The VLDB Journal (2007) 16:55–76
DOI 10.1007/s00778-006-0030-1

SPECIAL ISSUE PAPER

Consistent selectivity estimation via maximum entropy

V. Markl · P. J. Haas · M. Kutsch · N. Megiddo · U. Srivastava · T. M. Tran

Received: 15 January 2006 / Accepted: 3 August 2006 / Published online: 15 September 2006

Abstract Cost-based query optimizers need to estimate the selectivity of conjunctive predicates when comparing alternative query execution plans. To this end, advanced optimizers use multivariate statistics to improve information about the joint distribution of attribute values in a table. The joint distribution for all columns is almost always too large to store completely, and the resulting use of partial distribution information raises the possibility that multiple, non-equivalent selectivity estimates may be available for a given predicate. Current optimizers use cumbersome ad hoc methods to ensure that selectivities are estimated in a consistent manner. These methods ignore valuable information and tend to bias the optimizer toward query plans for which the least information is available, often yielding poor results. In this paper we present a novel method for consistent selectivity estimation based on the principle of maximum entropy (ME). Our method exploits all available information and avoids the bias problem. In the absence of detailed knowledge, the ME approach reduces to standard uniformity and independence assumptions. Experiments with our prototype implementation in DB2 UDB show that use of the ME approach can improve the optimizer's cardinality estimates by orders of magnitude, resulting in better plan quality and significantly reduced query execution times. For almost all queries, these improvements are obtained while adding only tens of milliseconds to the overall time required for query optimization.

V. Markl · P. J. Haas (✉) · N. Megiddo
IBM Almaden Research Center, San Jose, CA, USA
e-mail: peterh@almaden.ibm.com

V. Markl
e-mail: marklv@almaden.ibm.com

N. Megiddo
e-mail: megiddo@almaden.ibm.com

M. Kutsch
IBM Germany, Boeblingen, Germany
e-mail: kutschm@de.ibm.com

U. Srivastava
Stanford University, Stanford, CA, USA
e-mail: usriv@stanford.edu

T. M. Tran
IBM Silicon Valley Lab, San Jose, CA, USA
e-mail: minhtran@us.ibm.com

1 Introduction

Estimating the selectivity of predicates has always been a challenging task for a query optimizer in a relational database management system. A classic problem has been the lack of detailed information about the joint frequency distribution of attribute values in the table of interest. Perhaps ironically, the additional information now available to modern optimizers has in a certain sense made the selectivity-estimation problem even harder.

Specifically, consider the problem of estimating the selectivity $s_{1,2,\ldots,n}$ of a *conjunctive predicate* of the form $p_1 \wedge p_2 \wedge \cdots p_n$, where each p_i is a *simple predicate* (also called a *Boolean Factor* or BF) of the form "*column op literal*". Here *column* is a column name, *op* is a relational comparison operator such as "=", ">", or "LIKE", and *literal* is a literal in the domain of the column; some examples of simple predicates are 'MAKE = "Honda"' and 'YEAR > 1984'. By the *selectivity* of a predicate

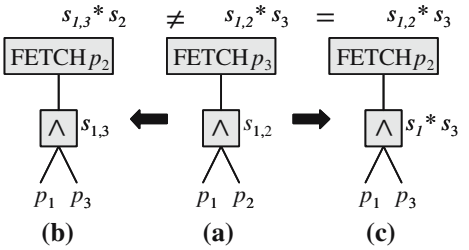

Fig. 1 Query execution plans(QEPs) and selectivity estimation

p, we mean, as usual, the fraction[1] of rows in the table that satisfy p.[2] In older optimizers, statistics are maintained on each individual column, so that the individual selectivities $s_1, s_2, \ldots, s_n$ of $p_1, p_2, \ldots, p_n$ are available. Such a query optimizer would then impose an *independence assumption* and estimate the desired selectivity as $s_{1,2,\ldots,n} = s_1 * s_2 * \cdots * s_n$. This type of estimate ignores correlations between attribute values, and consequently can be wildly inaccurate, often underestimating the true selectivity by orders of magnitude and leading to a poor choice of query execution plan (QEP).

Ideally, to overcome the problems caused by the independence assumption, the optimizer should store the multidimensional joint frequency distribution for all of the columns in the database. In practice, the amount of storage required for the full distribution is exponentially large, making this approach infeasible. Both researchers and system developers have therefore proposed storage of selected *multivariate statistics* (MVS) that summarize important partial information about the joint distribution. Proposed MVS include multidimensional histograms [29] on selected columns and simple "column-group statistics" as used in DB2 UDB [21]; see Sect. 9 for other examples. Thus, for predicates $p_1, p_2, \ldots, p_n$, the optimizer typically has access to the individual selectivities $s_1, s_2, \ldots, s_n$ as well as a limited collection of joint selectivities, such as $s_{1,2}, s_{3,5}$, and $s_{2,3,4}$. The independence assumption is then used to "fill in the gaps" in the incomplete information, e.g., we can estimate the unknown selectivity $s_{1,2,3}$ by $s_{1,2} * s_3$.

A new and serious problem now arises, however. There may be multiple, non-equivalent ways of estimating the selectivity for a given predicate. Figure 1, for example, shows possible QEPs for a query consisting of the conjunctive predicate $p_1 \wedge p_2 \wedge p_3$. The QEP in Fig. 1a uses an index-ANDing operation ($\wedge$) to apply $p_1 \wedge p_2$ and afterwards applies predicate p_3 by a FETCH operator, which retrieves rows from a base table according to the row identifiers returned from the index-ANDing operator.

Suppose that the optimizer knows the selectivities s_1, s_2, s_3 of the BFs p_1, p_2, p_3. Also suppose that it knows about a correlation between p_1 and p_2 via knowledge of the selectivity $s_{1,2}$ of $p_1 \wedge p_2$. Using independence, the optimizer might then estimate the selectivity of $p_1 \wedge p_2 \wedge p_3$ as $s^a_{1,2,3} = s_{1,2} * s_3$.

Figure 1b shows an alternative QEP that first applies $p_1 \wedge p_3$ and then applies p_2. If the optimizer also knows the selectivity $s_{1,3}$ of $p_1 \wedge p_3$, use of the independence assumption might yield a selectivity estimate $s^b_{1,2,3} = s_{1,3} * s_2$. However, this would result in an inconsistency if, as is likely, $s^a_{1,2,3} \neq s^b_{1,2,3}$. There are potentially other choices, such as $s_1 * s_2 * s_3$ or, if $s_{2,3}$ is known, then $s_{1,2} * s_{2,3}/s_2$; the latter estimate amounts to a conditional independence assumption (see Appendix C). Any choice of estimate will be arbitrary, since there is no supporting knowledge to justify ignoring a correlation or assuming conditional independence; such a choice will then arbitrarily bias the optimizer toward choosing one plan over the other. Even worse, if the optimizer does not use the same choice of estimate every time that it is required, then different plans will be costed inconsistently, leading to "apples and oranges" comparisons and unreliable plan choices.

Assuming that the QEP in Fig. 1a is the first to be evaluated, a modern optimizer would avoid the foregoing *consistency* problem by recording the fact that $s_{1,2}$ was applied and then avoiding future application of any other MVS that contain either p_1 or p_2, but not both. In our example, the selectivities for the QEP in Fig. 1c would be used and the ones in Fig. 1b would not. The optimizer would therefore compute the selectivity of $p_1 \wedge p_3$ to be $s_1 * s_3$ using independence, instead of using the MVS $s_{1,3}$. Thus the selectivity $s_{1,2,3}$ would be estimated in a manner consistent with Fig. 1a. Note that, when evaluating the QEP in Fig. 1a, the optimizer used the estimate $s^a_{1,2,3} = s_{1,2} * s_3$ rather than $s_1 * s_2 * s_3$, since, intuitively, the former estimate better exploits the available correlation information. In general, there may be many possible choices; the complicated (ad hoc) decision algorithm used by DB2 UDB is described in more detail in Appendix A.

Although the ad hoc method described above ensures consistency, it ignores valuable knowledge, e.g., of the correlation between p_1 and p_3. Moreover, this method complicates the logic of the optimizer, because cumbersome bookkeeping is required to keep track of how an estimate was derived initially and to ensure that it

[1] We sometimes refer to the *cardinality* of a predicate, which is the absolute number of satisfying rows.

[2] Note that without loss of generality each p_i can also be a disjunction of simple predicates or any other kind of predicate (e.g., subquery, IN-list). For this work we only require that the optimizer has some way to estimate the selectivity s_i of p_i.

will always be computed in the same way when costing other plans. Even worse, ignoring the known correlation between p_1 and p_3 also introduces *bias* towards certain QEPs: if, as is often the case with correlation, $s_{1,3} \gg s_1 * s_3$, and $s_{1,2} \gg s_1 * s_2$, and if $s_{1,2}$ and $s_{1,3}$ have comparable values, then the optimizer will be biased towards the plan in Fig. 1c, even though the plan in Fig. 1a might be cheaper, i.e., the optimizer thinks that the plan in Fig. 1c will produce fewer rows during index-ANDing, but this might not actually be the case. In general, an optimizer will often be drawn towards those QEPs about which it knows the least, because use of the independence assumption makes these plans seem cheaper due to underestimation. We call this problem "fleeing from knowledge to ignorance".[3]

In this paper, we provide a novel method for estimating the selectivity of a conjunctive predicate; the method exploits and combines all of the available MVS in a principled, consistent, and unbiased manner. Our technique rests on the principle of *maximum entropy* (ME) [16,17], which is a mathematical embodiment of Occam's Razor and provides the "simplest" possible selectivity estimate that is consistent with all of the available information. In the absence of detailed knowledge, the ME approach reduces to standard uniformity and independence assumptions; see Appendix B. Our new approach avoids the problems of inconsistent QEP comparisons and the flight from knowledge to ignorance. A preliminary version of this work appeared in [23,25].

We emphasize that the ME method is the first to systematically exploit *all* of the available MVS and actually refine the optimizer's cardinality model beyond the information explicitly given by the statistics. Our techniques are also the first to systematically address the problem of inconsistencies in available MVS. Finally, as discussed in Sect. 9, our results differ from virtually all current and previous work in this area, which deals only with recommending [5,7,8,13,21], constructing [6,32], storing [28], and maintaining [1,4,33] multivariate statistics. Indeed our methods can be used in conjunction with any of these latter technologies.

Thus the contributions of our paper are (1) enunciating and formalizing the problem of inconsistency and bias during QEP evaluation in the presence of partial knowledge about the joint frequency distribution, (2) proposing a new method for cardinality estimation in this setting that exploits all available distributional information, (3) providing an efficient and robust method for computing consistent and unbiased selectivity estimates using the ME principle, and (4) providing a detailed experimental evaluation of our approach with respect to quality and computation time, as well as a comparison to the method used by the DB2 UDB optimizer. Our work appears to be the first to apply information-theoretic ideas to the problem of producing consistent selectivity estimates.

The paper is organized as follows. Section 2 gives some background and formalizes the selectivity-estimation problem. In Sect. 3, we describe the ME approach to unbiased, efficient, and consistent selectivity estimation. We show how the iterative scaling algorithm can be applied in our setting; this well-known algorithm uses a Lagrange-multiplier approach to numerically compute an approximate ME solution. Section 4 describes how our method deals with inconsistencies in the available MVS that would otherwise prevent computation of a ME solution; such inconsistencies can arise when the single-column statistics in the catalog have been computed only approximately, or when the various statistics used by the optimizer have been computed at different points in time. We then show in Sect. 5 how the efficiency of the ME computation can be improved – often by orders of magnitude – by partitioning the predicates into disjoint sets and computing a ME distribution for each of the resulting subproblems. In Sect. 6, we extend our methodology to estimate selectivities of predicates that involve multiple tables, i.e., join predicates. Section 7 illustrates how our methodology can easily be integrated into a commercial system such as DB2 UDB, and Sect. 8 provides an experimental evaluation. After surveying related work in Sect. 9, we conclude in Sect. 10. Appendix A describes the current state of the art in using MVS for cardinality estimation in DB2 UDB, and Appendix B contains formal proofs of the assertion that our new ME approach generalizes classical independence and uniformity assumptions. We show in Appendix C how certain ME selectivities can be computed analytically.

2 Background

Commercial query optimizers [3,19,20,26] use statistical information on the number of rows in a table and the number of distinct values in a column to compute the selectivities of simple predicates. Assuming 10 distinct values in the MAKE column and using the *uniformity assumption*, the selectivity of the simple predicate p_1: 'MAKE = "Honda"' is estimated as $s_1 = 1/10$. Similarly, with 100 distinct values in the MODEL column and 10 distinct values in the COLOR column, we obtain $s_2 = 1/100$ for p_2: MODEL = "Accord" and $s_3 = 1/10$ for p_3: COLOR = "red". Advanced commercial optimizers can improve upon these basic estimates by maintaining

[3] This expression was originally coined by G. M. Lohman.

frequency histograms on the values in individual columns.

As indicated previously, current optimizers compute the selectivity of a conjunctive predicate using the independence assumption in the absence of other information. For instance, $p_{1,2,3} = p_1 \wedge p_2 \wedge p_3$ is the predicate restricting a query to retrieve all red Honda Accords, and the selectivity of $p_{1,2,3}$ is computed as $s_{1,2,3} = s_1 * s_2 * s_3$. In our example, the optimizer would estimate the selectivity of red Honda Accords to be 1/10,000. Because only Honda makes Accords, there is a strong correlation between these two columns, indeed, a functional dependency. The true selectivity of $p_{1,2}$ is therefore 1/100, and a more appropriate estimate of the selectivity of $p_{1,2,3}$ is 1/1,000, one order of magnitude larger than the estimate based on the independence assumption.

2.1 Formalizing the selectivity estimation problem

We now formalize the problem of selectivity estimation for conjunctive predicates, given partial MVS, and define some useful terminology. Let $P = \{p_1, \ldots, p_n\}$ be a set of BFs. For any $X \subseteq N = \{1, \ldots, n\}$, denote by p_X the conjunctive predicate $\wedge_{i \in X} p_i$. Let s be a probability measure over 2^N, the powerset of N, with the interpretation that s_X is the selectivity of the predicate p_X.[4] Usually, for $|X| = 1$, the histograms and column statistics from the system catalog determine s_X and are all known. For $|X| > 1$, the MVS may be stored in the database system catalog either as multidimensional histograms, index statistics, or some other form of column-group statistics or statistics on intermediate tables. In practice, s_X is not known for all possible predicate combinations due to the exponential number of combinations of columns that can be used to define MVS. Suppose that s_X is known for every X in some collection[5] $T \subset 2^N$. Then the selectivity estimation problem is to compute s_X for $X \in 2^N \backslash T$.

It is intuitively clear that the query optimizer should avoid any extraneous assumptions about the unknown selectivities while simultaneously exploiting all existing knowledge, in order to avoid unjustified bias towards any particular solution. In Appendix A, we survey the method that DB2 uses to compute missing selectivities and illustrate why this approach cannot use all existing knowledge without producing an inconsistent model. In the following section, we present the ME principle, which formalizes the notion of avoiding bias.

[4] The quantity s_X can also be interpreted as the probability that a randomly selected row satisfies p_X.

[5] Note that the empty set $\emptyset$ is always part of T, because $s_\emptyset = 1$ when applying no predicates.

2.2 The maximum-entropy principle

The ME principle [16,17] models all that is known and assumes nothing about the unknown. It is a method for analyzing the available information in order to determine a unique epistemic probability distribution. Information theory [31] defines for a probability distribution $\boldsymbol{q} = (q_1, q_2, \ldots)$ a measure of uncertainty called *entropy*:

$$H(\boldsymbol{q}) = -\sum_i q_i \log q_i.$$

The ME principle prescribes selection of the unique probability distribution that maximizes the entropy function $H(\boldsymbol{q})$ and is consistent with respect to the known information.

Entropy maximization without any additional information uses the single constraint that the sum of all probabilities equals 1. The ME probability distribution is then the uniform distribution. When constraints only involve marginals of a multivariate distribution, the ME solution coincides with the independence assumption. See Appendix B for a proof of these assertions.

Thus, query optimizers that do not use MVS actually estimate their selectivities for conjunctive queries according to the ME principle: they assume uniformity when no information about column distributions is available, and they assume independence because they do not know about any correlations. By integrating the more general ME framework into the optimizer's cardinality model, we thereby generalize these concepts of uniformity and independence. Our approach enables the optimizer to take advantage of all available information in a consistent way, avoiding inappropriate bias towards any given set of selectivity estimates.

3 Selectivity estimation via maximum entropy

The ME principle applied to selectivity estimation means that, given several selectivities of simple predicates and conjuncts, we choose the most uniform/independent selectivity model consistent with this knowledge.

3.1 The constrained optimization problem

Given a set of predicates $P = \{p_1, p_2, \ldots, p_n\}$, denote each of the corresponding *atoms* – terms in disjunctive normal form (DNF) – by a binary string of length n. For example, when $n = 3$, the string $b = 100$ denotes the atom $p_1 \wedge \neg p_2 \wedge \neg p_3$, and so forth. As before, set $N = \{1, 2, \ldots, n\}$ and denote by 2^N the set of all subsets of N. For each predicate $p_X, X \in 2^N$, denote by $\mathrm{C}(X)$

the set of *components* of X, i.e., the set of all atoms contributing to p_X. Formally,

$$C(X) = \{b \in \{0,1\}^n | b_i = 1 \text{ for all } i \in X\} \quad \text{and}$$
$$C(\emptyset) = \{0,1\}^n.$$

For example, for predicates p_1 and $p_{1,2}$ we have

$$C(\{1\}) = \{100, 110, 101, 111\} \text{ and } C(\{1,2\}) = \{110, 111\}.$$

Additionally, for each possible "knowledge set" $T \subseteq 2^N$, we denote by $P(b,T)$ the set of all $X \in T$ such that p_X has b as an atom in its DNF representation, i.e.,

$$P(b,T) = \{X \in T | b_i = 1 \text{ for all } i \in X\} \cup \{\emptyset\}.$$

Thus, for the atom 011 and $T = 2^{\{1,2,3\}}$ we have $P(b,T) = \{\{2\},\{3\},\{2,3\},\emptyset\}$.

Given s_X for $X \in T$, we compute s_X for $X \notin T$ according to the ME principle. To this end, we must solve the following *constrained optimization problem*:

$$\underset{x_b \,|\, b \in \{0,1\}^n}{\text{minimize}} \sum_{b \in \{0,1\}^n} x_b \log x_b \tag{3.1}$$

subject to the $|T|$ constraints

$$\sum_{b \in C(X)} x_b = s_X, \quad X \in T, \tag{3.2}$$

where $x_b \in [0,1]$ denotes the selectivity of atom b. The constraints correspond to the known selectivities; one of the included constraints is $s_\emptyset = \sum_{b \in \{0,1\}^n} x_b = 1$, which asserts that the combined selectivity of all atoms is 1. The solution is a probability distribution with the maximum value of uncertainty (entropy), subject to the constraints. Given this solution, we can compute any arbitrary selectivity s_X as $s_X = \Sigma_{b \chi C(X)} x_b$. We can solve the above problem analytically only in simple cases with a small number of unknowns. In general, a numerical solution method is required.

3.2 Example

Figure 2 shows the probability space for the case $N = \{1,2,3\}, T = \{\{1\},\{2\},\{3\},\{1,2\},\{1,3\},\emptyset\}$, and selectivities $s_1 = 0.1, s_2 = 0.2, s_3 = 0.25, s_{12} = 0.05, s_{13} = 0.03$, and $s_\emptyset = 1$.

We have the following six constraints:

(i) $s_1 = x_{100} + x_{110} + x_{101} + x_{111} = 0.1$
(ii) $s_2 = x_{010} + x_{011} + x_{110} + x_{111} = 0.2$
(iii) $s_3 = x_{001} + x_{011} + x_{101} + x_{111} = 0.25$
(iv) $s_{1,2} = x_{110} + x_{111} = 0.05$

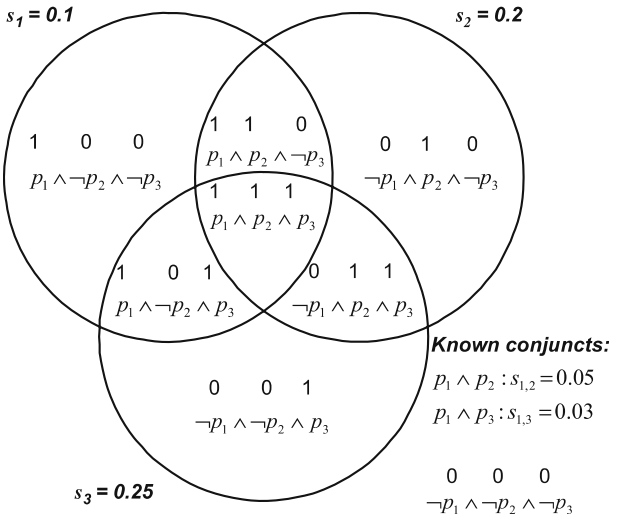

Fig. 2 Probability space for N = {1,2,3} and T = {{1}, {2}, {3}, {1,2}, {1,3}, ∅}

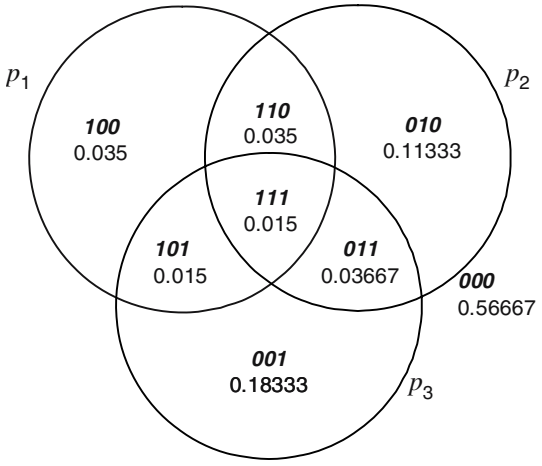

Fig. 3 Maximum entropy (ME) solution for example problem

(v) $s_{1,3} = x_{101} + x_{111} = 0.03$
(vi) $s_\emptyset = \sum_{b \in \{0,1\}^3} x_b = 1$

The task of selectivity estimation is to now compute a solution for all atoms $x_b, b \in \{0,1\}^3$ that maximizes the entropy function $-\sum_{b \in \{0,1\}^3} x_b \log x_b$ and satisfies the above six constraints. Any desired selectivity s_X can then be computed from the x_b values as indicated previously.

Figure 3 gives the results obtained when solving this constrained optimization problem. For instance, in this ME solution, we obtain the selectivity estimate $s_{1,2,3} = x_{111} = 0.015$ and $s_{2,3} = x_{111} + x_{011} = 0.05167$.

In the next section, we describe an algorithm to compute this result efficiently for an arbitrary collection P of simple predicates and an arbitrary knowledge set T.

Fig. 4 Iterative-scaling algorithm

Input:
A knowledge set $T \subset 2^N$ ($\emptyset \in T$) and selectivities $\{s_Y: Y \in T\}$

Output:
An approximate ME solution x_b for all atoms $b \in \{0, 1\}^n$

```
1.   FOR X ∈ T:
2.     z_X := 1; // ENDFOR X
3.   ε := 10^-6;
4.   REPEAT
5.     Δz := 0;
6.     FOR Y ∈ T:
7.       sum := 0;
8.       FOR b ∈ C(Y)
9.         product := 1
10.        FOR X ∈ P(b,T)\Y
11.          product *= z_X; // ENDFOR X
12.        sum += product; // ENDFOR b
13.      z_Y^0 := z_Y;
14.      z_Y := s_Y * e / sum;
15.      Δz += |( z_Y - z_Y^0 )/ z_Y^0 |; // ENDFOR Y
16.  UNTIL Δz < ε
```

3.3 The iterative scaling algorithm

To solve the constrained optimization problem in its general form, we first use the method of Lagrange multipliers to obtain a system of optimality equations. Since the entropy function is concave, a solution of this system yields the unique solution of the optimization problem [11]. We associate a Lagrange multiplier λ_X with each known s_X. This includes $\lambda_\emptyset$, a multiplier associated with the constraint $s_\emptyset = 1$. The Lagrangian for $\boldsymbol{x} = \{x_b : b \in \{0,1\}^n\}$ and $\boldsymbol{\lambda} = \{\lambda_X : X \chi T\}$ is given by

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = \sum_{b \in \{0,1\}^n} x_b \log x_b - \sum_{X \in T} \lambda_X \left(\sum_{b \in C(X)} x_b - s_X \right) \quad (3.3)$$

Taking derivatives with respect to the x_b and setting them equal to zero yields the optimality equations

$$\log x_b + 1 = \sum_{X \in P(b,T)} \lambda_X, \quad b \in \{0,1\}^n. \quad (3.4)$$

Setting $z_X = \mathrm{e}^{\lambda_X}$, we obtain the equivalent exponential form

$$x_b = \frac{1}{\mathrm{e}} \prod_{X \in P(b,T)} z_X. \quad (3.5)$$

Using Eq. (3.5) to eliminate x_b in Eq. (3.2), we find that

$$\sum_{b \in C(Y)} \prod_{X \in P(b,T)} z_X = s_Y * \mathrm{e}, \quad Y \in T. \quad (3.6)$$

The ME solution is thus obtained by determining the set of (exponentiated) Lagrange multipliers $Z = \{z_Y : Y \in T\}$ that satisfies Eq. (3.6) and then computing the x_b values using Eq. (3.5); these latter values determine s_X for all $X \in 2^N$.

It is usually impossible to compute Z analytically. We therefore use a variant of the iterative scaling algorithm [11] to efficiently obtain an approximation to Z, and hence an approximate ME solution. The algorithm produces a set of approximate multipliers by a process of iterative refinement. During an iteration step, the algorithm sequentially selects each element in $z_Y \in Z$ and adjusts the value of z_Y (while keeping the other multipliers constant) so that the particular equality in Eq. (3.6) that corresponds to s_Y is satisfied. More specifically, the algorithm sets

$$z_Y = \frac{s_Y * \mathrm{e}}{\sum_{b \in C(Y)} \prod_{X \in P(b,T) \setminus \{Y\}} z_X} \quad (3.7)$$

for each $Y \in T$. We call Eq. (3.7) the *iterative scaling equation*; it is obtained from Eq. (3.6) by observing that z_Y occurs in every summand on the left side, so that it can be factored out of the sum. In general, this adjustment will cause the remaining equalities in Eq. (3.6) to be violated, which is why the solution is only approximate. It can be shown, however, that the sequence of iterated solutions converges to Z in the limit [11]. The algorithm therefore terminates when Δz becomes negligible, where Δz denotes the absolute sum of the relative changes in the Lagrange-multiplier values during the current iteration. Figure 4 below shows the complete scaling algorithm.

3.4 A priori elimination of zero atoms

In practice, it is often the case that the constraints in Eq. (3.2) imply that the selectivity of certain atoms must be zero in any feasible solution. For instance, if $p_1 \Rightarrow p_2$, so that $s_1 = s_{1,2}$, then $x_{100} = x_{101} = 0$ in any solution $\boldsymbol{x}$. These *zero atoms* destabilize the iterative scaling algorithm. Indeed, although the entropy function $H(\boldsymbol{x})$ is well defined when $x_b = 0$ (because $\lim_{x\to 0} x \log x = 0$ by L'Hospital's rule), the derivative is infinite:

$$\partial H/\partial x_b = -1 - \log x_b = +\infty.$$

It follows that there does not exist a finite solution to the optimality equation (Eq. 3.4), and the iterative scaling algorithm fails to converge. All zero atoms must therefore be identified and explicitly removed from the both the objective function and the constraints in the ME optimization problem prior to execution of iterative scaling.

Identifying zero atoms is nontrivial in general, because the reasoning involved can be arbitrarily complex. In the following, we provide both an exact (but expensive) iterative method and a quick approximate method for automatically detecting zero atoms.

3.4.1 Iterative detection of zero atoms

The exact iterative method for detecting zero atoms is based on the following fact. If, for a given atom b, there exists a feasible solution x that satisfies all constraints in Eq. (3.2) and in which $x_b > 0$, then x_b is also positive in the ME solution. Intuitively, if $x_b = 0$ and $x_{b'} > 0$ in some feasible solution, then transferring probability mass by decreasing $x_{b'}$ and increasing x_b will make the distribution more uniform, and hence increase the entropy; some such transfer is possible because the existence of x as above shows that having $x_b > 0$ will not in general violate the constraints.

The initial set A_0 of candidate zero atoms contains all of the atoms: $A_0 = \{0,1\}^n$. In the first iteration we set $i = 0$ and solve the linear program (LP)

$$\underset{x_b \,|\, b\in 0,1^n}{\text{maximize}} \sum_{b\in A_i} x_b \text{ subject to Eq. (3.2) and to}$$
$$x_b \in [0,1], b \in 0,1^n \tag{3.8}$$

using, e.g., the simplex method. The idea is that a solution to the above problem will make each x_b as large as possible while satisfying the feasibility conditions. Any x_b that is equal to 0 in the LP solution is therefore likely to be a zero atom. However, an x_b that is equal to 0 in the LP solution is not *guaranteed* to be a zero atom; it could be the case that $x_b = 0$ in the computed optimal LP solution but not in other possible optimal LP solutions. The algorithm therefore refines the set of candidates as $A_1 = A_0 \backslash \{b | x_b \neq 0\}$, sets $i = 1$, and solves the resulting LP in Eq. (3.8). The algorithm proceeds in this manner, iterating until either (i) $A_i = \emptyset$ or (ii) $x_b = 0$ for every $b \epsilon A_i$. In the former case, each candidate atom has been shown to have positive probability in at least one feasible solution, and hence, as discussed previously, must be positive in the ME solution; we therefore conclude that there are no zero atoms. In the latter case, we see that the objective-function value for the solution to the LP in Eq. (3.8) is 0, and therefore each atom $b \epsilon A_i$ must have $x_b = 0$ in any feasible solution – otherwise, the optimal objective-function value would have been positive – and hence in the ME solution. Thus A_i is precisely the set of zero atoms.

3.4.2 Example: iterative zero detection

Suppose that $N = \{1,2,3\}, T = \{\{1\},\{2\},\{3\},\{1,2\},\{1,3\}, \{2,3\},\emptyset\}, s_1 = 0.23, s_2 = 0.01, s_3 = 0.015, s_{1,2} = 0.01, s_{1,3} = 0.01, s_{2,3} = 0.01$, and $s_\emptyset = 1$. The constraints in the ME optimization problem are

$$\begin{aligned}
s_1 &= x_{100} + x_{110} + x_{101} + x_{111} &&= 0.23\\
s_2 &= x_{010} + x_{011} + x_{110} + x_{111} &&= 0.01\\
s_3 &= x_{001} + x_{011} + x_{101} + x_{111} &&= 0.015\\
s_{1,2} &= x_{110} + x_{111} &&= 0.01\\
s_{1,3} &= x_{101} + x_{111} &&= 0.01\\
s_{2,3} &= x_{011} + x_{111} &&= 0.01\\
s_\emptyset &= \textstyle\sum_{b\in\{0,1\}^3} x_b &&= 1
\end{aligned}$$

Setting $A_0 = \{0,1\}^3$ and solving the LP in Eq. (3.8), we obtain a solution in which $x_{000}, x_{100}, x_{001}$, and x_{111} are nonzero. For the next iteration we therefore set $A_1 = \{010, 110, 101, 011\}$. Solving the resulting LP, we find that $x_{010} = x_{110} = x_{101} = x_{011} = 0$, so that A_2 is the set of zero atoms.

The foregoing method is guaranteed to discover every zero atom and not misclassify any nonzero atoms as zero atoms. Unfortunately, due to its iterative nature, the algorithm is so computationally expensive as to be impractical in real-world applications, even with a highly sophisticated LP solver.

3.4.3 Zero detection via approximation

In our actual implementation of the ME method, we employ an approximate detection technique that offers a reasonable trade-off between accuracy and execution time. The idea is to rewrite the selectivity of each atom as the sum of two new variables: $x_b = v_b + w_b$. We now

solve the following LP:

$$\underset{v_b, w_b}{\text{maximize}} \sum_{b \in \{0,1\}^n} v_b$$

subject to

$$\sum_{b \in C(X)} v_b + w_b = s_X, \quad X \in T$$

$$0 \leqslant w_b \leqslant 1 \text{ and } 0 \leqslant v_b \leqslant \varepsilon, \quad b \in \{0,1\}^n$$

where ε is a small value; in our implementation we set $\varepsilon = 0.0001$. After solving this LP, an atom b is considered to be a zero atom if and only if $w_b = v_b = 0$. The idea is that setting $x_b = 0$ requires setting $v_b = 0$, which significantly decreases the objective-function value because of the ε upper bound on all of the selectivities. Thus only "true" zero atoms are likely to be identified by the solution to the LP. We include the w_b variables because they provide the "padding" needed to ensure that the original constraints in Eq. (3.2) are satisfied. This algorithm finds all of the zero atoms, but is approximate in that it can mislabel some nonzero atoms as zero atoms. In practice, we found that such mislabelings are infrequent, so that the quality of the ultimate ME solution is good.

3.4.4 Example: zero detection via approximation

Again suppose that $N = \{1,2,3\}, T = \{\{1\},\{2\},\{3\}, \{1,2\},\{1,3\},\{2,3\},\emptyset\}, s_1 = 0.23, s_2 = 0.01, s_3 = 0.015, s_{1,2} = 0.01, s_{1,3} = 0.01, s_{2,3} = 0.01$, and $s_\emptyset = 1$. The constraints in the resulting LP are

$$\begin{aligned}
s_1 &= v_{100} + w_{100} + v_{110} + w_{110} + v_{101} + w_{101} + v_{111} \\
&\quad + w_{111} = 0.23 \\
s_2 &= v_{010} + w_{010} + v_{011} + w_{011} + v_{110} + w_{110} + v_{111} \\
&\quad + w_{111} = 0.01 \\
s_3 &= v_{001} + w_{001} + v_{011} + w_{011} + v_{101} + w_{101} + v_{111} \\
&\quad + w_{111} = 0.015 \\
s_{1,2} &= v_{110} + w_{110} + v_{111} + w_{111} = 0.01 \\
s_{1,3} &= v_{101} + w_{101} + v_{111} + w_{111} = 0.01 \\
s_{2,3} &= v_{011} + w_{011} + v_{111} + w_{111} = 0.01 \\
s_\emptyset &= \textstyle\sum_{b \in \{0,1\}^3} v_b + w_b = 1
\end{aligned}$$

In the solution returned by the simplex algorithm, the following variables are equal to zero: $v_{010}, w_{010}, v_{110}, w_{110}, v_{101}, w_{101}, v_{011}, w_{011}$. We therefore take the set of zero atoms as $\{010, 110, 101, 011\}$. Observe that this solution coincides with the solution returned by the exact iterative algorithm.

4 Dealing with inconsistencies

A significant problem encountered when applying ME techniques in a real-world database management system is that the given selectivities $\{s_X : X \epsilon T\}$ might not be mutually consistent. For example, the selectivities $s_1 = 0.1$ and $s_{1,2} = 0.15$ are inconsistent, because they violate the obvious requirement that $s_X \geq s_Y$ whenever $X \subseteq Y$. In the presence of inconsistent statistics, there will not exist *any* solutions to the constrained optimization problem in Eqs. (3.1) and (3.2), much less an ME solution, and the iterative scaling algorithm, if applied blindly, will fail to converge.

There are two typical causes of inconsistent statistics. First, the single-column statistics in T are often taken from the system catalog directly or derived by the optimizer from catalog statistics. Because collection of accurate statistics can be a highly cost-intensive process, commercial database systems typically compute catalog statistics using approximate methods such as random sampling or probabilistic counting. Even when the catalog statistics are exact, the selectivity estimates computed by the optimizer from these statistics often incorporate inaccurate uniformity assumptions or use rough histogram approximations based on a small number of known quantiles. The other cause of inconsistent knowledge is the fact that different statistics may be collected at different points in time, and the underlying data can change in between collection epochs. This problem is particularly acute in modern systems, where some of the MVS used by the optimizer might be based on query feedback or materialized statistical views.

We therefore need a scheme that will adjust the input selectivities to obtain a set of satisfiable constraints, prior to execution of the iterative-scaling algorithm. Ideally, the adjustments should be as small as possible to avoid introducing unnecessary bias into the selectivity estimates. Devising such a scheme is not straightforward, however. First, there exist inconsistencies that are not as obvious as the one described above. Furthermore, simply applying ad hoc adjustments, e.g. setting $s_1 = 0.15$ in the foregoing example, might introduce new inconsistencies.

We solve the inconsistency problem using the following technique. First, associate two "slack" variables a_X^+ and a_X^- with each of the original constraints in Eq. (3.2), except for the constraint corresponding to $s_\emptyset$; this latter constraint ensures that the atom selectivities sum to 1, and therefore must not be modified. Then solve the following LP:

$$\underset{a_X^+, a_X^-, x_b}{\text{minimize}} \sum_{X \in T \setminus \emptyset} a_X^+ + a_X^-$$

subject to

$$\sum_{b\in C(X)} x_b + a_X^+ - a_X^- = s_X, \quad X \in T\backslash\{\emptyset\}, \quad \sum_{b\in\{0,1\}^n} x_b = 1,$$

$$a_X^+ \geq 0,\ a_X^- \geq 0,$$
$$0 \leq s_X - a_X^+ + a_X^- \leq 1 \tag{4.1}$$

The slack variables represent either positive or negative adjustments to the selectivities needed to ensure the existence of a feasible solution. In this connection, note that, in the optimal solution to the LP, at most one of the two slack variables for a constraint will be nonzero; indeed, for a specified value $a_X^+ - a_X^-$ of the total adjustment, any solution that has a nonzero value for both slack variables will yield a higher value of the objective function than a solution with only a single nonzero slack variable. The presence of a nonzero slack variable both signals the presence of an inconsistency and indicates how to obtain consistency, namely, by setting $s_X^* := s_X - a_X^+ + a_X^-$. The constraint in Eq. (4.1) ensures that the adjusted selectivities lie in the range [0,1]. By taking the objective function as the sum of the slack variables, we ensure that the adjustments to the constraints are as small as possible.

Our approach offers interesting possibilities for enhancements in future implementations. Typically, for example, some statistics are more reliable than others, and it is possible to reflect this fact by weighting the terms in the objective function. A large weighting coefficient for the slack variables a_X^+ and a_X^- means that the corresponding selectivity s_X is reliable; this selectivity is relatively unlikely to be adjusted because an adjustment would incur a relatively large penalty in the objective function. By means of this device, unreliable statistics are more likely to be subject to adjustments than reliable ones.

Note that a newly consistent set of constraints may have zero atoms. In our complete algorithm, we first obtain a consistent starting set of constraints using the techniques described in this section and then detect and remove zero atoms as described in Sect. 3.4.

4.1 Example: inconsistency detection and removal

Suppose that $N = \{1,2\}, T = \{\{1\},\{2\},\{1,2\},\emptyset\}, s_1 = 0.99, s_2 = 0.99, s_{1,2} = 0.90$, and $s_\emptyset = 1$. The constraints for the LP are given by

$$\begin{aligned} s_1 &= x_{10} + x_{11} + a_1^+ - a_1^- &&= 0.99\\ s_2 &= x_{01} + x_{11} + a_2^+ - a_2^- &&= 0.99\\ s_{1,2} &= x_{11} + a_{1,2}^+ - a_{1,2}^- &&= 0.90\\ s_\emptyset &= x_{00} + x_{10} + x_{01} + x_{11} &&= 1 \end{aligned}$$

Minimizing the sum over all slack variables yields the following solution:

$$\begin{aligned} a_1^+ &= a_1^- &&= 0\\ a_2^+ &= a_2^- &&= 0\\ a_{1,2}^+ &&&= 0\\ a_{1,2}^- &&&= 0.08 \end{aligned}$$

Although it may not be readily apparent, the constraint set T contains an inconsistency, because $a_{1,2}^- = 0.08$. Applying the resulting adjustment, we obtain the set of consistent selectivities $s_1^* = 0.99, s_2^* = 0.99$, and $s_{1,2}^* = 0.98$.

5 Improving scalability by partitioning

Even though iterative scaling has a complexity of $O(|T|2^{|P|})$, we can often avoid executing the algorithm on the full predicate set P. The idea is to compute the ME solution by partitioning P into several disjoint subsets and executing the scaling algorithm on each subset independently. This approach reduces the complexity substantially, and makes the iterative scaling algorithm feasible even for extremely complex queries with large sets of predicates. In the following, we discuss some exact and approximate partitioning strategies. In our discussion, we assume that, as is usual in practice, individual selectivities $s_1, s_2, \ldots, s_n$ of single predicates $p_1, p_2, \ldots, p_n$ are always available.

5.1 Partitioning

Suppose that we can split $N = \{1,2,\ldots,n\}$ into nonempty disjoint subsets $N_1, N_2, \ldots, N_k$, such that for each $X \in T$ we have $X \subseteq N_i$ for some $i \in \{1,2,\ldots,k\}$. Partition P and T accordingly by setting

$$P_i = \{p_j | j \in N_i\} \quad \text{and} \quad T_i = \{X \in T | X \subseteq N_i\}$$

for $1 \leq i \leq k$.[6] We claim that the ME solution for (P,T) can be obtained by first using iterative scaling to compute the ME solutions for $(P_1,T_1),(P_2,T_2),\ldots,(P_k,T_k)$ and then using the independence assumption to

[6] Note that the T_i sets are not completely disjoint, because they all contain $\emptyset$.

combine selectivity estimates for predicates in different partitions; i.e., we compute

$$s_X = \prod_{i \in \{1,2,\ldots,k\}} s_{X \cap N_i}$$

for $X \in 2^N$. See Appendix B for a formal justification of this claim.

For example, when $N = \{1,2,3,4\}$ and $T = \{\{1\},\{1,2\},\{3\},\{3,4\},\emptyset\}$, we can take $N_1 = \{1,2\}, N_2 = \{3,4\}, T_1 = \{\{1\},\{1,2\},\emptyset\}$, and $T_2 = \{\{3\},\{3,4\},\emptyset\}$. We can then, e.g., compute s_2 by obtaining the ME solution for (P_1, T_1), compute s_4 by obtaining the ME solution for (P_2, T_2), and finally compute $s_{2,4}$ using the independence assumption as $s_{2,4} = s_2 * s_4$.

Partitioning reduces the computational complexity from $O(|T|2^{|N|})$ to $O(T_1 2^{|N_1|} + \cdots + T_k 2^{|N_k|})$, making iterative scaling feasible even for large sets of predicates N. For example, when $N = \{1,2,\ldots,12\}, |T| = 25, k = 4, |N_i| = 3$ and $|T_i| = 7$, the use of partitioning reduces the complexity by two orders of magnitude.

In practice, partitioning can reduce the computational complexity even more than is indicated above, by avoiding execution of the iterative scaling algorithm altogether for specified partitions (P_i, T_i). In the above example, for instance, we can compute the selectivity $s_{1,2,4}$ as $s_{1,2,4} = s_{1,2} * s_4$; because $s_{1,2}$ is known a priori. Thus we need only run iterative scaling on (P_2, T_2) and not on (P_1, T_1). Similarly, if a partition contains only a few predicates, it may be possible to compute the desired ME selectivity analytically, without requiring iterative scaling. For example, suppose that $P_i = \{p_1, p_2, p_3\}, T_i = \{\{1\},\{2\},\{3\},\{1,2\},\{2,3\},\emptyset\}$, and we need to compute $s_{1,2,3}$. It can be shown (see Appendix C) that $s_{1,2,3} = s_{1,2} * s_{2,3} / s_2$, so that no iterative scaling is needed. In general, it may be possible to maintain a "library" of such identities, and perform a quick syntactic analysis at selectivity-estimation time to see if any of the identities apply.

5.2 Forced partitioning

As shown by the experiments in Sect. 8, iterative scaling cannot compute the ME solution in sub-second time if the cardinality of N is too large. In order to guarantee sub-second computation time, we need to ensure that each partition N_i as in Sect. 5.1 has cardinality at most μ, where μ is a constant that depends on the computer hardware and system load.[7]

[7] In practice, the constant μ depends primarily on the CPU speed of the computer. On a single-user laptop using an Intel Pentium(R) III Mobile CPU 1,133 MHz with 512 MB RAM, we found that $\mu = 8$.

Unfortunately, it is not always possible to partition N as in Sect. 5.1 into subsets of cardinality at most μ for a given set T. For example, suppose that $N = \{1,2,3,4,5,6\}$ and $T = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{1,2\},\{2,3\},\{3,4\},\{4,5\},\{5,6\}\}$. Then N cannot be partitioned for any $\mu < 6$ since, for any partitioning of N into N_1 and N_2, there exists at least one $X \in T$ such that $X \cap N_1 \neq \emptyset$ and $X \cap N_2 \neq \emptyset$.

In such cases, we must remove elements from T in order to *force* a partitioning in which $|N_i| \leq \mu$. However, this forced partitioning will potentially degrade the quality of the ME solution, since we are discarding information. Ideally, a forced-partitioning algorithm should remove those elements that have the least impact on the ME solution. Such an algorithm, however, would be very computationally expensive, because it would have to consider the interaction of every constraint with every other constraint. In the following, we describe a pragmatic algorithm with complexity $O(|T| \log |T|)$ that ignores interactions and greedily removes elements from T with the goal of minimizing the impact on the ME solution.

When generating partitions, we keep track of the cardinality c_i of each partition N_i. At the beginning we start with $|N|$ partitions, $N_i = \{i\}$ and $c_i = 1$ for every partition. We then iteratively merge partitions based on each element $X \in T$, i.e,

$$N_i = \begin{cases} \bigcup_{j | X \cap N_j \neq \emptyset} N_j & \text{if } i = \min(X); \\ \emptyset & \text{otherwise,} \end{cases}$$

which reduces the number of partitions, but increases c_i. We only add an element $X \in T$ to the knowledge set T_i of partition N_i if afterwards the constraint $c_i \leq \mu$ will be satisfied; if the constraint will be violated, we ignore X, thereby removing X from T.

In order to have as little impact as possible on the overall ME solution, we need to avoid discarding those elements $X \epsilon T$ that correspond to knowledge about the largest deviations from independence. We can achieve this goal by processing elements in decreasing order of Δ_X, where $\Delta_X = \max(s_X / \prod_{i \in X} s_i, \prod_{i \in X} s_i / s_X)$. The quantity Δ_X measures the degree to which the corresponding s_X constraint in Eq. (3.2) forces the ME solution away from independence. Figure 5 summarizes the algorithm for forced partitioning. Note that an efficient implementation stores T as list sorted according to Δ_X and thus has a complexity of $O(|T| \log |T|)$.

5.2.1 Example: forced partitioning

Let $N = \{1,2,3,4,5,6\}$ and $T = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{1,2\},\{2,3\},\{3,4\},\{4,5\},\{5,6\}\}$. Also suppose that

Input:
A knowledge set $T \subset 2^N$ ($\emptyset \in T$) for the predicate index set N
and selectivities $\{s_Y: Y \in T\}$

Output:
A partitioning of N and T into $N_1, \ldots, N_k$ and $T_1, \ldots, T_k$ such that $|N_i| \leq \mu$ for each i

```
1   T := T \ {{1},{2},...,{n}}
2   FOR i := 1 to |N|
3       N_i := {i} and T_i := {{i}}
4   // ENDFOR i
5   WHILE T ≠ ∅
6       Select X ∈ T such that Δ_X ≥ Δ_Y for all Y ∈ T \ {X}
7       T := T \ {X}
8       IF |⋃_{j|X∩N_j≠∅} N_j| ≤ μ
9           FOR i ∈ X
10              IF i = min (X)
11                  N_i := ⋃_{j|X∩N_j≠∅} N_j  and  T_i := (⋃_{j|X∩N_j≠∅} T_j) ∪ {X}
12              ELSE
13                  N_i := ∅
14              // ENDIF
15          // ENDFOR i
16      // ENDIF
17  // ENDWHILE
```

Fig. 5 Forced-partitioning algorithm

$s_1 = 0.1, s_2 = 0.2, s_3 = 0.3, s_4 = 0.4, s_5 = 0.5, s_6 = 0.6$ and $s_{1,2} = 0.1, s_{2,3} = 0.2, s_{3,4} = 0.3, s_{4,5} = 0.4, s_{5,6} = 0.5$. It follows that $\Delta_{1,2} = 5, \Delta_{2,3} = 3.33, \Delta_{3,4} = 2.5, \Delta_{4,5} = 2$, and $\Delta_{5,6} = 1.67$. With $\mu = 3$, we therefore obtain $k = 2$, $N_1 = \{1,2,3\}, T_1 = \{\{1\},\{2\},\{3\},\{1,2\},\{2,3\}\}, N_2 = \{4,5,6\}$, and $T_2 = \{\{4\},\{5\},\{6\},\{4,5\},\{5,6\}\}$. The element {3,4} has been dropped in order to enable the partitioning of N into N_1 and N_2. Of all the elements that prevent partitions of size 3 from being constructed, the element {3,4} has the smallest impact on the ME solution.

6 Predicates over multiple tables

The discussion so far has focused on sets of predicates such that all of the predicates in the set refer to a single table. Our techniques extend naturally to sets of predicates that refer to two or more tables, i.e., sets containing both local and join predicates. The idea is to generalize the notion of selectivity in the usual way: if a set P of predicates refers to tables $R_1, R_2, \ldots, R_k$, then the selectivity of each predicate $p \epsilon P$ is defined as the fraction of elements of the Cartesian product $R_1 \times R_2 \times \cdots \times R_k$ that satisfy p. Observe that the selectivity of a *local* predicate (i.e., a predicate that refers only to the columns in a specified table R) coincides with the single-table selectivity defined previously. We can now proceed precisely as in previous sections.

For the case of, e.g., $k = 2$ tables and in the absence of any other information, the ME estimate of the selectivity of a join predicate is 0.5, since any pair of rows is as likely to join as to not join. In practice, enough information is usually available so that the estimated selectivity is much lower than 0.5. In one common scenario, for example, a join predicate p_i is known to be a foreign-key join, with table R_1 containing the key column and table R_2 containing the foreign-key column. Thus the number of elements in the Cartesian product $R_1 \times R_2$ that satisfy p_i is $|R_2|$. In this case, we add the element $X = \{i\}$ to the knowledge set T, with corresponding selectivity $s_x = 1/|R_1|$.

7 Integration into the DB2 optimizer

The existing DB2 UDB optimizer precomputes selectivities from single-column statistics and MVS prior to plan enumeration, costing, and selection. When

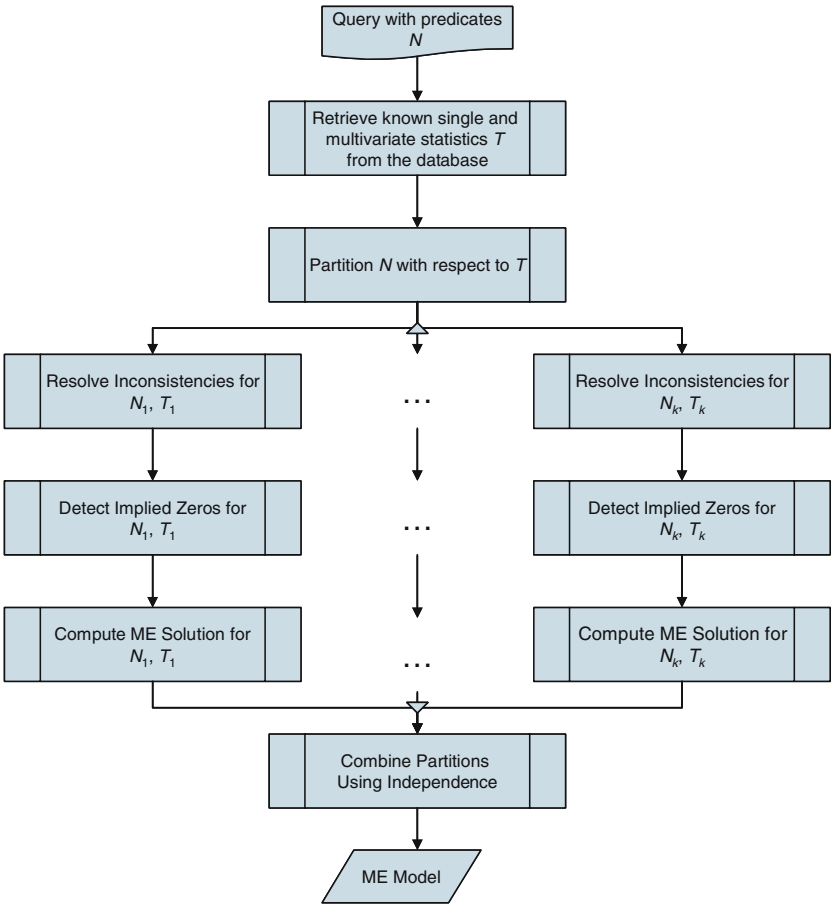

Fig. 6 Precomputation phase for computing ME model (k partitions)

costing a plan that involves a conjunctive predicate p, the optimizer estimates p's selectivity by using a precomputed selectivity if available, or else combining several precomputed selectivities using the ad hoc method outlined in Appendix A. The optimizer takes precautions to avoid inconsistencies by keeping track of how a missing estimate for a predicate p was computed when it was first requested by a subplan during join enumeration. Subsequent requests for p's selectivity estimate will use this recorded information to ensure that the estimate is always derived in the same way. Although this approach avoids inconsistencies, it discards useful information. Moreover, the plans chosen by the optimizer depend on how the selectivities are derived initially, which in turn depends on the order in which the selectivity estimates are requested during optimization; this order is completely arbitrary, leading to an arbitrary bias in the choice of plans.

The ME model can be easily integrated into the DB2 UDB optimizer, in that changes to the code are essentially isolated within a single subroutine. For our prototype implementation, we extended the precomputation phase to not only compute selectivities based on statistical information, but also to precompute all missing selectivities, using the techniques for partitioning, inconsistency resolution, zero detection, scaling, and combination described earlier. This precomputation eliminates the need to use the heuristic given in Appendix A for costing. It also eliminates the need to keep track of how selectivities were combined in order to avoid inconsistencies during estimation, because the ME estimate for a given selectivity is unique. The DB2 optimizer uses these precomputed selectivity estimates during dynamic programming to compute the cardinalities of each partial QEP it considers.

Figure 6 shows the overall architecture of the precomputation phase as implemented in our DB2 prototype.

Our extensions to the cardinality model enable the optimizer to use all available statistics in a consistent way, for all plans in the plan space. This improved knowledge results in better query plans and improved query execution times, as shown experimentally in the next section. Our modifications also simplify the query optimizer's logic, as consistency checks and record-keeping are no longer necessary during cardinality estimation.

8 Experimental evaluation

In this section we describe a set of experiments that we ran in order to evaluate the quality of the estimates produced by the ME approach, as well as the computational costs of the algorithms. We focused primarily on single-table queries, using all available information about both single and joint column frequencies.

The workload used in our experiments was derived from a real-world 1 GB database and collection of 200 queries, both obtained from a department of motor vehicles (DMV) data center. The workload involves four tables: OWNER, CAR, DEMOGRAPHICS, and ACCIDENTS. For all of our experiments except those in Sect. 8.3, we focused on the portion of each query that references the CAR table. Figure 7 shows the schema of the CAR table.

The CAR table has a base cardinality of 143,309 rows. The table also contains strong correlations between MAKE, MODEL, COLOR, and YEAR; multivariate statistics are required on at least a subset of these columns in order to obtain reasonable cardinality estimates. We focused on the three-way correlation between MAKE, MODEL, and COLOR, as this is the correlation most frequently encountered in actual queries.

8.1 Quality of the estimates

We first assessed the quality of the ME estimates, comparing our new methodology to the state-of-the-art (SOTA) estimation method in DB2 UDB v8.2 as described in Appendix A. Specifically, we ran experiments to estimate the selectivities of queries of the form "SELECT * FROM CAR WHERE p_1 AND p_2 AND p_3" with

p_1: CAR.MAKE = :literal1
p_2: CAR.MODEL = :literal2
p_3: CAR.COLOR = :literal3

We used a workload of 200 queries having different values for each of the three literals in each query instance. From the base statistics, the optimizer always knew the selectivities s_1, s_2, and s_3, of the simple predicates, i.e., $\{\{1\},\{2\},\{3\}\} \subseteq T$. Additionally, we created MVS to make the optimizer aware of several of the correlations between the three columns referenced by the predicates. We considered five major cases, each representing different available knowledge about joint column distributions. We label each case as "$f.c$," where f is the maximum number of BFs in any of the known conjuncts, and c is the number of such maximally sized conjuncts:

id	Integer (Primary Key)
ownerid	Integer (Foreign Key)
year	Integer
make	Char (20)
model	Char (20)
color	Char (20)

Fig. 7 Schema of the CAR table (DMV data-base)

Case 1.3 Only the selectivities of the single predicates (marginals) s_1, s_2, s_3 are known. In this special case, the ME solution can be computed analytically and is identical to the SOTA solution (Example c in Appendix A).

Case 2.1 Besides the marginals, the selectivity of one conjunctive predicate consisting of two BFs is known. Since our query comprises three BFs, we distinguish three possible subcases. In Case 2.1a the selectivity $s_{1,2}$ for the conjunct on MAKE and MODEL is known, in Case 2.1b the selectivity $s_{1,3}$ for the conjunct on MAKE and COLOR is known, and in Case 2.1c the selectivity $s_{2,3}$ for the conjunct on MODEL and COLOR is known. The ME estimate for $s_{1,2,3}$ is just the product $s_{1,2} * s_3$ for Case 2.1a, and analogously for the other cases. The ME estimates coincide with the SOTA estimates.

Case 2.2 Besides the marginals, the selectivity of two conjunctive predicates consisting of two BFs is known. In this case, we again distinguish three subcases, depending on which conjunctive predicates are known:

- 2.2a: $s_{1,2}, s_{1,3}$ for (MAKE, MODEL), (MAKE, COLOR)
- 2.2b: $s_{1,2}, s_{2,3}$ for (MAKE, MODEL), (MODEL, COLOR)
- 2.2c: $s_{1,3}, s_{2,3}$ for (MAKE, COLOR), (MODEL, COLOR)

The ME solution can again be computed analytically, as discussed in Appendix C: for Case 2.2a, $s_{1,2,3} = s_{1,2} * s_{1,3}/s_1$, and analogously for Cases 2.2b and 2.2c. Note that this solution does *not* correspond to the SOTA computation.

Case 2.3 Selectivities for all three conjunctive predicates consisting of two BFs are known. In this case, the ME solution can no longer be computed analytically, and the iterative-scaling algorithm must be applied. Note that the ME result in this case again does not correspond to the SOTA estimate.

Case 3.1 The selectivity $s_{1,2,3}$ of the overall conjunctive predicate consisting of three BFs is known. Since this is the selectivity that we seek, the ME algorithm simply returns $s_{1,2,3}$ as the answer. The SOTA estimate (Example a in Appendix A) is identical.

The experiments reported in this section do not use forced partitioning. Preliminary experiments indicated

that for a value of $\mu = 8$, use of forced partitioning had a negligible impact on the quality of the ME solution.

8.1.1 Using all information

We ran a workload of 200 queries; for each query we computed the absolute estimation error $E = |\hat{c} - c|$, where $\hat{c}$ is the ME estimate of the cardinality (i.e., number of rows returned by the query) and c is the true cardinality. The box plots in Fig. 8 summarize the absolute estimation errors of the ME estimates in each of the cases described above. The bottom of a box gives the first quartile (twenty-fifth percentile) of the error over the 200 queries, the horizontal line inside a box and the number to the right of the line give the value of the median error, the top of a box gives the third quartile (seventy-fifth percentile) of the error, and the endpoints of the vertical lines that extend above and below the box give the maximum and minimum error. The estimation error for Case 1.3 is considerable – with a median value of 788 and a maximum value of almost 10,000 – because the estimates use the independence assumption, thereby ignoring the strong correlations between the columns. Using MVS on even a single pair of columns reduces both the maximum and median error by an order of magnitude. Observe that the correlation between MAKE and MODEL is apparently stronger than the correlation between any other column pair, since the use of MVS for (MAKE, MODEL) yields the smallest error (both median and maximum) for Case 2.1.

If MVS are available for two of the column pairs, then the error is reduced even further. When one of those pairs comprises the strongly correlated MAKE and MODEL columns, both the median and maximum error are reduced by two orders of magnitude over Case 1.3 (known marginals only). Note that when we only have MVS for the not-so-strongly correlated (MAKE, COLOR) and (MODEL, COLOR) column pairs (Case 2.2c), the overall error reduction – while greater than when MVS are available for only one column pair – is not as pronounced as when MVS are available on (MAKE, MODEL). Both the median and seventy-fifth percentile of the absolute error are reduced even further in Case 2.3, where MVS are available for all three column pairs. In Case 3.1, perfect knowledge results in zero error.

Overall, the experiment shows that our ME estimation method has the desired property of reducing the absolute estimation error as more knowledge becomes available. We conducted further experiments for queries involving more than three predicates, with similar results.

8.1.2 Improvement over the state of the art

Figure 9 contrasts the absolute estimation error of our ME approach with the SOTA method for cardinality estimation described in Appendix A. Note that the estimation errors of SOTA and ME are identical for the Cases 1.3, 2.1, and 3.1, because the ME principle reduces to the assumptions of uniformity and independence used by SOTA. However, as soon as more than one selectivity for a two-conjunct predicate becomes available, SOTA and ME return different results. In these cases, SOTA cannot use the additional information to improve estimates, and thus is reduced to using information about only a single two-conjunct predicate. In contrast, ME exploits all available information, thereby yielding lower median and maximum errors than SOTA in every case. The improved performance of ME relative to SOTA is

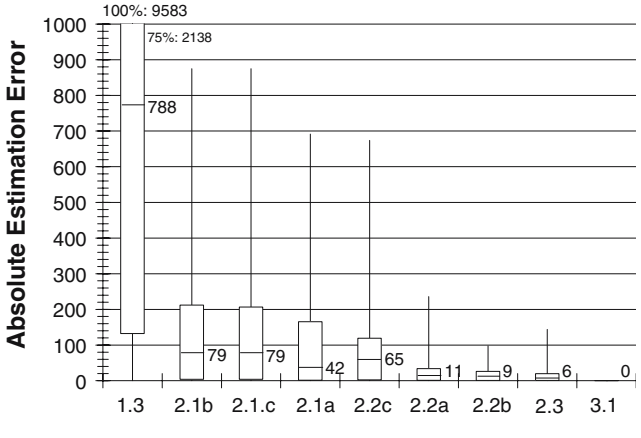

Fig. 8 Box plots of absolute estimation error for various knowledge sets (200 queries)

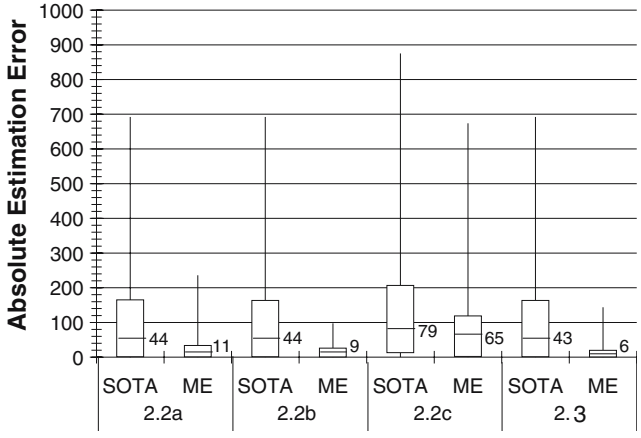

Fig. 9 Comparison of ME to SOTA (200 queries)

especially striking when one of the available selectivities is on the strongly correlated (MAKE, MODEL) column pair (Cases 2.2a and 2.2b) and when selectivities are available on all three column pairs (Case 2.3). Experiments for $|P| > 3$ (not reported here) show even more dramatic improvements of ME over SOTA.

8.1.3 Backward Computation

We can also use the iterative scaling algorithm for backward computation, i.e., using knowledge of s_X to compute s_Y, where $Y \subset X$. Thus if we have three-way selectivities on (MAKE, MODEL, COLOR), then the ME approach allows us exploit this information in order to handle the two-column correlations between MAKE and MODEL, MAKE and COLOR, or MODEL and COLOR without resorting to erroneous independence assumptions. Figure 10 displays box plots of the absolute estimation error for the SOTA method versus the ME approach when computing the cardinalities for predicates on these column pairs.

As can be seen, use of the ME method can reduce both the median and the maximum estimation error substantially. Note that the error reduction is smaller for MAKE and MODEL than for the other cases, because ME – when knowing nothing about the two-way correlations – apportions the correlation "uniformly" among all three two-way correlations. The true correlation between MAKE and MODEL is the strongest of the two-way correlations, so that the selectivity correction achieved by ME in this case is smaller than in the other two cases.

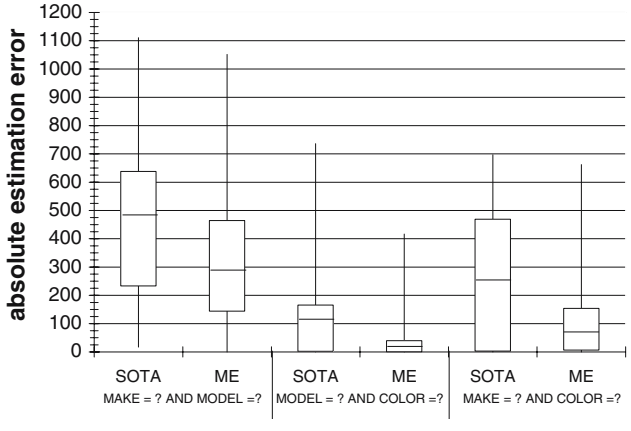

Fig. 10 Comparison of ME and State of the art (SOTA) when estimating $s_{1,2}, s_{2,3}$, or $s_{1,3}$ from $s_{1,2,3}$ (200 queries)

8.2 Performance of the ME method

In this section we study both the processing time required to produce the ME estimates and the numerical stability of the ME computation, examining the impact of our various preprocessing steps.

8.2.1 Iterative scaling without partitioning

The time complexity of iterative scaling, as described in Sect. 3.3, is exponential in $|P|$, the number of predicates, and linear in $|T|$, the size of the knowledge set (or, equivalently, the number of constraints in Eq. (3.2)). Therefore, it is important to analyze the maximum number of predicates for which this algorithm is feasible in practice. We ran the iterative scaling algorithm for different numbers of predicates and different-sized knowledge sets, using an Intel Pentium(R) III Mobile CPU 1,133 MHz with 512 MB RAM. For our experiments, we ensured that zero atoms were not a problem, in order to focus on the performance impact of the number of predicates.

Figure 11 shows the elapsed time (in seconds) until the iterative-scaling algorithm converges as a function of $|P|$ and $|\Delta T|$, where $\Delta T = T \backslash \{\{1\}, \{2\}, \ldots, n\}$. When $|P| < 5$, the time until convergence is less than 1 s for all values of $|T|$, and thus is omitted from the figure. When $|P| < 8$, the iterative-scaling algorithm exhibits sub-second convergence time and the performance impact of $|T|$ is negligible.

As $|P|$ increases, the computation time rises exponentially. We found that iterative scaling is impractical for more than ten predicates, for then the response time exceeds 1 s and thus has a clearly noticeable impact on the overall query optimization time. When $|P|$ is large, moreover, the amount of available knowledge $|T|$ has a noticeable impact on the overall performance. Fortunately, the algorithm's performance is generally

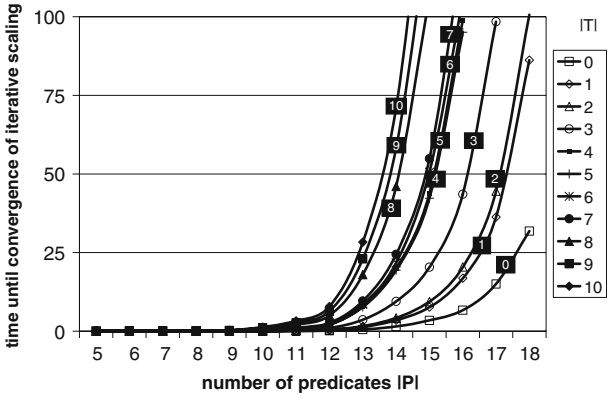

Fig. 11 Time (seconds) until convergence of iterative scaling, for various $|P|$ and $|\Delta T|$

acceptable when iterative scaling is performed over the local predicates on each single table separately (i.e., P being the set of local predicates on a table). We rarely encountered more than ten local predicates on a single table, even for very complex real-world queries. Indeed, all but five queries in the customer workloads available to the authors had less than 6 local predicates on a single table, even though the total number of predicates in many queries exceeded 50. Only three queries had more than 8 predicates on a single table, 11 being the maximum. As discussed in Sect. 5, when the number of predicates exceeds 8 we can often use preprocessing steps to bring down the number of predicates used for scaling to a reasonable number.

8.2.2 Comparison of algorithms for zero-atom detection

This section considers the relative performance impact of the two algorithms for zero-atom detection given in Sect. 3.4. Based on a set of 500 experimental queries, the box plots in Fig. 12 summarize the required computation times for these algorithms. The performance of the iterative ("I") and the approximation ("A") algorithms is essentially identical for up to six predicates, and then the computation times diverge sharply from seven predicates onward, with the approximation algorithm becoming much faster than the iterative algorithm.

Further experiments with ten or more predicates indicated that the median processing time for iterative zero-detection exceeds 10 s in the presence of 11 predicates and increases exponentially as the number of predicates increases further, whereas the processing time for the approximation method is less than 1 s in all cases considered.

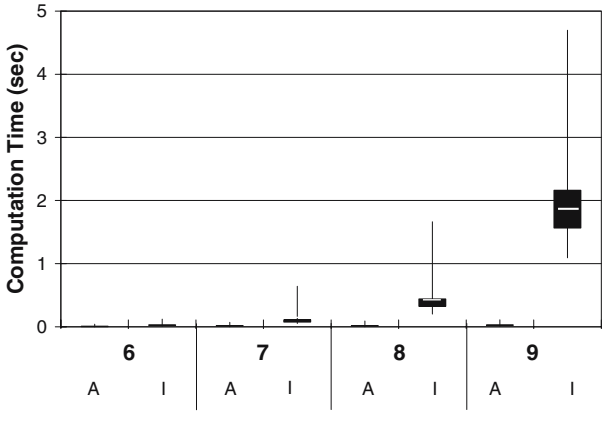

Fig. 12 Performance of preprocessing using two zero-atom detection algorithms, $|P| = 6, 7, 8, 9$

8.2.3 Impact of partitioning

We analyzed the impact of partitioning on a synthetic workload of 2,000 queries with between 10 and 14 predicates. The knowledge sets T were generated randomly. The box plots in Fig. 13 summarize the cumulative elapsed times for preprocessing and iterative scaling (i) without partitioning, (ii) with forced partitioning (as in Sect. 5.2) using various values for the partition-size parameter μ, and (iii) with "unforced" partitioning as in Sect. 5.1 ($\mu = \infty$). Although the median time of 2.5 s for the unpartitioned case may appear to be almost acceptable, the worst-case time of 136 s prevents this algorithm from being practical. Without impacting the quality of the solution, unforced partitioning reduces the median to 0.1 s. In this case, the 75% of the elapsed times are also sub-second, while the maximum elapsed time is still unacceptably large at about 62 s. With little impact on the overall ME solution quality, forced partitioning with $\mu = 10$ brings the maximum computation time down to 2 s. Reducing μ further to 8 (and 5) achieves sub-second computation times for all queries.

Figure 14 shows the impact of partitioning on the quality of the ME solution for this workload. For each

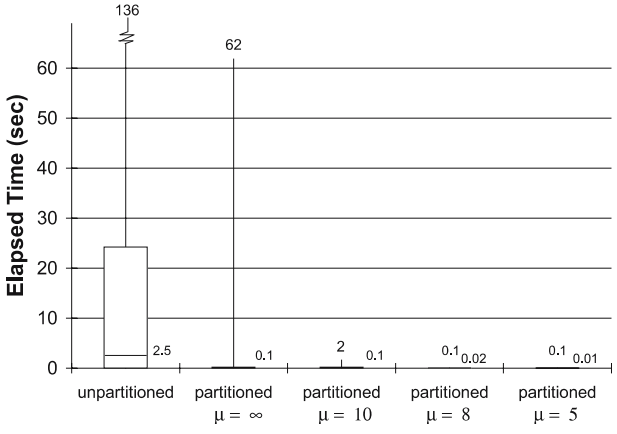

Fig. 13 Performance of partitioning

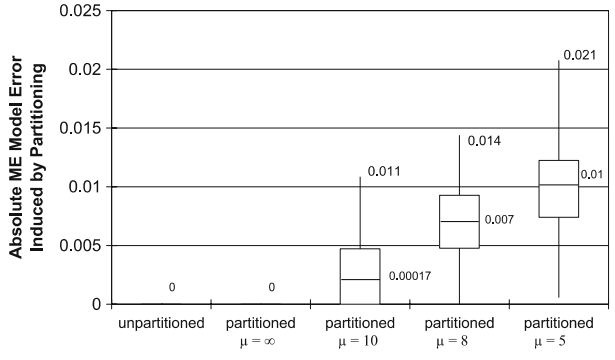

Fig. 14 Quality of partitioning

query and each partitioning scheme considered, we computed the absolute error $E = |s' - s|$, where s is the exact ME selectivity estimate (i.e., with no partitioning) and s' is the corresponding selectivity estimate based on partitioning. As can be seen from the box plots in the figure, the error is trivially zero in the cases of unpartitioned iterative scaling and unforced partitioning, and increases with decreasing μ.

In our experiments, a median error of less than 0.007 (about 1,000 rows) had no significant impact on ultimate query performance, and so we selected $\mu = 8$ as our default value for partition sizes.

8.2.4 *Overall numerical stability and performance of iterative scaling with preprocessing*

Figure 15 shows the effect of the preprocessing steps on the numerical stability of the ME computation. We used 600 random queries with BFs involving between 3 and 14 columns. The first (respectively, second) three bars show the effect of the preprocessing without (respectively, with) partitioning. With no preprocessing (Bar 1), the iterative-scaling algorithm fails for 24% of the queries due to inconsistent constraints and for an additional 2% of the queries due to implied zeros. Inconsistency resolution (Bar 2) guarantees the existence of a maximum entropy solution, but iterative scaling still converges for only 74% of the queries because of implied zeros. Elimination of implied zeros (Bar 3) produces a maximum entropy solution for every query. Bars 1P–3P show similar results in the presence of partitioning.

Based on a workload of 2,000 queries as in the partitioning experiments of Sect. 8.2, Fig. 16 shows the average processing time (in ms) for each step of the ME computation. As in the previous section, we distinguish between various partitioning strategies: unpartitioned (denoted by "U" in the figure), unforced partitioning

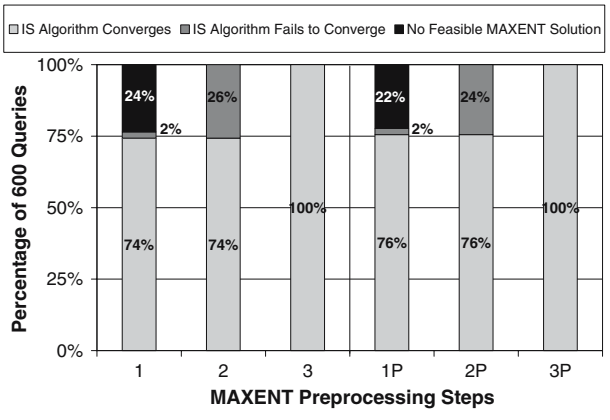

Fig. 15 Robustness of the ME computation

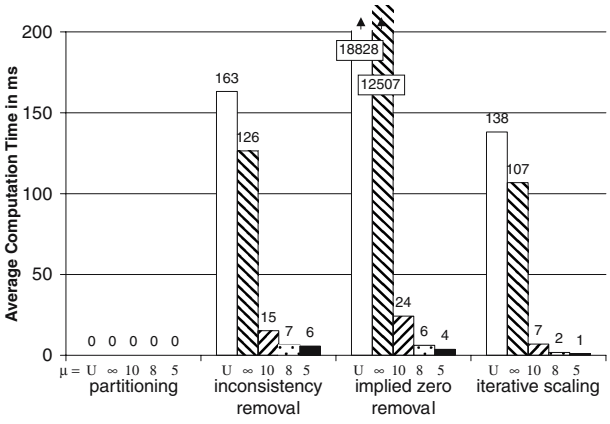

Fig. 16 Performance of iterative scaling with preprocessing

($\mu = \infty$), and forced partitioning with various values of μ. We again see that forced partitioning has a tremendous impact on the overall computation time. Also note that, with $\mu = 8$, the overall time required to compute the ME solution is only about 15 ms on average, while the impact on the quality of the ME solution is negligible, as mentioned previously.

8.3 Query execution time

Although improving the cardinality estimates yields a better model for the query optimizer, the bottom line of optimization is the improvement in query execution time. We measured the impact that use of the ME method has on the query execution time for our experimental workload of 200 queries against the DMV database. Recall that, in the previous experiments, we used only the part of each query that selects a particular MAKE, MODEL, and COLOR from the CAR table, ignoring all other tables referenced by the query. In this experiment we used each query in its entirety. Besides applying selection predicates to the CAR table, each full query also joins the CAR table with up to three additional tables and applies additional local predicates on the other tables. For all queries, the execution time of iterative scaling was less than one second and also below 1% of the total query execution time.

Figure 17 shows the performance benefit observed when running the queries with MVS created for all three two-way correlations on CAR, improving the estimates for the CAR table as in Case 2.3 in Sect. 4.1. Of the 200 queries, 92 showed only marginal performance gains or no gains at all. On the other hand, many queries executed between two and five times faster when using the ME estimates as opposed to the SOTA estimates. Nine queries had a performance gain of more than an order

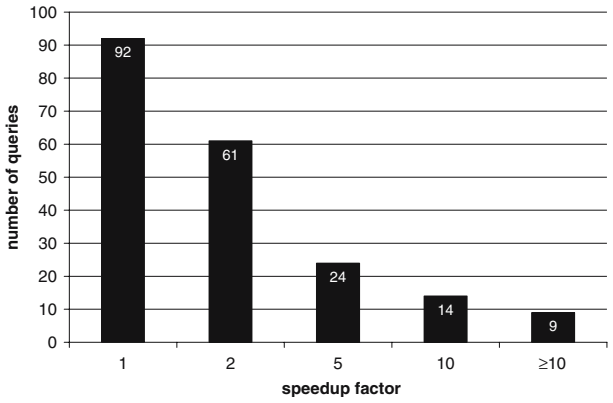

Fig. 17 Performance benefit of using two-way MVS

of magnitude, due to improvements in join order and/or join methods; we emphasize that these improvements resulted from the better cardinality estimates on just the CAR table alone.

9 Related work

Cost-based query optimization was introduced in [30]. Many researchers have investigated the use of statistics for query optimization, especially for estimating the selectivity of single-column predicates using histograms [18,27,28] and for estimating join sizes [22,34,35] using parametric methods [10,24]. Some MVS proposed for cardinality estimation include multidimensional histograms [29], statistics on views [6,13], and Bayesian [15] or other forms [12] of probabilistic models.

Recent research in selectivity estimation has mostly focused on determining which statistics to collect and how to collect them efficiently. The methods in [5,8] analyze the query workload to select a set of statistics to maintain, such as MVS on base data or query expressions. In [33], a feedback loop is employed in which query executions are monitored, the monitored information is analyzed to determine estimation errors, and the feedback information is used to adjust various stored statistics. The papers of [1,4,32] provide approaches to maintaining MVS based on query feedback by incrementally building a multidimensional histogram that can be used to estimate the selectivity of conjunctive predicates. The techniques in [2] use runtime feedback and other information to determine when and how to collect statistics. [21] use a sampling-based chi-squared test to determine the most relevant MVS to collect for query optimization.

Although a large body of work, as surveyed above, focuses on various aspects of recommending, computing, storing, and maintaining statistics for query optimization, no previous research has addressed the problem of combining selectivities derived from arbitrary (possibly inconsistent) available MVS to improve selectivity estimation of conjunctive predicates. Prior methods have derived selectivities for each conjunct from single-column statistics and combined them using the assumption of statistical independence, typically in an ad hoc manner.

Information theory and the broad mathematical principle of maximum entropy have been applied to other domains such as machine translation [14] and information retrieval [9]. Recently, [32] have proposed the use of ME methods for maintaining multivariate histograms, i.e., not for combining MVS as in the current paper, but rather for producing MVS in the first place.

10 Conclusions

We have presented a novel ME method for estimating the selectivity of conjunctive predicates, based on an information-theoretically sound approach that takes into account available statistics on both single columns and groups of columns. The model avoids arbitrary biases, inconsistencies, and the flight from knowledge to ignorance by deriving missing knowledge using the ME principle. This principle consistently extends the principles of uniformity and independence used in state-of-the-art selectivity models to exploit any available multi-attribute information. The specific embodiment of our approach described here solves the constrained ME optimization problem using Lagrange multipliers and an iterative scaling algorithm. We have implemented this method in a prototype version of DB2 UDB, improving the quality of the query optimizer while also simplifying its logic.

We have also developed several enhancements of the basic technology in order to handle the various real-world challenges that we encountered during implementation and testing. These include partitioning strategies that permit application of our methods to complex queries with many predicates, techniques for dealing with inconsistent input data, and an approach to handling join queries.

In our experiments, use of the ME approach improved selectivity estimates significantly over the state-of-the-art model used in DB2 UDB, often by orders of magnitude. This increased accuracy in turn yielded considerable improvements in query execution times for an example workload on our DMV dataset, with several queries running faster by orders of magnitude. Iterative scaling, together with our preprocessing techniques,

produced selectivity estimates for real-world queries with response times on the order of tens of milliseconds, adding less than a second to overall optimization time in all but the most complex queries.

Future work includes the investigation of efficient alternatives to the iterative scaling algorithm to further reduce the time required to compute the ME solution; we are currently investigating an algorithm based on Newton's method. We are also investigating more sophisticated approaches to forced partitioning; the idea is to use a criterion for discarding an element of T that explicitly depends upon whether the element spans multiple tables, the set of elements that have been discarded previously, and so forth. In addition, we are working on extending the scope of our ME method to permit cardinality estimation for distinct projections; the latter functionality is needed for optimizing queries with DISTINCT or GROUP BY clauses. Finally, we are studying a variety of potential applications of the ME principle to the recommendation, construction, and maintenance of MVS.

Acknowledgements The authors wish to thank the reviewers for helpful comments that greatly improved the paper.

Appendix A: selectivity estimation for conjunctive predicates in DB2 UDB

We describe the state of the art in exploiting MVS during query optimization. To our knowledge, prior research has neither formalized nor described solutions to this problem that go beyond classical independence and/or using MVS when they exactly match the columns referenced in a conjunctive predicate. We therefore describe the approach currently taken by DB2 UDB, using the notation introduced in Sects. 2.1 and 3.1.

In the following we assume that the selectivities of simple predicates are always available, so that the knowledge set T satisfies $T \supseteq T_0$, where $T_0 = \{\{1\}, \{2\}, \ldots, \{n\}\}$. The goal is to estimate the selectivity s_X for some $X \in 2^{\{1,2,\ldots,n\}}$ with $|X| > 1$. We say that an element $Y \in T$ is *relevant* to s_X if $Y \subseteq X$, and denote by R_X the set of all elements in T that are relevant to s_X. We also write $T_{0,X} = \{\{i\} | i \in X\} \cup \{\emptyset\}$. For $Y, Z \in R_X$, we say that *Y has a higher degree of correlation than Z*, and write $Y \succ Z$, if either (i) $|Y| > |Z|$ or (ii) $|Y| = |Z|$ and $s_Y / \prod_{i \in Y} s_i \geq s_Z / \prod_{i \in Z} s_i$.

DB2 UDB estimates s_X as $\prod_{i=1}^{k} s_{X_i}$, where the sets $X_1, X_2, \ldots, X_k \in R_X$ are chosen such that

(1) $X = \bigcup_{i=1}^{k} X_i$,
(2) $X_i \cap X_j = \emptyset$ for $i \neq j$, and
(3) $X_i \succ Y$ for $Y \in R_X \backslash \{X_1, X_2, \ldots, X_i\}$, $i = 1, 2, \ldots, k$,

i.e., the X_i*'s* are exhaustive and mutually exclusive, and the X_i sequence is constructed by selecting elements of R_X in decreasing order of degree of correlation. Any ties with respect to degree of correlation are broken arbitrarily; thus if there exist several possible sequences of X_i*'s* that satisfy the above criteria, then one such sequence is selected arbitrarily.

Examples:

(a) Estimate $s_{1,2,3}$ given $R_X = T_{0,X} \cup \{\{1,2,3\}\}$:
$s_{1,2,3} = s_{1,2,3}$.
(b) Estimate $s_{1,2,3}$ given $R_X = T_{0,X} \cup \{\{1,2\},\{2,3\},\{1,2,3\}\}$:
$s_{1,2,3} = s_{1,2,3}$.
(c) Estimate $s_{1,2,3}$ given $R_X = T_{0,X}$:
$s_{1,2,3} = s_1 * s_2 * s_3$.
(d) Estimate $s_{1,2,3,4,5}$ given $R_X = T_{0,X} \cup \{\{1,2\},\{3,4\}\}$:
$s_{1,2,3,4,5} = s_{1,2} * s_{3,4} * s_5$.
(e) Estimate $s_{1,2,3,4,5}$ given $R_X = T_{0,X} \cup \{\{1,2,5\},\{3,4\}\}$:
$s_{1,2,3,4,5} = s_{1,2,5} * s_{3,4}$.
(f) Estimate $s_{1,2,3,4,5}$ given $R_X = T_{0,X} \cup \{\{1,2\},\{2,3,4\}\}$:
$s_{1,2,3,4,5} = s_{2,3,4} * s_1 * s_5$.
(g) Estimate $s_{1,2,3,4,5}$ given $R_X = T_{0,X} \cup \{\{1,2\},\{2,3\}\}$ with $\{1,2\} \succ \{2,3\}$:
$s_{1,2,3,4,5} = s_{1,2} * s_3 * s_4 * s_5$.
(h) Estimate $s_{1,2,3,4,5}$ given $R_X = T_{0,X} \cup \{\{1,2\},\{2,3\},\{1,3\}\}$ with $\{1,2\} \succ \{2,3\}$ and $\{1,2\} \succ \{1,3\}$:
$s_{1,2,3,4,5} = s_{1,2} * s_3 * s_4 * s_5$.

Typically, MVS are provided to the optimizer precisely for those column sets in which the independence assumption breaks down. It follows that in scenarios such as Examples (f) through (h), the foregoing estimation method can produce large errors by ignoring crucial information. For instance, in Example (f), the final estimate ignores any correlation between s_1 and s_2. Likewise, in Examples (g) and (h), the estimation method assumes (probably erroneously) that s_3 is completely independent of s_1 and s_2. Use of this method can cripple dynamic optimization and reoptimization schemes, in which increasing amounts of information are provided to the optimizer over time. For example, although the optimizer has been provided with one more piece of information in Example (h) relative to Example (g), the estimate of $s_{1,2,3,4,5}$ does not improve. These problems motivate our new ME estimation method, which uses all available information.

Appendix B: validity of partitioning

It suffices to prove the claim in Sect. 5.1 for the case $k = 2$; the general case then follows from a straightforward induction argument. The claim is a direct consequence of the following theorem. Consider two sets $A = \{a_1, a_2, \ldots, a_m\}$ and $B = \{b_1, b_2, \ldots, b_n\}$, and let $\{p_{ij} : 1 \le i \le m, 1 \le j \le n\}$ be the ME probability distribution over the Cartesian product $A \times B$, subject to sets of constraints of the form

$$\sum_{i \in U_s} \sum_{j=1}^{n} p_{ij} = \phi_s, \quad s = 1, 2, \ldots, S \tag{B.1}$$

and

$$\sum_{j \in V_t} \sum_{i=1}^{m} p_{ij} = \theta_t, \quad t = 1, 2, \ldots, T, \tag{B.2}$$

where each U_s is a subset of $\{1, 2, \ldots, m\}$ and each V_t is a subset of $\{1, 2, \ldots, n\}$. Let $\{q_i : 1 \le i \le m\}$ be the ME distribution over the elements in A, subject to the constraints in Eq. (B.1), which can be rewritten as

$$\sum_{i \in U_s} q_i = \phi_s, \quad s = 1, 2, \ldots, S.$$

Similarly, let $\{r_j : 1 \le j \le n\}$ be the ME distribution over the elements in B, subject to the constraints in Equation B.2.

Theorem *The joint ME distribution satisfies $p_{ij} = q_i * r_j$ for $1 \le i \le m, 1 \le j \le n$.*

To apply this result in the setting of selectivity estimation, we identify the elements of A with the atoms corresponding to the predicate set P_1 and the elements of B with the atoms corresponding to the predicate set P_2.

We prove the theorem by forming the Lagrangian for the full ME optimization problem over the Cartesian product, where we denote by λ_s the Lagrange multiplier corresponding to the sth constraint in Eq. (B.1), by μ_t the multiplier corresponding to the tth constraint in Eq. (B.2), and by γ the multiplier corresponding to the constraint that the p_{ij}'s sum to 1. Taking derivatives, we find that the optimality equations are given by

$$\log p_{ij} + 1 = \sum_{s \in \Gamma_i} \lambda_s + \sum_{t \in \Lambda_j} \mu_t + \gamma,$$

where

$$\Gamma_i = \{s | i \in U_s\} \quad \text{and} \quad \Lambda_j = \{t | j \in V_t\}.$$

Exponentiating and dividing by e, we see that the joint probabilities in the ME distribution have the form

$$p_{ij} = c\alpha_i \beta_j. \tag{B.3}$$

Define marginal probabilities for the above ME distribution by setting

$$y_i = \sum_{j=1}^{n} p_{ij} \quad \text{and} \quad z_j = \sum_{i=1}^{m} p_{ij}.$$

Substituting the relation in Eq. (B.3) into the above equations and solving, we find that

$$\alpha_i = \frac{y_i}{c \sum_j \beta_j} \quad \text{and} \quad \beta_j = \frac{z_j}{c \sum_i \alpha_i},$$

so that

$$p_{ij} = \frac{x_i * y_j}{c \sum_i \alpha_i \sum_j \beta_j}.$$

Substituting Eq. (B.3) into the relation $\Sigma_{i,j} p_{ij} = 1$, we see that the denominator of the foregoing equation is equal to 1, so that $p_{ij} = x_i * y_j$ for all i and j. Thus the joint ME distribution over the Cartesian product is equal to the product of two marginal distributions. To complete the proof, observe that, for the joint ME distribution,

$$\begin{aligned} \sum_{i,j} p_{ij} \log p_{ij} &= \sum_{i,j} x_i y_j \log x_i y_j = \sum_{i,j} x_i y_j (\log x_i + \log y_j) \\ &= \sum_i x_i \log x_i + \sum_j y_j \log y_j, \end{aligned}$$

where we have used the fact that the x_i's and y_j's both sum to 1. Thus, to maximize the entropy, we must take $x_i = q_i$ and $y_j = r_j$.

The foregoing proof formally establishes the fact that, in the absence of information about joint probabilities, the ME condition reduces to the independence assumption. An even easier proof shows that, for a marginal distribution with no available information, the ME condition reduces to the uniformity assumption. Indeed, for a distribution $\{x_i : 1 \le i \le m\}$, let λ be the Lagrange multiplier corresponding to the constraint that the probabilities sum to 1. Then the optimality equation is

$$\log x_i + 1 = \lambda, \quad 1 \leqslant i \leqslant m,$$

so that that $x_i = x_j$ for all i, j, and hence the distribution is uniform: $x_i = 1/m$ for all i.

Appendix C: conditional independence under ME

Suppose that $N = \{1, 2, 3\}$ and $T = \{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{2, 3\}, \emptyset\}$, and we are given arbitrary but fixed feasible values for the selectivities $s_1, s_2, s_3, s_{1,2}$ and $s_{2,3}$. Also suppose that the unknown selectivity $s_{1,2,3}$ is computed using the ME principle. We establish the identity $s_{1,2,3} = s_{1,2} * s_{2,3}/s_2$ that is used in the text. Note that, if

we divide both sides by s_2, then we obtain the equivalent identity

$$\frac{s_{1,2,3}}{s_2} = \frac{s_{1,2}}{s_2} * \frac{s_{2,3}}{s_2},$$

which is equivalent to $s_{1,3|2} = s_{1|2} * s_{3|2}$, where $s_{Y|X}$ can be interpreted as the probability that a randomly selected row satisfies the predicates $\{p_j : j \epsilon Y\}$, given that the row satisfies the predicates $\{p_j : j \epsilon X\}$. Thus the foregoing identity asserts that p_1 and p_3 are conditionally independent, given that p_2 holds. In this sense the ME distribution tries to be "as independent as possible" given the knowledge set T.

To establish the desired result, we observe that the optimality equations are given by

$$\begin{aligned}
x_{000} &= c \\
x_{100} &= c * z_1 \\
x_{010} &= c * z_2 \\
x_{001} &= c * z_3 \\
x_{101} &= c * z_1 * z_3 \\
x_{110} &= c * z_1 * z_2 * z_{1,2} \\
x_{011} &= c * z_2 * z_3 * z_{2,3} \\
x_{111} &= c * z_1 * z_2 * z_3 * z_{1,2} * z_{2,3},
\end{aligned}$$

where $c = z_\emptyset / e$. Using these identities, we find that

$$\begin{aligned}
\frac{s_{1,2}s_{2,3}}{s_2} &= \frac{(x_{110}+x_{111})(x_{011}+x_{111})}{x_{010}+x_{110}+x_{011}+x_{111}} \\
&= \frac{(cz_1z_2z_{1,2} + cz_1z_2z_3z_{1,2}z_{2,3})(cz_2z_3z_{2,3} + cz_1z_2z_3z_{1,2}z_{2,3})}{cz_2 + cz_1z_2z_{1,2} + cz_2z_3z_{2,3} + cz_1z_2z_3z_{1,2}z_{2,3}} \\
&= \frac{cz_1z_2z_{1,2}(1+z_3z_{2,3})cz_2z_3z_{2,3}(1+z_1z_{1,2})}{cz_2(1+z_1z_{1,2}+z_3z_{2,3}+z_1z_3z_{1,2}z_{2,3})} \\
&= cz_1z_2z_3z_{1,2}z_{2,3} \cdot \frac{(1+z_3z_{2,3})(1+z_1z_{1,2})}{(1+z_1z_{1,2}+z_3z_{2,3}+z_1z_3z_{1,2}z_{2,3})} \\
&= cz_1z_2z_3z_{1,2}z_{2,3} \\
&= s_{1,2,3}.
\end{aligned}$$

References

1. Aboulnaga, A., Chaudhuri, S.: Self-tuning histograms: Building histograms without looking at data. SIGMOD 181–192 (1999)
2. Aboulnaga, A., Haas, P., Lightstone, S., et al.: Automated statistics collection in DB2 UDB. VLDB 1146–1157 (2004)
3. Ault, M., Tumma, M., Liu, D., et al.: Oracle Database 10 g new features: Oracle10 g reference for advanced tuning and administration. Rampant TechPress (2003)
4. Bruno, N., Chaudhuri, S., Gravano, L.: STHoles: a multidimensional workload-aware histogram. SIGMOD 211–222 (2001)
5. Bruno, N., Chaudhuri, S.: Exploiting statistics on query expressions for optimization. SIGMOD 263–274 (2002)
6. Bruno, N., Chaudhuri, S.: Efficient creation of statistics over query expressions. ICDE 201–212 (2003)
7. Bruno, N., Chaudhuri, S.: Conditional selectivity for statistics on query expressions. SIGMOD 311–322 (2004)
8. Chaudhuri, S., Narasayya, V.: Automating statistics management for query optimizers. ICDE 339–348 (2000)
9. Chiu, D., Wong, A., Cheung, B.: Information discovery through hierarchical maximum-entropy discretization and synthesis. In: Piatesky-Shapiro, G., Fracley, W.J., (eds.), Knowledge Discovery in Databases. pp. 125–140 MIT Press, Cambridge (1991)
10. Christodoulakis, S.: Estimating record selectivities. Inf. Syst. **8**(2):105–115 (1983)
11. Darroch, J.N., Ratcliff, D.: Generalized iterative scaling for log-linear models. Ann. Math. Statist. 43:1470–1480 (1972)
12. Deshpande, A., Garofalakis, M., Rastogi, R.: Independence is good: dependency-based histogram synopses for high-dimensional data. SIGMOD 199–210 (2001)
13. Galindo-Legaria, C., Joshi, M., Waas, F., et al.: Statistics on views. VLDB 952–962 (2003)
14. García-Varea, I., Och, F., Ney, H., et al.: Refined Lexikon models for statistical machine translation using a maximum-entropy approach. ACL 204–211 (2001)
15. Getoor, L., Taskar, B., Koller, D.: Selectivity estimation using probabilistic models. SIGMOD 461–472 (2001)
16. Greiff, W., Ponte, J.: The maximum-entropy approach and probabilistic IR models. ACM Trans. Inform. Sys. **18**(3):246–287 (2000)
17. Guiasu, S., Shenitzer, A.: The principle of maximum-entropy. Math. Intell. **7**(1):42–48 (1985)
18. Haas, P., Swami, A.: Sampling-based selectivity estimation for joins using augmented frequent-value statistics. ICDE 522–531 (1995)
19. IBM Corp.: DB2 Universal Database for iSeries: Database Performance and Query Optimization (2002)
20. IBM Corp.: DB2 v8.2 Performance Guide (2004)
21. Ilyas, I.F., Markl, V., Haas, P.J., Brown, P.G., Aboulnaga, A.: CORDS: automatic discovery of correlations and soft functional dependencies. SIGMOD 647–658 (2004)
22. Ioannidis, Y.E., Christodoulakis, S.: Propagation of errors in the size of join results. SIGMOD 268–277 (1991)
23. Kutsch, M., Haas, P.J., Markl, V., Megiddo, N., Tran, T.M.: Integrating a maximum-entropy cardinality estimator into DB2 UDB. EDBT 1092–1096 (2006)
24. Lynch, C.A.: Selectivity estimation and query optimization in large databases with highly skewed distribution of column values. VLDB 240–251 (1988)
25. Markl, V., Megiddo, N., Kutsch, M., Tran, T.M., Haas, P.J., Srivastava, U.: Consistently estimating the selectivity of conjuncts of predicates. VLDB 378–384 (2005)
26. Microsoft Corp.: SQL Server 2000 Books Online v8.00.02 (2004)
27. Piatetsky-Shapiro, G., Connell, C.: Accurate estimation of the number of tuples satisfying a condition. SIGMOD 256–276 (1984)
28. Poosala, V., et al.: Improved histograms for selectivity estimation of range predicates. SIGMOD 294–305 (1996)
29. Poosala, V., Ioannidis, Y.: Selectivity estimation without the attribute value independence assumption. VLDB 486–495 (1997)

30. Selinger, P.G., et al.: Access path selection in a relational DBMS. SIGMOD 23–34 (1979)
31. Shannon, C.E.: A mathematical theory of communication. Bell Sys. Tech. J. **27**, 379–423 623–656 (1948)
32. Srivastava, U., Haas, P.J., Markl, V., Megiddo, N.: ISOMER: consistent histogram construction using query feedback. ICDE 6 (2006)
33. Stillger, M., Lohman, G., Markl, V., Kandil, M.: LEO – DB2's learning optimizer. VLDB 19–28 (2001)
34. Swami, A.N., Schiefer, K.B.: On the estimation of join result sizes. EDBT 287–300 (1994)
35. Van Gelder, A.: Multiple join size estimation by virtual domains. PODS 180–189 (1993)