical correlations and "soft" functional dependencies (FDs) between columns. By "correlations," we mean general statistical dependencies, not merely approximate linear relationships as measured, for example, by the Pearson correlation coefficient [7, p. 265]. By a soft FD between columns $C_1$ and $C_2$, we mean a generalization of the classical notion of a "hard" FD in which the value of $C_1$ completely determines the value of $C_2$. In a soft FD (denoted by $C_1 \Rightarrow C_2$), the value of $C_1$ determines the value of $C_2$ not with certainty, but merely with high probability. An example of a hard FD is given by `Country` and `Continent`; the former completely determines the latter. On the other hand, for cars, `Make` is determined by `Model` via a soft dependency: given that `Model` = `323`, we know that `Make` = `Mazda` with high probability, but there is also a small chance that `Make` = `BMW`.

CORDS both builds upon and significantly modifies the technology of the B-HUNT system as described in [12]. As with B-HUNT, CORDS searches for column pairs that might have interesting and useful correlations by systematically enumerating candidate pairs and simultaneously pruning unpromising candidates using a flexible set of heuristics. Also as with B-HUNT, CORDS analyses a sample of rows in order to ensure scalability to very large tables. The similarities end here, however. Whereas B-HUNT uses "bump hunting" techniques to discover soft algebraic relationships between numerical attributes, CORDS employs a robust chi-squared analysis to identify correlations between both numerical and categorical attributes, and an analysis of the number of distinct values in the sampled columns to detect soft FDs. The sample size required for the chi-squared analysis is essentially independent of the database size, so that the algorithm is highly scalable.

CORDS can serve as a data mining tool, e.g., its output can be converted to a dependency graph as in Figure 6.

SIGMOD 2004, June 13–18, 2004, Paris, France.
Copyright 2004 ACM 1-58113-859-8/04/06...$5.00.

Such graphs are of intrinsic interest. Our primary focus, however, is on the application of CORDS in query optimization. Our proposed scheme uses the output of CORDS to recommend groups of columns on which to maintain certain simple joint statistics, such as the number of distinct combinations of values in the columns. An optimizer can then use such "column-group" (CG) statistics to avoid inaccurate selectivity estimates caused by naive independence assumptions. CORDS can be used in conjunction with a query feedback system, leveraging the infrastructure of such a system to correct bad selectivity estimates and ameliorating the poor performance of feedback systems during slow learning phases.

CORDS currently considers only pairs of columns, and not triples, quadruples, and so forth. This restriction vastly reduces the complexity of the algorithm, and our experiments (see Section 5.3) indicate that most of the benefit in query performance is obtained by maintaining the CG statistics of order 2. The CORDS approach, because of its simplicity, restriction to column pairs, and judicious use of sampling, is relatively easy to implement in existing commercial systems, has very low overhead, and scales well to the large numbers of columns and large table sizes found in real-world databases.

The rest of the paper is organized as follows. Section 2 gives an overview of recent technology for relaxing the independence assumption in query optimization, and relates the results in the current paper to this body of work. Section 3 describes the CORDS detection algorithm in detail, including generation of candidate column pairs, pruning of unpromising candidates, sampling-based chi-squared analysis of correlation, and discovery of soft FDs. In Section 4, we discuss our simple but effective method for using the output of CORDS to improve an optimizer's selectivity estimates and hence its choice of query plans. Section 5 contains an experimental evaluation of CORDS in which we validate the correlation detection algorithm using a database with known correlations. We also explore the effect of varying the sampling rate and of using higher-order CG statistics, and demonstrate the speedup in query execution times that can result from the use of CORDS. In Section 6 we describe an application of CORDS to several real-world and benchmark databases. Section 7 contains our conclusions and recommendations for future work.

## 2. RELATED WORK

Recent years have witnessed a large body of work aimed at relaxing the independence assumption in selectivity estimation. As in [12], we can distinguish between "data driven" and "query driven" approaches to detecting and exploiting dependencies among database columns.

### 2.1 Query-Driven Approaches

Some query-driven approaches focus on information contained in a query workload, i.e., a list of relevant queries. For example, Bruno and Chaudhuri [4] use the query workload together with optimizer estimates of query execution times to determine a beneficial set of "SITs" to retain. SITs are statistics (typically multidimensional histograms) on query expressions that can be used to avoid large selectivity estimation errors due to independence assumptions.

Alternatively, a *query feedback system* (QFS) uses feedback from query execution to increase optimizer accuracy. LEO, the DB2 learning optimizer [20], is a typical example of a QFS. LEO compares the actual selectivities of query results with the optimizer's estimated selectivities. In this way, LEO can detect errors caused by faulty independence assumptions and create adjustment factors which can be applied in the future to improve the optimizer's selectivity estimates.

In [1, 5], query feedback is used to incrementally build a multidimensional histogram that can be used to estimate the selectivity of conjunctive predicates. The algorithms in [1, 5] do not discover correlation between columns, however: the set of columns over which to build the histogram must be specified *a priori*.

The SASH algorithm [14] decomposes the set of columns in a table into disjoint clusters. Columns within a cluster are considered correlated and a multidimensional histogram is maintained for these columns. Columns in different clusters are treated as independent. In other words, SASH approximates the full joint distribution of the columns by maintaining detailed histograms on certain low-dimensional marginals in accordance with a high-level statistical interaction model—namely, a Markov network model—and then computing joint frequencies as a product of the marginals. SASH uses query feedback together with a steepest-descent method to incrementally adjust both the structure of the high-level model and the frequency counts in the various histogram buckets.

The advantage of query-driven approaches is that they are efficient, scale well, and result in immediate performance gains, since they focus their efforts on columns that appear in actual queries. The disadvantage is that the system can produce poor estimates—and hence choose poor plans—if it has not yet received enough feedback, either during the initial start-up period or after a sudden change in the workload. Indeed, during one of these slow learning phases the optimizer is likely to avoid query plans with accurate feedback-based cost estimates in favor of more expensive plans that appear to be cheap due to cost estimates based on limited real data and faulty independence assumptions.

### 2.2 Data-Driven Approaches

Data-driven methods analyze the base data to discover correlations between database columns, usually without considering the query workload. These methods can complement the query-driven approaches by identifying correlations between columns that have not yet been referenced in the workload. By proactively gathering information, a data-driven method can mitigate the poor performance of a query-driven method during a slow learning phase.

Most data-driven methods use discovered correlations to construct and maintain a synopsis (lossy compressed representation) of the joint distribution of all the attributes. In some methods, the synopsis takes the form of a graphical statistical model. Getoor, et al. [10], for example, use probabilistic relational models—which extend Bayesian network models to the relational setting—for selectivity estimation. Deshpande, et al. [8] provide a technique that, similarly to SASH, combines a Markov network model with a set of low-dimensional histograms. In [8], however, the synopsis is constructed based on a full scan of the base data rather than from query feedback. Both of the foregoing techniques search through the space of possible models and evaluate them according to a scoring function. Because the number of possible models is exponential in the number of attributes,[1] efficient search techniques and pruning heuristics are essential ingredients of these approaches.

The method of Cheng, et al. [6] typifies a slightly different approach to constructing a synopsis (specifically, a Bayesian network model) from the base data. Instead of searching through a space of possible models as in [8, 10], the method assesses the dependency between pairs of columns by using conditional independence tests based on a "mutual information" measure. The method requires that all attributes be discrete and that there be no missing data values. The method also requires processing of the entire dataset and has a complexity of $O(K^4)$ with respect to $K$, the number of attributes.

As discussed in [12] and references therein, there has also been a great deal of work related to mining of hard FDS and related dependencies. The focus has been on obtaining exact or almost-exact results, and so the amount of work required by these algorithms is much greater than in our setting, where probabilistic results suffice.

Another related body of work concerns the mining of association rules and semantic integrity constraints; see, for example, [18, 19]. These dependencies, however, involve relationships between a few specific values of a pair of attributes, rather than an overall relationship between the attributes themselves. For example, an association might assert that 10% of married people between age 50 and 60 have at least 2 cars. This rule concerns specific values of the `MartialStatus`, `Age`, and `NumberOfCars` attributes.

## 2.3 Relation of CORDS to Previous Work

In terms of the above taxonomy, CORDS is fundamentally a data-driven method, taking as input the base data as well as certain database catalog statistics that are maintained in virtually every commercial database system. We note, however, that CORDS can also exploit workload information to limit the search space of possible column pairs; indeed, CORDS can also exploit schema constraints or semantics provided by users. The CORDS approach is well-suited to integration with a QFS such as LEO, and can potentially be used to initialize a more elaborate data-driven algorithm, such as that in [8]. Unlike some of the schemes discussed above, CORDS can handle either numerical or categorical data.

Because CORDS merely identifies correlated columns instead of producing detailed approximations to the joint attribute frequency distribution, the method's overhead typically is much lower than that of the more elaborate techniques in [6, 8, 10]. The use of sampling in CORDS further reduces its overhead relative to other methods, so that CORDS is well-suited to real-world databases having thousands of tables and hundreds of columns per table. CORDS can also deal relatively well with data that is continually changing, since rechecking column pairs for correlation is much less expensive than updating a synopsis. Implementing CORDS and integrating CORDS with existing commercial optimizers is also much easier than for the more elaborate schemes. The downside is that the optimizer tends to exploit the dependency information provided by CORDS in a much coarser manner than would be the case with a joint-distribution synopsis—as discussed in Section 4.1, the LEO optimizer in essence replaces a rather crude selectivity estimate by a less-crude one. On the other hand, our experiments indicate that even this simple approach can yield dramatic improvements in worst-case query performance.

CORDS may not detect column pairs in which only certain subsets of the rows are correlated. Such detailed relationships are more effectively detected via query feedback. CORDS is designed to quickly obtain a global picture of the data relationships, which can then be refined by a QFS.

# 3. DISCOVERING DEPENDENCIES

In this section, we describe how CORDS discovers correlations and soft FDS. As in the B-HUNT algorithm in [12], the first step is to generate candidate column pairs that potentially have interesting and useful dependencies. For each candidate, the actual dependencies, if any, are detected using both catalog statistics and sampled values from the columns.

## 3.1 Generating Candidates

Similarly to [12], we define a *candidate* as a triple $(a_1, a_2, P)$, where $a_i$ $(i = 1, 2)$ is an attribute (column) of the form $R.c$, such as `EMPLOYEES.ID` or `ACCOUNTS.Balance`. The quantity $P$ is a "pairing rule" that specifies which particular $a_1$ values get paired with which particular $a_2$ values to form the set of pairs of potentially correlated values. When the columns lie in the same table $R$ and each $a_1$ value is paired with the $a_2$ value in the same row, the pairing rule is then trivial and we denote it by the symbol $\emptyset_R$. CORDS also allows columns $a_1$ and $a_2$ to lie in different tables, say $R$ and $S$, where $R$ and $S$ might plausibly be joined during query processing. A pairing rule $P$ in this case is simply a two-table join predicate between $R$ and $S$.[2]

CORDS proceeds similarly to B-HUNT, first generating all candidates having a trivial pairing rule. CORDS then finds all nontrivial pairing rules that "look like" a key-to-foreign-key join predicate, since such join predicates are likely to occur in query workloads. Each nontrivial pairing rule $P$ connects a pair of tables $R$ and $S$, and CORDS generates all candidates of the form $(R.a, S.b, P)$. To find the nontrivial pairing rules, CORDS first identifies the set $K$ comprising columns that are either declared primary or unique keys, together with each column $a$ not of these two types such that

$$\frac{\#\text{distinctValues}(a)}{\#\text{rows}(a)} \geq 1 - \epsilon.$$

(Here $\epsilon$ is a parameter of the algorithm.) For each column $a \in K$, CORDS examines every other column in the schema to find potential matches. A column $b$ is considered a match for column $a$ if either (1) column $a$ is a declared primary key and column $b$ is a declared foreign key for the primary key, or (2) every data value in a sample from column $b$ has a matching value in column $a$.[3] The sample used to check the condition in (2) need not be large; in our implementation the sample size was set at a few hundred rows.

The number of potential candidates is typically quite large for a complex schema with a large number of columns. Therefore CORDS, like B-HUNT, applies a flexible set of heuristic pruning rules to reduce the search space. Some useful types of pruning rules are as follows.

[1] E.g., even though Deshpande, et al. limit their search to "decomposable" models, it can be shown that the number of such models corresponding to five attributes is 1,233.

[2] We allow tables $R$ and $S$ to coincide, so that $P$ can be a self-join predicate.

[3] As in [12], this procedure can be extended to handle certain compound keys.

- *Type Constraints*: Prune candidate columns whose data type does not belong to a specified set, for example, columns whose data are not either integers or strings of less than 100 characters.

- *Statistical Constraints*: Prune tables and columns that do not have specified statistical properties, for example, columns having a small number of distinct values or tables having too few rows. See [12] for further discussion.

- *Pairing Constraints*: Prune candidates whose pairing rules fail to meet specified criteria. For example, only allow pairing rules that correspond to explicitly declared primary-key-to-foreign-key relationships. This constraint would rule out a semantically meaningless join condition such as `CAR.ID = OWNER.ID` on relations `CAR(ID, OwnerID, ...)` and `OWNER(ID, ...)`.

- *Workload Constraints*: Prune candidate columns that do not appear at least once in an equality predicate in a query workload. These and similar constraints can dramatically reduce the size of the search space.

The experiments in Section 5 indicate that even if none of these pruning rules are applied, so that the search space consists of all possible column pairs, the overhead incurred by CORDS may still be acceptable in many cases.

## 3.2 Sampling-Based Testing for Correlation

Each candidate $(a_1, a_2, P)$ generated by CORDS corresponds to a set—call it $\Omega$—containing $N$ $(> 1)$ column-value pairs of the form $(x_1, x_2)$, where the domain of $x_i$ comprises $d_i$ distinct values $(i = 1, 2)$. In this section, we describe how CORDS detects the presence of correlation between the columns based on a random sample of pairs from $\Omega$.[4] Our key observation is that the sample size required for a specified degree of accuracy is essentially independent of the size of the database, so that the correlation-detection algorithm is highly scalable. We provide a closed-form approximate expression for the required sample size.

Suppose initially that $d_1$ and $d_2$ are not too large, and without loss of generality suppose that the $i$th domain is $D_i = \{1, 2, \ldots, d_i\}$; we assume throughout that the domain is known, e.g., from the system catalog. Also suppose initially that the sample used for correlation detection is a simple random sample drawn with replacement from $\Omega$; because the sampling rate is typically very low, the with-replacement sampling assumption has a negligible effect on our analysis.

There are many different measures of association between two attributes: see [11] for some classical statistical measures of association and [3, 17] for some general families of association measures that include Chernov, Kolmogorov, and Kullback-Leibler distances. We measure association using the *mean-square contingency* [7, p.282]:

$$\phi^2 = \frac{1}{d-1} \sum_{i=1}^{d_1} \sum_{j=1}^{d_2} \frac{(\pi_{ij} - \pi_{i\cdot}\pi_{\cdot j})^2}{\pi_{i\cdot}\pi_{\cdot j}}.$$

Here $d = \min(d_1, d_2)$ and, for each $i \in D_1$ and $j \in D_2$, the quantity $\pi_{ij}$ is the fraction of the $N$ $(x_1, x_2)$-pairs for which $x_1 = i$ and $x_2 = j$, and the quantities $\pi_{i\cdot}$ and $\pi_{\cdot j}$ denote marginal totals: $\pi_{i\cdot} = \sum_j \pi_{ij}$ and $\pi_{\cdot j} = \sum_i \pi_{ij}$. We use this measure because it is convenient and has been well studied; our basic techniques can be adapted to other measures of association. As shown in [7], $0 \le \phi^2 \le 1$. The case $\phi^2 = 0$ corresponds to complete factorizability of the joint frequency distribution: $\pi_{ij} = \pi_{i\cdot}\pi_{\cdot j}$ for all $i$ and $j$. If the factorizability condition holds and we sample a column-value pair $(X_1, X_2)$ at random from $\Omega$, then $X_1$ is statistically independent of $X_2$ in that $P\{X_1 = i \text{ and } X_2 = j\} = P\{X_1 = i\} P\{X_2 = j\}$ for all $i \in D_1$ and $j \in D_2$. With a slight (and standard) abuse of terminology, we therefore often use the term "independence" instead of "factorizability" when describing the relationship between such attributes. The case $\phi^2 = 1$ corresponds to a hard FD.[5]

In real-world datasets—or even synthetic datasets where column values are generated "independently" using pseudorandom number generators—the measure $\phi^2$ will never be exactly equal to 0, so we consider the attributes independent if $\phi^2 \le \epsilon$ for some small $\epsilon > 0$. One plausible way of choosing a value of $\epsilon$ is to consider a hypothetical set of $N$ data pairs such that the data is generated according to a random mechanism under which the two components are truly statistically independent, and then to choose $\epsilon$ so that $\phi^2$ exceeds $\epsilon$ with a probability that is small, say, less than 0.01. It is well known that under our model the quantity $(d-1)N\phi^2$ has approximately a chi-squared distribution with $\nu = (d_1 - 1)(d_2 - 1)$ degrees of freedom; see, e.g., [7, 16]. Denoting the cumulative distribution function (CDF) of such a random variable by $G_\nu$ and the inverse of this function by $G_\nu^{-1}$, we have $\epsilon = G_\nu^{-1}(0.99)/\big(N(d-1)\big)$. The quantity $\epsilon$ is typically very small; e.g., for $d = 20$ we have $\epsilon \approx 20/N$, where $N$ typically exceeds $10^6$.

To test for correlation, i.e., to test whether $\phi^2 > \epsilon$, we take a random sample of $n$ pairs from $\Omega$ and consider $\hat{\phi}^2$, the sample estimate of $\phi^2$. This estimate is defined as $\hat{\phi}^2 = \chi^2/\big(n(d-1)\big)$, where

$$\chi^2 = \sum_{i=1}^{d_1} \sum_{j=1}^{d_2} \frac{(n_{ij} - n_{i\cdot}n_{\cdot j})^2}{n_{i\cdot}n_{\cdot j}}, \tag{1}$$

$n_{ij}$ is the number of pairs $(x_1, x_2)$ in the sample for which $x_1 = i$ and $x_2 = j$, and $n_{i\cdot}$ and $n_{\cdot j}$ are the corresponding marginal totals. Our test rejects the hypothesis that the attributes are independent if $\hat{\phi}^2 > u$, where $u$ is a specified constant; equivalently, our test criterion is of the form $\chi^2 > t$. We choose $t$ so that the worst-case probability of incorrectly rejecting the independence hypothesis is smaller than a prespecified value $p$. By "worst-case," we mean the scenario in which the independence hypothesis "barely holds" in that $\phi^2 = \epsilon$. When $\phi^2 = \epsilon$, the test statistic $\chi^2$ has approximately a noncentral chi-squared distribution with $\nu$ degrees of freedom and noncentrality parameter equal to $n(d-1)\epsilon$; see, e.g., [16].[6] Because $\epsilon$ is very small, we

[4] When the pairing rule $P$ is trivial, this sample is obtained by sampling from the table that contains the two columns. Otherwise, CORDS samples from the table containing the "foreign-key" column in the key-to-foreign-key join condition specified by $P$ and then, for each sampled row, finds the unique matching row in the table containing the key column.

[5] The mean-square contingency is not well suited to direct testing for FDs, however, because the value of $\phi^2$ does not determine which of the attributes depends on the other. Moreover, as discussed in the sequel, we often bucketize the data, in which case $\phi^2$ is typically strictly less than 1 even in the presence of a hard FD.

[6] In general, a noncentral chi-squared distribution with $\nu$ degrees of freedom and noncentrality parameter $\lambda$ is characterized as the distri-

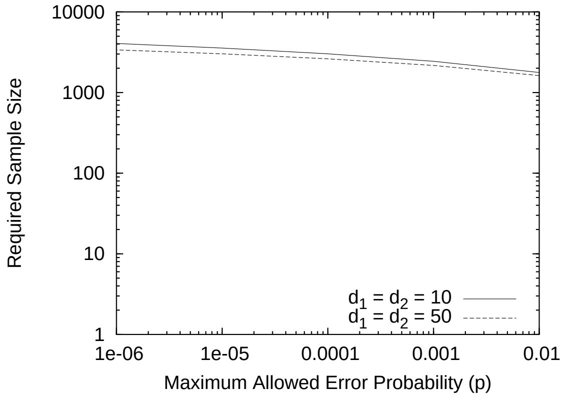


**Figure 1: Required sample size for correlation detection ($\delta = 0.005$).**

can approximate the distribution of $\chi^2$ by an ordinary chi-squared distribution with $\nu$ degrees of freedom. Thus we set $t = G_\nu^{-1}(1-p)$.

Note that our procedure is equivalent to computing the quantity $p^* = 1 - G_\nu(\chi^2)$ and rejecting the independence hypothesis if $p^*$ is less than a cutoff value $p$. The quantity $p^*$, which is called the *p-value* of the test, is the probability of seeing a value of $\chi^2$ as large as (or larger than) the value actually observed, assuming that the attributes are truly independent. Thus the smaller the $p$-value, the more likely that the columns are correlated. The cutoff value $p$ is the maximum allowable probability of incorrectly asserting correlation in the presence of independence.

To determine the sample size $n$, we fix a small constant $\delta > \epsilon$ and take enough samples so that, whenever $\phi^2 \geq \delta$, the test will correctly reject the independence hypothesis with probability at least $1-p$. That is, we choose $n$ such that $G_{\nu,\lambda}(t) = p$. Here $G_{\nu,\lambda}$ denotes the CDF of a noncentral chi-squared distribution with $\nu$ degrees of freedom and noncentrality parameter $\lambda$, with $\lambda$ depending on $n$ via the relation $\lambda = n(d-1)\delta$. Observe that, as asserted earlier, the sample size is essentially independent of the database size $N$. For the case $\delta = 0.005$, Figure 1 displays the required sample size $n$ as a function of $p$ and of the domain sizes $d_1$ and $d_2$. It can be seen that the sample size is relatively insensitive to the values of these parameters. The following approximation can be used to determine $n$ from $p$, $d_1$, and $d_2$:

$$n \approx \frac{\left[-16\,\nu \log\left(p\sqrt{2\pi}\right)\right]^{1/2} - 8\log\left(p\sqrt{2\pi}\right)}{1.69\,\delta(d-1)\nu^{-0.071}}. \qquad (2)$$

Here $\nu = (d_1-1)(d_2-1)$ and $d = \min(d_1, d_2)$ as before. This approximation is accurate to within roughly 2% for $10^{-6} \leq p \leq 10^{-3}$ and $361 \leq \nu \leq 2401$, and to within roughly 5% for $10^{-6} \leq p \leq 10^{-2}$ and $81 \leq \nu \leq 2401$. We obtain the approximation by using the fact [16] that a noncentral chi-squared random variable can be viewed as the sum of $\nu$ i.i.d. squared normal random variables. Thus

$$G_{\nu,\lambda}(x) \approx \Phi\left(\frac{x-(\nu+\lambda)}{\sqrt{2\nu+4\lambda}}\right),$$

when $\nu$ is large, where $\Phi$ denotes the CDF of a standard normal random variable. We then use a standard normal tail estimate [9, p. 175] to approximate $\Phi^{-1}$. Finally, we apply a small empirical correction factor of $\nu^{0.071}/1.69$.

The foregoing discussion suggests that a sample of several thousand rows should yield acceptable results, regardless of the database size. In Section 5.2 we empirically evaluate the effect of sample size on accuracy and execution time.

There are a couple of important issues that must be addressed when applying the foregoing chi-squared analysis in practice. One potential problem occurs whenever many of the $n_{ij}$'s are equal to 0 because the corresponding $\pi_{ij}$'s are nonzero but very small; in this case the chi-squared test, whose derivation assumes a large sample, tends to be unreliable. Indeed, one rule of thumb asserts that chi-squared results cannot be trusted unless at least 80% of $n_{ij}$'s exceed 5. The $n_{ij}$'s can be too small if either of the domains $D_1$ or $D_2$ is very large. CORDS deals with this situation by decomposing each large domain into a relatively small number of disjoint categories. The data is then bucketized accordingly, so that $d_1$ and $d_2$ now represent the number of categories for the two columns, and $n_{ij}$ is now interpreted as the number of pairs $(x_1, x_2)$ such that $x_1$ (resp., $x_2$) belongs to category $i$ (resp., $j$). To help ensure that each $n_{ij}$ is sufficiently large, CORDS examines the frequent-value statistics found in most database system catalogs. Specifically, CORDS checks whether the frequent values of a column account for most of the data, so that the data distribution is noticeably skewed. If so, CORDS uses these frequent values as the column categories (see Section 3.3 for details). Otherwise, CORDS bucketizes the data by hashing, in effect creating categories by dividing the domain into equal size subsets. CORDS also uses the bucketization approach to deal with real-valued data; in this case the categories correspond to disjoint subintervals of the real line.

Another issue concerns *structural zeros*, that is, $(i, j)$ pairs for which $\pi_{ij} = 0$. The chi-squared test is not designed to deal with structural zeros. This problem is partially alleviated by the bucketization procedure described above. If many zero-valued $n_{ij}$'s are still present after bucketization, then CORDS considers these to be structural zeros, and takes the presence of such zeros as direct evidence of correlation between the columns.

## 3.3 The Detection Algorithm

Figure 2 displays the complete CORDS detection algorithm. The algorithm, besides detecting correlations between columns, also identifies soft FDs and soft keys. For ease of exposition, we give the algorithm for the case in which the two columns of interest both lie in some table $R$ and are related via the trivial pairing rule $\emptyset_R$. The modifications required to handle the case of a nontrivial pairing rule $P$ are straightforward; in effect, one simply needs to view table $R$ as the result of executing the join query specified by $P$. We also restrict the discussion to the case of categorical or discrete-valued data—see [15] for a discussion of how to apply the chi-squared test to real-valued data.

In the algorithm, $|C|_T$ denotes the number of distinct values in column $C$ of table $T$ and $|T|$ denotes the number of

bution of the sum of $\nu$ independent squared normal random variables, the $i$th normal random variable having mean $\mu_i$ and variance 1. Here $\lambda = \mu_1^2 + \cdots + \mu_\nu^2$.

```
ALGORITHM DetectCorrelation
INPUT : A column pair $C_1, C_2$ with $|C_1|_R \geq |C_2|_R$

Discover Trivial Cases
1. a.   IF $|C_i|_R \geq (1 - \epsilon_1)|R|$ for $i = 1$ or $i = 2$
        THEN $C_i$ is a soft key; RETURN.
   b.   IF $|C_i|_R = 1$ for $i = 1$ or $i = 2$
        THEN $C_i$ is a trivial column; RETURN.

Sampling
2.      Sample $R$ to produce a reduced table $S$.

Detect Soft Functional Dependencies in the Sample
3. a.   Query $S$ to get $|C_1|_S$, $|C_2|_S$ and $|C_1, C_2|_S$.
   b.   IF $|C_1, C_2|_S \leq \epsilon_2|S|$
          AND $|C_1|_S \geq (1 - \epsilon_3)|C_1, C_2|_S$
        THEN $C_1 \Rightarrow C_2$; RETURN.

Skew Handling for Chi-Squared Test
4. FOR $i = 1, 2$
   a.   IF $\sum_{j=1}^{N_i} F_{ij} \geq (1 - \epsilon_4)|R|$
        THEN
          SKEW$_i$ = TRUE;
          $d_i = N_i$;
          FILTER = "$C_i$ IN $\{V_{i1}, \ldots, V_{iN_i}\}$"
        ELSE
          SKEW$_i$ = FALSE;
          $d_i = \min(|C_i|_R, d_{\max})$;
          FILTER = NULL.
   b.   Apply FILTER.

Sampling-Based Chi-Squared Test
5. a.   Initialize each $n_{ij}$, $n_{i\cdot}$, and $n_{\cdot j}$ to 0.
   b.   FOR EACH column-value pair $(x_1, x_2)$
          $i$ = Category$(1, x_1, d_1, \text{SKEW}_1)$;
          $j$ = Category$(2, x_2, d_2, \text{SKEW}_2)$;
          Increment $n_{ij}$, $n_{i\cdot}$, and $n_{\cdot j}$ by 1;
   c.   IF $\sum_{i=1}^{d_1} \sum_{j=1}^{d_2}$ IsZero$(n_{ij}) > \epsilon_5 d_1 d_2$
        THEN $C_1$ and $C_2$ are correlated; RETURN.
   d.   Compute $\chi^2$ as in (1); set $\nu = (d_1 - 1)(d_2 - 1)$
        and $t = G_\nu^{-1}(1 - p)$.
   e.   IF $\chi^2 > t$
        THEN $C_1$ and $C_2$ are correlated; RETURN.
        ELSE $C_1$ and $C_2$ are independent; RETURN.
```

**Figure 2: The correlation detection algorithm.**

rows in table $T$. Similarly, $|C_1, C_2|_T$ denotes the number of distinct values in the concatenation of columns $C_1$ and $C_2$ in table $T$. We denote the $j$th most frequent value in a column $C_i$ by $V_{ij}$ and the corresponding frequency of the value by $F_{ij}$; we denote by $N_i$ the number of frequent values for $C_i$ that are stored in the system catalog. Finally, we denote by $d_i$ the number of distinct categories for the $C_i$ values in the chi-squared test. Recall that we write $C_1 \Rightarrow C_2$ to denote a soft FD of $C_2$ on $C_1$. The algorithm parameters $\epsilon_1$ through $\epsilon_5$ are small positive constants lying between 0 and 1. The parameter $p$ is the maximum allowed probability of a false-positive or false-negative result for the chi-squared test. Some key details of the algorithm are as follows.

*Step 1.* The quantities $|C_i|_R$ and $|R|$ are obtained from the system catalog. If the number of distinct values in a column $C_i$ is close to the table cardinality $|R|$, then this column is "almost" a key, and we call such a column a *soft key.* A soft key column is trivially statistically correlated with every other column in $R$ because the value of $C_i$ with high probability determines the row, and hence the value in any other column. Therefore, CORDS prunes any column pair for which at least one column is a soft key. CORDS similarly prunes pairs in which at least one column is single-valued, because if $|C_i|_R = 1$, then each column $C_j$ with $j \neq i$ functionally determines $C_i$ in a trivial manner.

*Step 2.* Many commercial systems use either a row-level or page-level Bernoulli sampling scheme. Although these schemes differ somewhat from the simple random sampling scheme assumed in the sample size analysis of Section 3.2, both theoretical and empirical considerations nonetheless continue to support the conclusion that a relatively small fixed sample size—independent of the table cardinality—is adequate for the chi-squared analysis. This is a crucial feature that allows CORDS to scale to large databases. Soft FDs are harder to detect from a sample because, as discussed below, such detection requires accurate sampling-based estimates of the number of distinct values in the two columns and in the concatenation of the two columns. Such estimation is notoriously difficult [13], and the required sample size for a specified degree of accuracy is determined as a percentage of the table cardinality. Fortunately, CORDS usually correctly detects correlation between two columns even when it fails to detect a FD, and correlation detection suffices for the purpose of recommending CG statistics.

*Step 3.* If, in table $R$, column $C_1$ functionally determines $C_2$, then $|C_1|_R/|C_1, C_2|_R = 1$. Therefore, CORDS asserts the existence of a soft FD $C_1 \Rightarrow C_2$ if $|C_1|_S/|C_1, C_2|_S$ is "close" to 1 (specifically, within a distance of $\epsilon_3$). Because determining FDs from samples is somewhat risky, CORDS only tests for a soft FD if the reduced table $S$ contains enough "information" in the sense that $|S| \gg |C_1, C_2|_S$. Intuitively, if the sample $S$ is so small that most column-value pairs $(x, y)$ in the sample are distinct, then a spurious FD will likely be detected. We define the *strength* of a soft FD $C_1 \Rightarrow C_2$ as $|C_1|_R/|C_1, C_2|_R$. This strength is always less than or equal to 1, and a soft FD with strength equal to 1 coincides with a hard functional dependency. CORDS estimates the strength of a soft FD by $|C_1|_S/|C_1, C_2|_S$. Note that CORDS, because it works from a sample, can never assert with complete certainty that a discovered FD is a hard dependency.

*Step 4.* As discussed previously, CORDS uses each frequent value in a column as a category for the chi-squared test when the data distribution is highly skewed; in this case $d_i$, the number of categories, is equal to $N_i$, the number of frequent values in the system catalog. In effect, the chi-squared analysis is applied to a reduced table that is obtained by eliminating infrequent values. Note that the filtering process is not determined by properties of the sample, so that the statistical properties of the test are not distorted. When the data is not skewed, the number of categories is equal to the number of distinct column values, up to a specified limit $d_{\max}$.[7]

*Step 5.* In Step 5.b, CORDS bucketizes the data, if necessary, and creates the "contingency table" for the chi-squared test—i.e., the two-dimensional array of $n_{ij}$ values along with the marginal row ($n_{i\cdot}$) and column ($n_{\cdot j}$) totals. Figure 3 displays the function *Category* that is used for this computation. In the figure, *Hash* is a specified integer-valued hash function.[8] In Step 5.c, CORDS declares columns $C_1$ and $C_2$

[7] Our implementation uses a value of $d_{\max} = 50$.

[8] When $|C_i|_R \leq d_{\max}$, *Hash* can be taken as a simple 1-to-1 mapping

```
FUNCTION Category $(i, x, d, \mathrm{SKEW})$

IF SKEW = TRUE
THEN RETURN $j$ such that $x = V_{ij}$
ELSE RETURN 1+[$Hash(x) \mod d$].
```

**Figure 3: Assigning column values to categories for the chi-squared test.**

to be correlated if there are too many zeros in the contingency table, essentially considering these zeros to be structural zeros because they have persisted in the presence of bucketization. The function *IsZero* that is used in Step 5.c returns 1 if its argument is 0 and returns 0 otherwise.

## 3.4 Displaying the Results Graphically

Especially in the context of data mining, it can be illuminating to display the output of CORDS as a dependency graph in which nodes correspond to columns and arcs correspond to correlation or soft FD relationships; see, for example, Figures 6, 11, and 14 below. Indeed, the name CORDS was partially inspired by the visual "cords" that connect correlated columns. The thickness of the arcs (or, alternatively, the color) can be used to show the strength of the relationships. For soft functional dependencies, the thickness can be an increasing function of the estimated strength, as defined in Section 3.3. For correlations, the thickness can be a decreasing function of the $p$-value or an increasing function of the estimated mean-square contingency.

# 4. CORDS AND QUERY OPTIMIZATION

In this section we describe one possible scheme for using CORDS to improve the accuracy of a query optimizer. In this scheme, CORDS recommends a set of CG statistics for the optimizer to maintain.

## 4.1 Use of Column-Group Statistics

Consider two columns $C_1$ and $C_2$ from some specified table, and suppose that we wish to estimate the selectivity of a conjunctive predicate $p_1 \wedge p_2$, where each $p_i$ is an equality predicate of the form "$C_i = v_i$." A simple and commonly-used selectivity estimate is obtained by first estimating the selectivity of each $p_i$ as $S_{p_i} = 1/|C_i|$, where $|C_i|$ is the number of distinct values in $C_i$, and then obtaining the final selectivity estimate as $S_{p_1 \wedge p_2} = S_{p_1} \cdot S_{p_2} = 1/|C_1| \cdot 1/|C_2|$.

The foregoing estimate usually suffers from two sources of error. First, the selectivity estimates for the individual predicates $p_1$ and $p_2$ assume that the data values in each column are uniformly distributed. Second, the selectivity estimate for $p_1 \wedge p_2$ assumes that $C_1$ and $C_2$ are independent. This latter assumption often results in underestimation of the true selectivity by orders of magnitude. Query optimizers typically impose uniformity assumptions when detailed distributional statistics on individual columns are not available in the system catalog, and impose independence assumptions when statistics are available only for individual columns and not for groups of columns.

| ID | Make | Model |
|---|---|---|
| 1 | Honda | Accord |
| 2 | Honda | Civic |
| 3 | Toyota | Camry |
| 4 | Nissan | Sentra |
| 5 | Toyota | Corolla |
| 6 | BMW | 323 |
| 7 | Mazda | 323 |
| 8 | Saab | 95i |
| 9 | Ford | F150 |
| 10 | Mazda | 323 |

**Figure 4: A table with correlated columns.**

We focus on ameliorating the latter type of error; errors caused by inappropriate independence assumptions often dominate the total error when estimating the selectivity of a conjunctive predicate. Our approach uses CORDS to identify a set of column pairs that are strongly correlated; for each such pair $(C_1, C_2)$, the optimizer collects the CG statistic $|C_1, C_2|$, the number of distinct combinations of values in the two columns. The optimizer then produces the improved selectivity estimate $S'_{p_1 \wedge p_2} = 1/|C_1, C_2|$.

CORDS can profitably be used in conjunction with a QFS, such as LEO, that stores and applies a set of multiplicative adjustment factors to the optimizer's selectivity estimates. E.g., in an enhanced version of LEO, an adjustment factor may be derived either from query feedback or from information provided by CORDS. In this way the strengths of both query-driven and data-driven approaches can be combined. For our particular example, the adjustment factor for $S_{p_1 \wedge p_2}$ is simply $|C_1|\,|C_2|/|C_1, C_2|$. This adjustment factor equals 1 when the columns are truly independent and grows larger as the correlation between the two columns becomes stronger.

EXAMPLE 1. Consider the automobile data in Figure 4, along with a query having a selection predicate $p_1 \wedge p_2$, where $p_1$ = "Make = Honda" and $p_2$ = "Model = Accord". The true selectivity of this predicate is $\sigma_{p_1 \wedge p_2} = 1/10$. The naive estimate is $S_{p_1 \wedge p_2} = 1/|\texttt{Make}| \cdot 1/|\texttt{Model}| = 1/7 \cdot 1/8 = 1/56$, which underestimates the true selectivity by a factor of 5.6, or about −82%. After applying a correction factor of $|\texttt{Make}|\,|\texttt{Model}|/|\texttt{Make}, \texttt{Model}| = 56/9$, we obtain an adjusted estimate of $S'_{p_1 \wedge p_2} = 1/9$, which has an error of only 11%; this residual error is caused by departures from the uniformity assumption. Thus, by using a CG statistic, we have removed the major source of the estimation error. □

Of course, the output of CORDS can be used in more sophisticated ways. For example the optimizer, prompted by CORDS, might collect a more detailed set of CG statistics comprising not only distinct-value information, but also the frequencies for the $k$ most frequent values, along with quantiles. This additional information can be used to alleviate selectivity-estimation errors arising from the uniformity assumption. CORDS can potentially provide guidance when maintaining more complex synopses such as those in [1, 8, 14]. As indicated by the foregoing example and the experiments in Section 5, however, even the simple approach described above can result in dramatic reductions in query processing time. This approach has the important practical advantage of being relatively easy to implement in existing commercial optimizers.

from the set of distinct $C_i$ values to $\{1, 2, \ldots, |C_i|_R\}$. Of course, neither mapping nor hashing may be needed when the data is integer-valued.

```
ALGORITHM RecommendCGS
INPUT: Discovered correlations and soft FDs

1.  Sort correlated pairs, $(C_i, C_j)$ in ascending order
    of $p$-value
2.  Sort soft FDs in descending order of estimated
    strength
3.  Break ties by sorting in descending order of the
    adjustment factor $|C_i|\,|C_j|/|C_i, C_j|$.
4.  Recommend the top $k_1$ correlated column pairs
    and the top $k_2$ soft FDs to the optimizer
```

**Figure 5: Ranking Correlations and Soft FDs.**

## 4.2 Recommending Column-Group Statistics

Real-world databases typically contain large numbers of correlated column pairs. Maintaining CG statistics on each pair for purposes of optimization is usually too expensive and increases the complexity of the selectivity estimation process to an unacceptable degree. This raises the question of exactly which of the discovered pairs should be recommended to the optimizer.

Suppose that, based on storage and processing constraints, the user is willing to store CG statistics for $k_1$ correlated column pairs and $k_2$ soft FDs. Then the algorithm in Figure 4.2 can be used to recommend a set of CG statistics to the optimizer. In Step 1, we can alternatively sort the correlated pairs in descending order of estimated mean-square contingency; either choice ensures that the higher the rank of a column pair, the stronger the correlation between the columns. The tie-breaking rule attempts to select the CG statistics that will have the biggest impact on the optimizer selectivity estimates via the adjustment factor (as described in Section 4.1). Indeed, the adjustment factor may be used as the primary ranking criterion and not just as a tie breaker. Finally, we note that more sophisticated methods, e.g., in the spirit of [4], can potentially be used to decide the set of CG statistics to maintain.

## 5. EXPERIMENTAL EVALUATION

We implemented a CORDS prototype as an application program on top of DB2 V8.1 and applied CORDS to several synthetic, real-world, and benchmark databases. In this section, we describe the validation of CORDS using a synthetic database having known correlations and soft FDs. We also examine the effects of changing the sample size and of using higher-order CG statistics, as well as the overall impact of CORDS on query execution times. All experiments were performed on a UNIX machine with two 400 MHz processors and 1 GB of RAM.

In our experiments, we applied a subset of the pruning rules discussed in Section 3.1. Specifically, we excluded columns having data types comprised of very many bits and limited pairing rules to joins between columns having a declared primary-key-to-foreign-key relationship. We also conducted experiments in which CORDS did not use any pruning rules, in order to test scalability.

## 5.1 Validation

When generating the synthetic database, we created a predetermined set of correlations and soft FDs. We could then analyze the accuracy of CORDS by comparing the set of correlations and soft FDs discovered by the algorithm to the true set.

The schema of the *Accidents* database that we generated contains four relations: `CAR`, `OWNER`, `DEMOGRAPHICS`, and `ACCIDENTS`. Several primary-key-to-foreign-key relationships exist between the tables, and a number of soft FDs between attributes, such as `Make` and `Model`, are inherent in the attribute definitions. We also generated correlations between columns in a table and between columns in different tables by manipulating the joint frequency distribution. For example, we created a correlation between `Model` and `Color`: most of the Accords are Silver, and so forth. The size of the synthetic database is around 1 GB. This size is relatively unimportant, at least with respect to the chi-squared analysis, because CORDS works from a sample and, as discussed previously, the accuracy depends only on the absolute sample size.

We did not explicitly declare any of the primary-key-to-foreign-key relationships in the *Accidents* database, which had the effect of restricting CORDS to discover correlations and soft FDs only between columns in the same table. Such relationships are the most important with respect to the scheme in Section 4.1 for improving selectivity estimates, because optimizers typically do not maintain CG statistics for columns in different tables.

CORDS, using a sample of 4000 rows from each table, discovered all of the synthetically generated correlations and soft FDs, and did not incorrectly detect any spurious relationships. Figure 6 displays the dependency graph for each of the tables in the *Accidents* database. In the figure, the thickness of a solid line ("cord") that represents a correlation increases as the $p$-value decreases, and each dashed arrow that represents a soft FD is labeled with the estimated strength of the FD as defined in Section 3.3.

The graph shows a soft FD `Model` ⇒ `Make` with an estimated strength[9] of 0.92. That the estimated strength of this soft FD is strictly less than 1—and would be less than 1 even for a sampling rate of 100%—is expected since certain models share the same make (e.g., Mazda 323 and BMW 323). The soft FDs in the dependency graph for the `OWNER` relation accurately reflect the hierarchies in the data: `City` ⇒ `State`, `State` ⇒ `Country`, and the transitive closure of these FDs. There were no trivial columns in this example, but CORDS identified a number of soft keys. For example, CORDS identified the columns `ID` and `Name` in `OWNER` table as soft keys. Note that `ID` is an actual key; `Name` is not a key, but has a large number of distinct values close to the relation cardinality, so that any functional dependency or correlation that involves `Name` is likely to be spurious.

## 5.2 The Effect of Sample Size

Our implementation of CORDS exploits the efficient page-level Bernoulli sampling technique supported by DB2. For the sample sizes considered, the cost of retrieving a sample is approximately a linear function of the sample size, and is independent of the size of the database. Figure 7 displays the overall execution time of CORDS as a function of the sample size. As can be seen, the execution time of CORDS increases

[9] The actual strength was 0.9236. CORDS estimated the strengths of each soft FD in the *Accidents* database to within 0.5%. Although this result is encouraging, the accuracy when estimating soft FDs depends in general on the size of both the database and the sample.

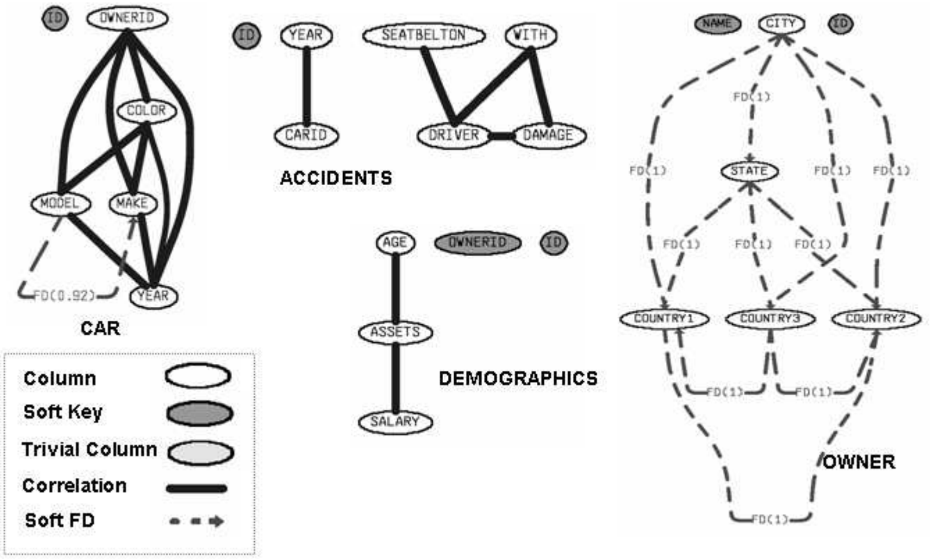


**Figure 6: Dependency graph for the *Accidents* database.**

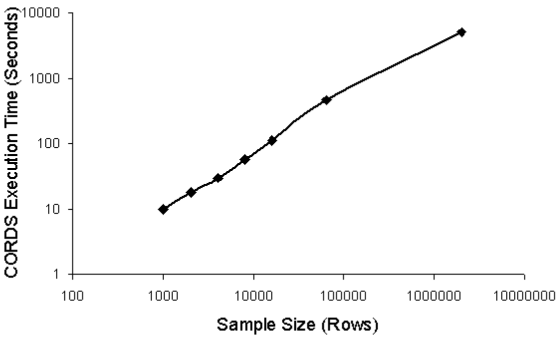


**Figure 7: Effect of sample size on execution time.**

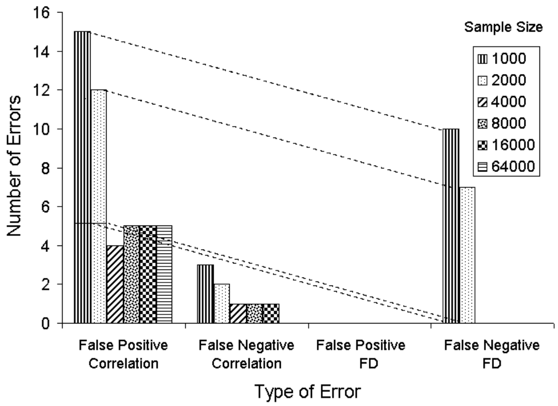


**Figure 8: Accuracy versus sample size.**

in a slightly sublinear manner as the sample size increases from 1000 rows to the entire database (no sampling). For a sample size of few thousand rows, CORDS takes less than a minute to complete, regardless of the size of the database.

The next set of experiments explores the effect of sample size on the quality of the results. Note that CORDS can commit four types of errors. A *false positive* correlation occurs when CORDS erroneously identifies an independent column pair as correlated, and a *false negative* correlation occurs when CORDS erroneously identifies a correlated column pair as independent. False positive and false negative FDs are defined in an analogous manner. Figure 8 displays the frequencies of the four types of errors for various sample sizes.

The number of false positive correlations drops substantially as the sample size increases from 1000 or 2000 rows to 4000 rows. Increasing the sample size beyond 4000 rows does not improve the results. The number of false positive FDs is equal to 0 even for small sample sizes because of the conservative criterion that CORDS uses to detect FDs—see Step 3.b of the algorithm in Figure 2. Although the number of errors at sample sizes of 1000 and 2000 rows may seem relatively large at first, the situation—at least from the point of view of query optimization—is actually not so bad. Observe that a soft FD is a special type of correlation between two columns. Although CORDS misses some soft FDs when there is not enough data (i.e., at small sample sizes), CORDS nonetheless identifies these soft FDs as general correlations. Although each such misclassification results in both a false negative FD and a false positive correlation, CORDS correctly recommends that the optimizer maintain CG statistics on the column pair, so that the two errors cancel out. Thus, if we follow the dotted lines in Figure 8 and subtract the number of false negative FDs from the number of false positive correlations, then we find that the accuracy of CORDS at sample sizes of 1000 and 2000 rows is comparable to the accuracy at larger sample sizes.

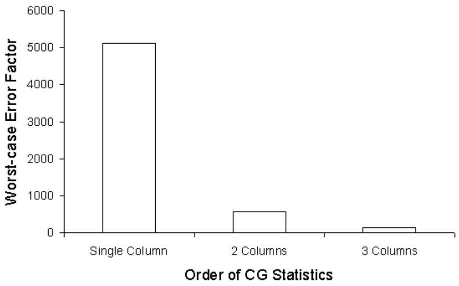


**Figure 9: Effect of column-group order on accuracy of selectivity estimates.**

### 5.3 Higher-Order CG Statistics

CORDS discovers correlations only between pairs of columns. The rationale behind this approach is that such pairs can be identified in a fast and highly scalable fashion, and the resulting CG statistics can be used to correct for most of the errors in selectivity estimation. We tested the latter assumption by measuring the selectivity estimation error when there are either (1) statistics on single columns only, so that the independence assumption is used throughout, (2) CG statistics on column pairs, or (3) CG statistics on column triples. The workload consisted of 300 queries, where each query had a set of three or more equality predicates on correlated columns. The CG statistics for column pairs were obtained by running CORDS, and the CG statistics for column triples were computed for all triples that appeared in at least one predicate in the query workload.[10] For each of the three scenarios, we computed the error factor in selectivity estimation for each query, that is, the ratio of the true to the estimated selectivity. Figure 9 displays the worst-case error factors. As can be seen, maintaining CG statistics on column pairs reduces the worst-case error by an order of magnitude. The figure also clearly shows the diminishing return from maintaining higher order CG statistics, and provides some justification for our approach.

### 5.4 Performance Benefit for Query Execution

To evaluate the overall benefit of using CORDS in the context of query optimization, we ran a workload of 300 queries on the *Accidents* database, both with and without the CG statistics recommended by CORDS. Each query joins the four tables in the schema and applies multiple local predicates on correlated columns. Figures 10(a) and (b) display the box plot and scatter plot for the workload in these two scenarios. As can be seen from Figure 10(a), there is a slight decrease in the median query execution time, and a dramatic decrease—by an order of magnitude—in the worst-case execution time. Inspection of the scatter plot in Figure 10(b) shows that almost all of the plotted points lie below the 45 degree line, i.e., almost all of the queries benefitted from the use of CORDS. Many queries benefitted significantly. Increases in query execution times, when present, were small, and resulted from small inaccuracies in the optimizer's cost model.

## 6. CORDS IN ACTION

This section reports some results that were obtained by applying CORDS to several real-world and benchmark databases. We first explored a subset of the *Census* database comprising a single table having 123 columns. We then looked at the *Auto* database; this real-world database contains motor vehicle information and has over 20 tables and hundreds of columns. As indicated previously, a sample size of a few thousand rows suffices for query optimization. For purposes of data mining, where we would like to more accurately capture soft FDs, initial experiments found a sample size of 10,000 rows to suffice. We therefore use a sample size of either 2000 or 10,000 rows throughout.

As seen from the dependency graph[11] for the *Census* database, based on 2000 rows and displayed in Figure 11, correlations and FDs abound in real-world databases. Figure 12(a) shows the number of discovered correlations corresponding to various upper cutoff levels for the $p$-value and lower cutoff levels for the adjustment factor (defined as in Section 4.1). Figure 12(b) shows the number of discovered soft FDs corresponding to various lower cutoff levels for the estimated strength and the adjustment factor. Figure 13 is analogous to Figure 12, but based on a sample of 10,000 rows from the *Auto* database. The foregoing figures show how real-world databases can contain large numbers of correlations that will likely cause huge errors in selectivity estimation. For example, in the *Auto* database, there are more than 150 correlated pairs that can cause selectivity estimation errors by a factor exceeding 256.

We also applied CORDS to the *TPC-H* benchmark database. CORDS discovered various associations between columns that resulted from the data generation process, as well as from naturally arising primary-key-to-foreign-key relationships and FDs. For example, based on a sample of 10,000 rows, CORDS identified two clusters of attributes in the LINEITEM table; see Figure 14. The first cluster pertains to an order, while the second cluster pertains to a supplier. These hidden correlations can result in order-of-magnitude errors in estimated selectivities, and hence query plans that run in hours instead of seconds. We emphasize that the clusters were discovered without requiring any explicit semantic knowledge about the application domain—CORDS is completely data-driven.

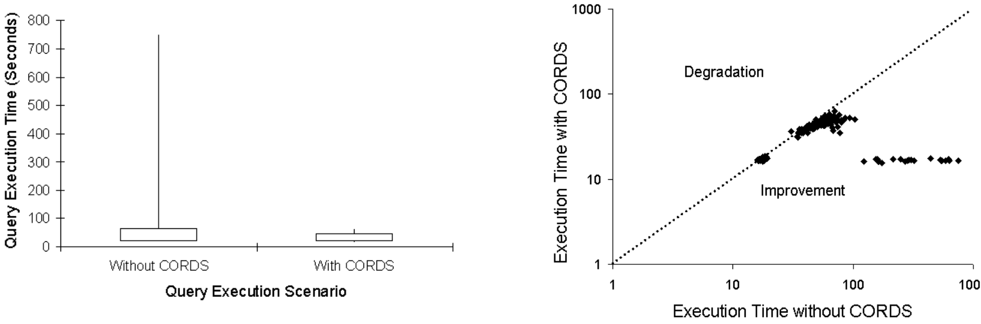
## 7. CONCLUSIONS AND FUTURE WORK

CORDS is a fast and scalable technique for detecting correlations and soft FDs between attributes in a relational database. By combining the candidate-generation framework from the B-HUNT system with robust chi-squared analysis and a judicious use of sampling, CORDS can discover correlations in a fast, efficient and highly scalable manner. Indeed, our experiments show that CORDS can discover correlations and soft FDs in very large databases in a matter of minutes. Our empirical study also shows that real-world databases are rich in such statistical relationships.

CORDS can be used as a standalone database mining tool. Indeed, CORDS is well suited to visualization, because the output of CORDS can easily be transformed into a dependency graph. An interesting topic for future research is how best to display the relationships between database attributes

[10] Presumably an extension of CORDS to column triples would recommend some subset of the CG statistics that we used.

[11] The dependency graphs in this section have the same legend as that for Figure 6.



(a) Box Plot

(b) Scatter Plot

**Figure 10: Effect of CORDS on query execution time.**

using linewidth, texture, color, and so forth. Although we have presented static dependency graphs, one can envision a dynamic graphical front end that permits the user to display first the strongest relationships, then the next strongest relationships, and so forth, by manipulating, e.g., a slider bar. CORDS can also complement current database miners to permit more efficient pruning during the traditional association-rule discovery process. Finally, CORDS can facilitate the preliminary phases of schema discovery and schema design.

We have shown the potential of CORDS as a practical tool for relaxing the independence assumption used by commercial query optimizers when estimating selectivities. We found CORDS relatively easy to prototype in a commercial DBMS, and our experiments show that the use of CORDS can lead to order-of-magnitude improvements both in the accuracy of selectivity estimates and in the resulting query execution times. Moreover, by simultaneously using both CORDS and LEO-style feedback algorithms as a basis for recommending CG statistics to the optimizer, we can potentially obtain the best features of both data- and query-driven methods.

Although CORDS discovers only pairwise correlations, our experiments indicate that exploiting such correlations can remove most of the correlation-induced selectivity estimation errors, though perhaps the discovery and exploitation of three-way correlations may be worthwhile. We suspect however that, for most data-driven methods, the increase in accuracy obtained by capturing high-dimensional joint distributions does not justify the resulting increase in execution cost and programming complexity. A better strategy is to use CORDS as a bootstrapping technique for a QFS if higher-order CG statistics are needed.

In future work, we plan to investigate techniques for extending CORDS to detect three-way correlations, and to see if such an extension is worthwhile for query optimization; one possible approach to this problem is to replace the chi-squared analysis by a more general log-linear analysis as described in [2]. We will also investigate extensions for facilitating schema discovery, for example, discovering soft composite keys and soft FDs of the form $X, .., Y \Rightarrow Z$. We believe that the CORDS technique can be applied in contexts other than relational databases, for example, discovering relationships and schema information in XML data.

# 8. REFERENCES

[1] A. Aboulnaga and S. Chaudhuri. Self-tuning histograms: Building histograms without looking at data. In *Proc. 1999 ACM SIGMOD*, pages 181–192. ACM Press, 1999.

[2] A. Agresti. *Categorical Data Analysis.* Wiley, second edition, 2002.

[3] S. M. Ali and S. D. Silvey. A general class of coefficients of divergence of one distribution from another. *J. Royal Statist. Soc. Ser. B*, 28:131–142, 1966.

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

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

[6] J. Cheng, D. A. Bell, and W. Liu. Learning belief networks from data: An information theory based approach. In *Proc. ACM Conf. Info. Knowledge Mgmt. (CIKM '97)*, pages 325–331, 1997.

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

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

[9] W. Feller. *An Introduction to Probability Theory and Its Applications, Volume I.* Wiley, 1968.

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

[11] L. A. Goodman and W. H. Kruskal. Measures of association for cross-classifications. *J. Amer. Statist. Assoc.*, 49:733–763, 1954.

[12] P. J. Haas and P. G. Brown. BHUNT: Automatic discovery of fuzzy algebraic constraints in relational data. In *Proc. 29th VLDB*, pages 668–679. Morgan Kaufmann, 2003.

[13] P. J. Haas and L. Stokes. Estimating the number of classes in a finite population. *J. Amer. Statist. Assoc.*, 93:1475–1487, 1998.

[14] L. Lim, M. Wang, and J. S. Vitter. SASH: A self-adaptive histogram set for dynamically changing workloads. In *Proc. 29th VLDB*, pages 369–380. Morgan Kaufmann, 2003.

[15] H. B. Mann and A. Wald. On the choice of the number of class intervals in the application of the chi-square test. *Ann. Math. Statist.*, 13:306–317, 1942.

[16] P. B. Patnaik. The non-central $\chi^2$- and $F$-distributions and their applications. *Biometrika*, 36:202–232, 1949.

[17] T. R. C. Read and N. A. C. Cressie. *Goodness-of-Fit Statistics for Discrete Multivariate Data.* Springer, 1988.

[18] M. Siegel, E. Sciore, and S. Salveter. A method for automatic rule derivation to support semantic query optimization. *ACM Trans. Database Syst.*, 17:563–600, 1992.

[19] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. 1996 ACM SIGMOD*, pages 1–12. ACM Press, 1996.

[20] M. Stillger, G. M. Lohman, V. Markl, and M. Kandil. LEO — DB2's LEarning Optimizer. In *Proc. 27th VLDB*, pages 19–28. Morgan Kaufmann, 2001.

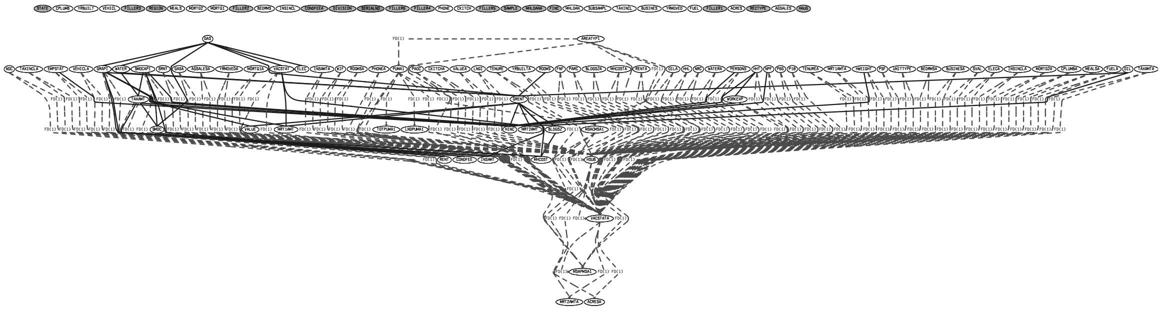
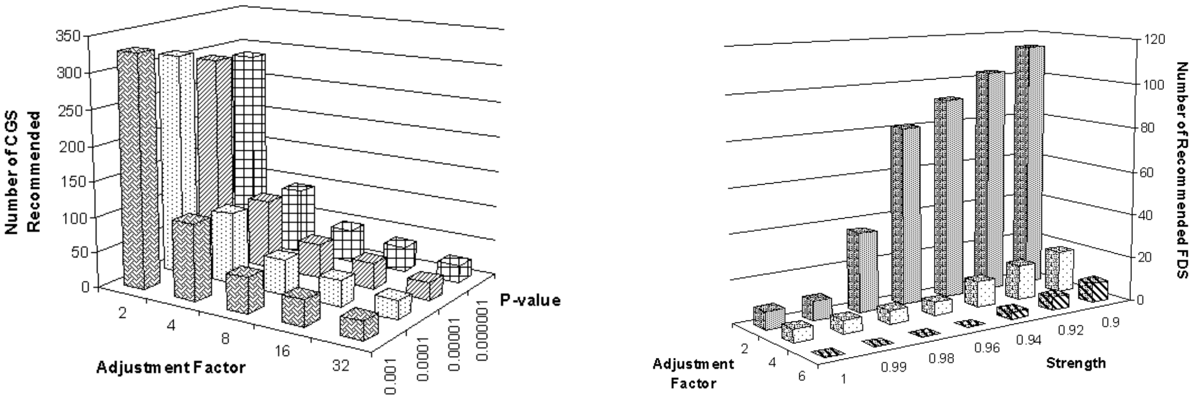


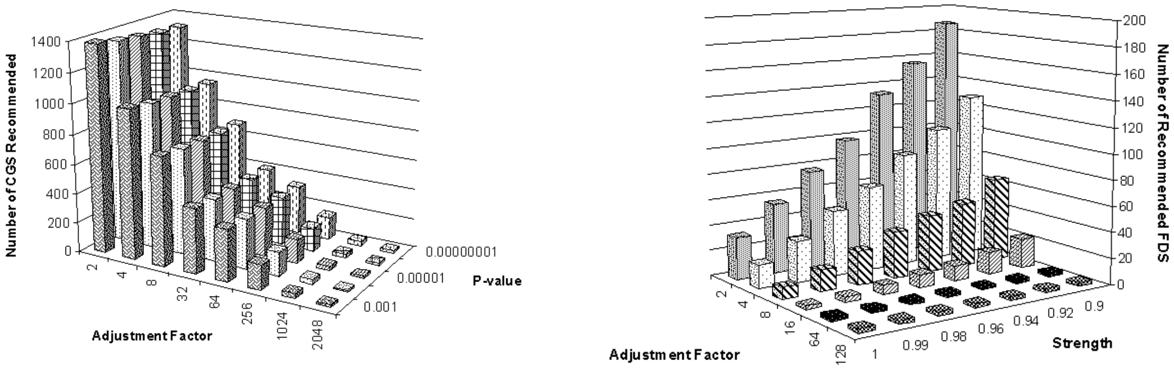
Figure 11: Dependency graph for the *Census* database.



(a) Correlations

(b) Soft FDs

Figure 12: Number of recommended correlations and soft FDs in the *Census* database.



(a) Correlations

(b) Soft FDs

Figure 13: Number of recommended correlations and soft FDs in the *Auto* database.

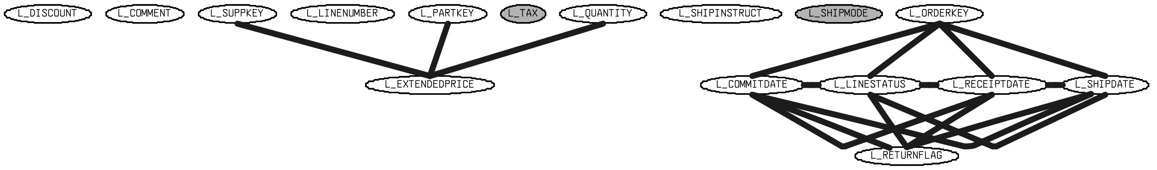


Figure 14: Dependency graph for the LINEITEM table in *TPC-H*.