Evaluating Aggregate Operations Over Imprecise Data

Arbee L.P. Chen, *Member, IEEE Computer Society*, Jui-Shang Chiu, and Frank S.C. Tseng, *Member, IEEE Computer Society*

Abstract—Imprecise data in databases were originally denoted as *null values*, which represent the meaning of "values unknown at present." More generally, a partial value corresponds to a finite set of possible values for an attribute in which exactly one of the values is the "true" value. In this paper, we define a set of extended aggregate operations, namely *sum, average, count, maximum,* and *minimum*, which can be applied to an attribute containing partial values. Two types of aggregate operators are considered: *scalar aggregates* and *aggregate functions.* We study the properties of the aggregate operations and develop efficient algorithms for *count, maximum* and *minimum*. However, for *sum* and *average*, we point out that in general it takes exponential time complexity to do the computations.

Index Terms—Relational databases, null values, partial values, scalar aggregates, aggregate functions, graph theory.

1 Introduction

INCOMPLETE information can be classified into two aspects, i.e., imprecise information and uncertain information. Dubois and Prade [12] distinguish between imprecise and uncertain information by stating that the concept of *imprecision* is relevant to the content of an attribute value, while the concept of *uncertainty* is relevant to the degree of truth of its attribute value. In this paper, we focus our attention on the imprecision aspect. To the uncertainty aspect, two major approaches are the *possibility* approach based on the fuzzy set theory [33] and the *probability* approach based on the probability theory.

Imprecise data exist in the real world. *Null values* were originally adopted to represent the meaning of "values unknown at present" in database systems. Codd [6], [7] pioneered the work on extended relational algebra to manipulate null values. For a concise review of handling null values by algebraic approaches, refer to [21].

The concept of null values has been generalized to the concept of *partial values* by Grant [14]. Instead of being treated as an atomic value, an attribute value in a table is considered as a nonempty subset of the corresponding domain. A partial value is represented as an interval such that exactly one of the values in the interval is the "true" value. In our work, a partial value is considered as a finite set of *possible* values in which exactly one of the values is the "true" value [11]. Therefore, an *applicable null value* [8] is a partial value which corresponds to the whole domain. Also, a definite value a is a partial value which corresponds to a set containing a only.

Partial values have been considered by DeMichiel [11] for resolving domain mismatch problems in multidatabase systems, in which an algebraic approach for operating on partial values is proposed. Grant and Minker [15] consider query answering for indefinite (disjunctive) databases that contain disjunctive formulas in first-order logic in which each corresponds to a finite set of possible tuples under the relational model. Ola [23] presents an approach to processing relations containing exclusive disjunctive data (such as partial values). In [29], we discuss the elimination of redundant partial values. Moreover, in [32] we study the problem of incrementally refining partial values into more informative ones or into definite values by utilizing the knowledge of integrity constraints. Lipski [20] presents a general discussion on manipulating imprecise information including partial values.

To the uncertainty aspect, various kinds of fuzzy relational databases based on the possibility approach were proposed, such as Bosc et al. [3], Buckles and Petry [4], Prade and Testemale [25], and Zemankova and Kandel [34]. Moreover, the work with the probability approach includes Barbará et al. [1], Cavallo and Pittarelli [5], Tsai and Chen [28], and Tseng et al. [30]. In [28], [30], we generalize partial values into *probabilistic partial values* to provide a probabilistic approach to query processing in heterogeneous database systems.

In practice, relational algebra or calculus are inadequate for many important applications involving statistical information or aggregations. Therefore, modern query languages, like SQL [9], are "more than" *relational complete* and equipped with some useful aggregate operations. However, most of the previous works on the manipulation of incomplete information usually discuss the extended relational algebra and ignore the extended aggregate operators.

Aggregate operations can be applied to all the tuples in a table, to a subset of the table (specified by a WHERE

• *A.L.P. Chen and J.-S. Chiu are with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan 300, Republic of China. E-mail: alpchen@cs.nthu.edu.tw.*
• *F.S.C. Tseng is with the Department of Information Management, Yuan-Ze Institute of Technology, 135, Far-Eastern Road, Chung-Li, Taiwan 32026, Republic of China.*

Manuscript received June 22, 1993; revised May 24, 1994.
For information on obtaining reprints of this article, please send e-mail to: transactions@computer.org, and reference IEEECS Log Number K96023.

clause), or to one or more groups of tuples in the table (specified by a GROUP BY clause). For each set of tuples to which an aggregate operation is applied, a single value is generated. Aggregate operations are often used with the GROUP BY clause for a more powerful functionality. Optionally, the keyword DISTINCT can be used with *count* to eliminate duplicate values before the aggregate operation is applied.

In this paper, we define a set of extended aggregate operations, namely *sum, average, count, maximum,* and *minimum,* which can be applied to attributes containing partial values. These aggregate operators are divided into the following two types (depending on whether the GROUP BY clause is used).

1) *Scalar aggregates*—They take a set of partial values as input and produce a single partial value as a result.
2) *Aggregate functions*—They compute an aggregation on attribute A_y of a relation $T[\ldots, A_x, A_y, \ldots]$ containing partial values based on the distinct scalar values a_x of attribute A_x, and return $<a_x, \eta_y>$, where η_y is the aggregation of A_y based on a_x. For databases containing partial values, the extended aggregate functions produce η_y as a partial value.

In [10], DeMichiel provides an *approximation* of aggregate operators over partial values. However, from the definitions in [10], only the *boundary possible values* (i.e., the maximum and minimum possible values) are returned for some aggregate operators. In comparison, our work generates *all possible values* for each aggregate operator instead of *boundary possible values.*

Özsoyoglu et al. [24] study some aggregate operations over *set-valued* attributes. All values in the set are the true values. The *grouping attributes* if specified can be set-valued attributes. However, it is required that the *aggregate attribute* (to which an aggregate function is applied) should be a single-valued attribute. In comparison, in our work, both grouping attributes and aggregate attributes are partial values.

Rundensteiner and Bic [26], [27] study aggregate operations in possibilistic databases. The possibilistic relational model is characterized by a representation which allows for data values to be modeled by discrete possibility distributions. Their aggregate operators generate all possible values each with a possibility in exponential time. In comparison, based on partial values, our work generate all possible values without possibilities possibly in polynomial time.

Database practitioners are concerned primarily with the *performance* of a database system and many of the query evaluation algorithms for processing imprecise queries are very complex and inefficient. This inefficiency, as pointed out by Motro [22], is one of the reason for the slow acceptance of imprecision and uncertainty handling capabilities into commercial database systems. Therefore, this work is devoted to the accommodation of imprecise data in database systems with elaboration on speeding up its processing efficiency.

Since the brute-force computation of the extended aggregates is very time-consuming in general, we find some important properties and develop efficient algorithms (in polynomial time complexity) for *count, maximum,* and *minimum.* However, for *sum* and *average,* we point out that in general it takes exponential time complexity to do the computations.

The following section presents some basic concepts and definitions of partial values. Section 3 is devoted to the definition of our extended scalar aggregates and studies their properties. In Section 4, the definition of our extended aggregate functions and their evaluation algorithms are presented. Section 5 concludes our work.

2 Basic Concepts and Definitions

2.1 Basic Concept of Partial Values

Partial values model data imprecision in databases in the sense that, for an imprecise datum, its "true" value can be restricted to be in a specific set of possible values [11] or an interval of values [14]. In our work, we follow the definition of a partial value given in [11], which is formally stated as follows.

Definition 2.1. *A partial value, denoted $\eta = [a_1, a_2, \ldots, a_p]$, corresponds to a finite set of possible values, $\{a_1, a_2, \ldots, a_p\}$, $p \geq 1$, of the same domain for an attribute, in which exactly one of the values in η is the "true" value of η.*

For a partial value $\eta = [a_1, a_2, \ldots, a_p]$, a function ν is defined on it [11], where ν maps the partial value to its corresponding finite set of *possible* values; that is, $\nu(\eta) = \{a_1, a_2, \ldots, a_p\}$. Recall that an *applicable null value* [8], $\aleph$, can be considered as a partial value with $\nu(\aleph) = D$, where D is the whole domain. A relation containing tuples with partial values is called a *partial relation* [11]. Otherwise, it will be called a *definite relation.* We assume that each partial relation contains primary key attributes with definite values.

The *cardinality* of a partial value η is defined as $|\nu(\eta)|$ in [11]. When the cardinality of a partial value equals 1, i.e., there exists only one *possible* value, say a, in the partial value, then the partial value $[a]$ actually corresponds to the definite "true" value a. A definite value a can be also represented as a partial value $[a]$. Moreover, a partial value with cardinality greater than 1 is referred to as a *proper partial value* in [11]. For any two proper partial values, say η_1 and η_2, the "true" value of η_1 may not be the same as the "true" value of η_2 even if $\nu(\eta_1) = \nu(\eta_2)$.

2.2 Alternate Worlds of a Partial Relation

Let T be a partial relation on a set of attributes $\{A_1, A_2, \ldots, A_m\}$ containing a set of tuples $\{t_1, t_2, \ldots, t_n\}$ where $t_i = <\eta_{i1}, \eta_{i2}, \ldots, \eta_{im}>$, $1 \leq i \leq n$. Let $T[A_j]$ denote the projection of a partial relation T on an attribute A_j. We may enumerate all the possible cases that $T[A_j]$ represents by the following definition.

Definition 2.2. *Let $T[A_j] = \{\eta_{1j}, \eta_{2j}, \ldots, \eta_{nj}\}$ where $\eta_{ij} = t_i.A_j$, $t_i \in T$. An* interpretation *α of $T[A_j]$ is an assignment of values from $T[A_j]$ denoted by an n–nary vector $\alpha = (a_{1j}, a_{2j}, \ldots, a_{nj})$ where $a_{ij} \in \nu(\eta_{ij})$, $1 \leq i \leq n$.*

By Definition 2.2, for $T[A_j] = \{\eta_{1j}, \eta_{2j}, \ldots, \eta_{nj}\}$, $\nu(\eta_{1j}) \times \nu(\eta_{2j}) \times \cdots \times \nu(\eta_{nj})$ is the set of all interpretations of $T[A_j]$. Notice that η_{ij} and η_{lj}, $i \neq l$, can both appear in $T[A_j]$ even when $\nu(\eta_{ij}) = \nu(\eta_{lj})$.

EXAMPLE 2.1. Suppose $T[A_j] = \{\overbrace{[a, b]}^{\eta_{1j}}, \overbrace{[c, d]}^{\eta_{2j}}\}$, there are four interpretations, $\alpha_1 = (a, c)$, $\alpha_2 = (a, d)$, $\alpha_3 = (b, c)$, and $\alpha_4 = (b, d)$.

For $t_i \in T$, we may enumerate all the possible tuples that t_i represents by the following definitions.

DEFINITION 2.3. *Let $t_i = <\eta_{i1}, \eta_{i2}, \ldots, \eta_{im}> \in T$. An interpretation, $\alpha = (a_{i1}, a_{i2}, \ldots, a_{im})$, of t_i is an assignment of values from partial values in t_i such that $a_{ij} \in \nu(\eta_{ij})$, $1 \le j \le m$.*

DEFINITION 2.4. *Let α_k be an interpretation of t_i, $t_i \in T$. $\alpha_k(t_i)$ is defined as a tuple $< a_{i1_k}, a_{i2_k}, \ldots, a_{im_k} >$ where a_{ij_k} denotes the value of attribute A_j for tuple t_i under interpretation α_k. The set of all possible tuples represented by t_i,* rep(t_i), *is defined as* $\{\alpha_k(t_i) \mid$ for each interpretation α_k of $t_i\}$.

For a partial relation T, we may enumerate all the possible relations that T represents by the following definitions.

DEFINITION 2.5 *Let T be defined as the above. An interpretation α of T is an assignment of values from partial values in T denoted by an $n \times m$ matrix*

$$\alpha = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix},$$

where $a_{ij} \in \nu(\eta_{ij})$, $1 \le i \le n$, $1 \le j \le m$.

By Definition 2.5, for partial relation T,

$$\left\{ \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} \middle| \; a_{ij} \in \nu(\eta_{ij}),\ 1 \le i \le n,\ 1 \le j \le m \right\}$$

is the set of all interpretations of T.

DEFINITION 2.6. *Let α_k be an interpretation of T. $\alpha_k(T)$ is defined as a relation*

$$\{\alpha_k(t_i) \mid \alpha_k(t_i) = < a_{i1_k}, a_{i2_k}, \ldots, a_{im_k} >,\ t_i \in T\}.$$

The set of all possible relations represented by partial relation T, rep(T), *is defined as* $\{R_k \mid R_k = \alpha_k(T)$, for each interpretation α_k of $T\}$.

EXAMPLE 2.2. Consider the following partial relation T.

T

A_1	A_2
a	c
b	$[c, d]$

There are two interpretations, α_1 and α_2, of T:

$$\alpha_1 = \begin{bmatrix} a & c \\ b & c \end{bmatrix} \text{ and } \alpha_2 = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

The possible relations represented by T, $\alpha_1(T) = R_1$ and $\alpha_2(T) = R_2$, are shown below:

R_1

A_1	A_2
a	c
b	c

R_2

A_1	A_2
a	c
b	d

3 EXTENDED SCALAR AGGREGATES OVER PARTIAL VALUES

An extended scalar aggregate can be applied to an attribute containing partial values. It in turn produces a partial value.

3.1 Definitions of the Extended Scalar Aggregates

Each extended scalar aggregate is defined as follows through the function ν, which maps the result of an extended scalar aggregate (which is also a partial value) to the finite set of values to which it corresponds.

In the following, T denotes a partial relation and A_j denotes an attribute of T.

DEFINITION 3.1. The extended count *(for the number of distinct values) on A_j of T, denoted count $(T.A_j)$, is defined as:*

$$\nu(count(T.A_j)) = \{\, | R_k[A_j] | \mid R_k \in rep(T)\}.$$

DEFINITION 3.2. *The* extended sum *on A_j of T, denoted $sum(T.A_j)$, is defined as:*

$$\nu(sum(T.A_j)) = \left\{ \sum_{t_i \in R_k} a_{ij_k} \mid R_k \in rep(T) \right\}.$$

DEFINITION 3.3. *The* extended average *on A_j of T, denoted $avg(T.A_j)$, is defined as:*

$$\nu(avg(T.A_j)) = \left\{ \frac{\sum_{t_i \in R_k} a_{ij_k}}{|R_k|} \mid R_k \in rep(T) \right\}.$$

DEFINITION 3.4. *The* extended maximum *on A_j of T, denoted $max(T.A_j)$, is defined as:*

$$\nu(max(T.A_j)) = \left\{ \max_{t_i \in R_k} a_{ij_k} \mid R_k \in rep(T) \right\}.$$

DEFINITION 3.5. *The* extended minimum *on A_j of T, denoted $min(T.A_j)$, is defined as:*

$$\nu(min(T.A_j)) = \left\{ \min_{t_i \in R_k} a_{ij_k} \mid R_k \in rep(T) \right\}.$$

The following example illustrates these extended aggregates.

EXAMPLE 3.1. Consider the following partial relation.

Employees

name	*salary*
a	[20k, 80k]
b	[60k, 75k]
c	[20k, 90k]

We compute the set of possible relations rep(*Employee*), as follows:

rep(*Employees*)

{

a	20k
b	60k
c	20k

,

a	20k
b	60k
c	90k

,

a	20k
b	75k
c	20k

,

a	20k
b	75k
c	90k

,

a	80k
b	60k
c	20k

,

a	80k
b	60k
c	90k

,

a	80k
b	75k
c	20k

,

a	80k
b	75k
c	90k

}

Then the extended scalar aggregates can be computed as follows. Let $T = Employees$ and $A_j = salary$.

$$v(sum(Employees.salary)) = \left\{ \sum_{t_i \in R_k} a_{ij_k} \mid R_k \in rep(T) \right\}$$

$$= \{100\text{k}, 170\text{k}, 115\text{k}, 185\text{k}, 160\text{k}, 230\text{k}, 175\text{k}, 245\text{k}\}.$$

$$v(count(Employees.salary)) = \left\{ \left| R_k[A_j] \right| \mid R_k \in rep(T) \right\} = \{2, 3\}.$$

$$v(avg(Employees.salary)) = \left\{ \frac{\sum_{t_i \in R_k} a_{ij_k}}{|R_k|} \mid R_k \in rep(T) \right\}$$

$$= \left\{ \frac{100\text{k}}{3}, \frac{170\text{k}}{3}, \frac{115\text{k}}{3}, \frac{185\text{k}}{3}, \frac{160\text{k}}{3}, \frac{230\text{k}}{3}, \frac{175\text{k}}{3}, \frac{245\text{k}}{3} \right\}.$$

$$v(max(Employees.salary)) = \left\{ \max_{t_i \in R_k} a_{ij_k} \mid R_k \in rep(T) \right\}$$

$$= \{60\text{k}, 90\text{k}, 75\text{k}, 80\text{k}\}.$$

$$v(min(Employees.salary)) = \left\{ \min_{t_i \in R_k} a_{ij_k} \mid R_k \in rep(T) \right\}$$

$$= \{20\text{k}, 60\text{k}, 75\text{k}\}.$$

That is,

$sum(Employees.salary) =$

$\{100\text{k}, 170\text{k}, 115\text{k}, 185\text{k}, 160\text{k}, 230\text{k}, 175\text{k}, 245\text{k}\}$,

$count(Employees.salary) = [2, 3]$,

$avg(Emplyees.salary) =$

$$\left[\frac{100\text{k}}{3}, \frac{170\text{k}}{3}, \frac{115\text{k}}{3}, \frac{185k}{3}, \frac{160\text{k}}{3}, \frac{230\text{k}}{3}, \frac{175\text{k}}{3}, \frac{245\text{k}}{3} \right],$$

$max(Employees.salary) = [60\text{k}, 90\text{k}, 75\text{k}, 80\text{k}]$, and

$min(Employees.salary) = [20\text{k}, 60\text{k}, 75\text{k}]$.

3.2 Properties and Algorithms for Evaluating Scalar Aggregates over Partial Values

When $|T| = n$ and $|\eta_{ij}| = p$ for all $t_i \in T$, there are p^n interpretations of $T[A_j]$. Therefore, brute force methods for computing the extended scalar aggregates are exponential. In the following, we develop polynomial time algorithms to compute *count*, *maximum* and *minimum* based on their properties. Since the cardinalities of $\nu(sum(T.A_j))$ and $\nu(avg(T.A_j))$ are equal to p^n in the worst case (see Example 3.1), no polynomial time algorithms can be found for *sum* and *average*.

3.2.1 Properties of the Scalar Aggregate—Count

In this section, we use techniques in graph theory [2] to develop an efficient algorithm for *count*. Before discussing the details, we define some terminologies as follows.

DEFINITION 3.6. *For a set S of vertices in a graph $G(V, E)$, $S \subseteq V$, the* neighbor set *of S, denoted $N(S)$, is defined to be the set of all vertices adjacent to the vertices in S.*

DEFINITION 3.7. *For a bipartite graph $G = (X \cup Y, E)$, we say a set S, $S \subseteq X$,* covers *Y if $N(S) = Y$. If deleting any vertex in S would make S cease to* cover *Y then S is a* minimal *set that covers Y. S is a* minimum *set if it has the minimum cardinality among these cover sets. Note that Y can be an empty set.*

DEFINITION 3.8. *Let $Y = \{\eta_1, \eta_2, \ldots, \eta_n\}$ and $X = \bigcup_{1 \le i \le n} \nu(\eta_i) = \{a_1, a_2, \ldots, a_q\}$. The* membership graph *of Y over X is a bipartite graph $G = (X \cup Y, E)$, where*

$$E = \{(a_i, \eta_j) \mid a_i \in \nu(\eta_j), 1 \le i \le q, 1 \le j \le n\}.$$

DEFINITION 3.9. *For a bipartite graph $G = (X \cup Y, E)$, $|X| \le |Y|$, we say there is a* complete matching *M from X to Y if there is a matching of cardinality $|X|$, that is, each vertex in X is adjacent to a distinct vertex in Y.*

Hall [16] has given a necessary and sufficient condition under which there exists a complete matching M from X to Y for a bipartite graph $G = (X \cup Y, E)$.

THEOREM 3.1. ([16]) *Let $G = (X \cup Y, E)$ be a bipartite graph. There exists a complete matching from X to Y if and only if $|N(S)| \ge |S|, \forall S \subseteq X$.*

Now we state the properties for *count* as follows. We discuss the lower bound of $\nu(count(T.A_j))$ first. Then the upper bound of $\nu(count(T.A_j))$ is addressed. Finally, we prove that each integer number between the lower bound and the upper bound is in $\nu(count(T.A_j))$.

LEMMA 3.1. *Let $Y = T[A_j] = \{\eta_{1j}, \eta_{2j}, \ldots, \eta_{nj}\}$, $X = \bigcup_{1 \le i \le n} \nu(\eta_{ij})$, and $G = (X \cup Y, E)$ be the membership graph of Y over X. Let l denote the cardinality of the minimum set S, $S \subseteq X$, that covers Y, then for all $c \in \nu(count(T.A_j))$, $c \ge l$.*

PROOF. Suppose there is an element c of $\nu(count(T.A_j))$ such that $c < l$. That is, there is an interpretation

$$\alpha_k = (a_{1j_k}, a_{2j_k}, \ldots, a_{nj_k})$$

of $T[A_j]$ with $|R_k[A_j]| = c < l$, where

$$R_k[A_j] = \left\{ a_{ij_k} \mid 1 \le i \le n \right\}.$$

Since $a_{ij_k} \in v(\eta_{ij})$, for $1 \le i \le n$, $R_k[A_j]$ covers Y. But $|R_k[A_j]| < l$, which contradicts the assumption that the cardinality of the minimum set that covers Y is l. Hence, for all $c \in \mathcal{V}(count(T.A_j))$, $c \ge l$. □

The following lemma states the upper bound of $\mathcal{V}(count(T.A_j))$.

LEMMA 3.2. *Let $Y = T[A_j] = \{\eta_{1j}, \eta_{2j}, \ldots, \eta_{nj}\}$, $X = \bigcup_{1 \le i \le n} \mathcal{V}(\eta_{ij})$, and $G = (X \cup Y, E)$ be the membership graph of Y over X. Let u denote the cardinality of the maximum matching M in G, then for all $c \in \mathcal{V}(count(T.A_j))$, $c \le u$.*

PROOF. Suppose there is an element c of $\mathcal{V}(count(T.A_j))$ such that $c > u$. That is, there is an interpretation

$$\alpha_k = (a_{1j_k}, a_{2j_k}, \ldots, a_{nj_k})$$

of $T[A_j]$ with $|R_k[A_j]| = c > u$, where

$$R_k[A_j] = \left\{a_{ij_k} \mid 1 \le i \le n\right\}.$$

In other words, for each $a_{ij_k} \in R_k[A_j]$, there exists at least a distinct $\eta_{ij} \in T[A_j]$ such that $a_{ij_k} \in v(\eta_{ij})$. That implies, for all $S \subseteq R_k[A_j]$, $|S| \le |N(S)|$ in the membership graph of $T[A_j]$ over $R_k[A_j]$, $G' = (R_k[A_j] \cup T[A_j], E')$. Therefore, by Theorem 3.1, there exists a complete matching M^* from $R_k[A_j]$ to $T[A_j]$ in G'. By

$$R_k[A_j] = \left\{a_{ij_k} \mid 1 \le i \le n\right\} \subseteq X,$$

M^* is also a matching in G. But $|M^*| = |R_k[A_j]| > u$, which contradicts the assumption that the maximum matching in G has cardinality u. Hence, for all $c \in \mathcal{V}(count(T.A_j))$, $c \le u$. □

The following lemma gives a necessary and sufficient condition for an element to be in $\mathcal{V}(count(T.A_j))$.

LEMMA 3.3. *Let $Y = T[A_j] = \{\eta_{1j}, \eta_{2j}, \ldots, \eta_{nj}\}$ and $X = \bigcup_{1 \le i \le n} \mathcal{V}(\eta_{ij})$. There is a complete matching from S, $S \subseteq X$, to Y in the membership graph of Y over S, $G' = (S \cup Y, E')$, and S covers Y in $G = (X \cup Y, E)$ if and only if $|S| \in \mathcal{V}(count(T.A_j))$.*

PROOF. ($\Leftarrow$) Suppose $|S| = c \in \mathcal{V}(count(T.A_j))$. That is, there is an interpretation $\alpha_k = (a_{1j_k}, a_{2j_k}, \ldots, a_{nj_k})$ of $T[A_j]$ with $R_k[A_j] = S$. In other words, for each $a_{ij_k} \in v(\eta_{ij})$, there exists at least a distinct $\eta_{ij} \in T[A_j]$ such that $a_{ij_k} \in v(\eta_{ij})$. That implies, for all $S' \subseteq S$, $|S'| \le |N(S')|$. Therefore, by Theorem 3.1, we can construct a complete matching M^* from $R_k[A_j] = S$ to Y in G'. Besides, since $S = R_k[A_j] = \{a_{ij_k} \mid 1 \le i \le n\}$ and $a_{ij_k} \in v(S)$, for $1 \le i \le n$, S covers Y in G.

($\Rightarrow$) Assume there is a complete matching M from S to Y in the membership graph of Y over S and S covers Y in G. By S covers Y in G, we obtain $S \cap \eta_{ij} \ne \emptyset$, for all $\eta_{ij} \in Y$. To show that $|S| \in \mathcal{V}(count(T.A_j))$, we can generate an interpretation $\alpha_k = (a_{ij_k}, a_{2j_k}, \ldots, a_{nj_k})$ of $T[A_j]$ such that

$$a_{ij_k} = \begin{cases} a & \text{if } (a, \eta_{ij}) \in M, \\ \text{any } a \in S \cap \eta_{ij} & \text{otherwise.} \end{cases}$$

Then, we obtain $|R_k[A_j]| = |S| \in \mathcal{V}(count(T.A_j))$. □

The following theorem states that all integers between the lower bound and the upper bound are all in $\mathcal{V}(count(T.A_j))$.

THEOREM 3.2. *Let $Y = T[A_j] = \{\eta_{1j}, \eta_{2j}, \ldots, \eta_{nj}\}$, $X = \bigcup_{1\le i\le n} \mathcal{V}(\eta_{ij})$, and $G = (X \cup Y, E)$ be the membership graph of Y over X. Let l denote the cardinality of the minimum set S, $S \subseteq X$, that covers Y in G, and u denote the cardinality of the maximum matching M in G, then $\mathcal{V}(count(T.A_j)) = \{c \mid l \le c \le u\}$.*

PROOF. By Lemmas 3.1 and 3.2, we obtain for all $c \in \mathcal{V}(count(T.A_j))$, $l \le c \le u$. Now we need to show every integer between l and u (including l and u) is an element of $\mathcal{V}(count(T.A_j))$. To show that $l \in \mathcal{V}(count(T.A_j))$, since S covers Y in G, we can generate an interpretation $\alpha_k = (a_{1j_k}, a_{2k_k}, \ldots, a_{nj_k})$ of $T[A_j]$ such that $a_{ij_k} \in S$, for $1 \le i \le n$. That is, $|R_k[A_j]| \le |S| = l$, where

$$R_k[A_j] = \{a_{ij_k} \mid 1 \le i \le n\}.$$

By the result of Lemma 3.1, $|R_k[A_j]| \ge l$. Therefore, we conclude $|R_k[A_j]| = l \in \mathcal{V}(count(T.A_j))$.

To show that $u \in \mathcal{V}(count(T.A_j))$, we can generate an interpretation $\alpha_k = (a_{1j_k}, a_{2j_k}, \ldots, a_{nj_k})$ of $T[A_j]$ such that

$$a_{ij_k} = \begin{cases} a & \text{if } (a, \eta_{ij}) \in M, \\ \text{any } a \in v(\eta_{ij}) & \text{otherwise.} \end{cases}$$

Then, we obtain $|R_k[A_j]| \ge |M| = u$. By the result of Lemma 3.2, $|R_k[A_j]| \le u$. Therefore, we conclude $|R_k[A_j]| = u \in \mathcal{V}(count(T.A_j))$.

Finally, we need to show that all integers between l and u are elements of $\mathcal{V}(count(T.A_j))$. Let $V(M)$ be the set of vertices in M. Define $U = V(M) \cap X$, then $|U| = u$. For each c, $l < c < u$, there exists a set S', $S \subset S' \subset U \subseteq X$, with $|S'| = c$. Since $u \in \mathcal{V}(count(T.A_j))$, by Lemma 3.3, there is a complete matching from U to Y in the membership graph of Y over U. By Theorem 3.1, there is also a complete matching from S' to Y in the membership graph of Y over S'. Moreover, since S covers Y in G, S' also covers Y in G. By Lemma 3.3, we conclude that $|S'| \in \mathcal{V}(count(T.A_j))$. □

By Theorem 3.2, for a set of partial values $T[A_j] = Y = \{\eta_{1j}, \eta_{2j}, \ldots, \eta_{nj}\}$, let $X = \bigcup_{1\le i\le n} \mathcal{V}(\eta_{ij})$. We can first construct the membership graph of Y over X, $G = (X \cup Y, E)$, then compute the minimum set S, $S \subseteq X$, that covers Y in G and compute the maximum matching M in G, then $\mathcal{V}(count(T.A_j)) = \{c \mid l \le c \le u\}$, where l and u denote the cardinality of A and M, respectively. Although the minimum cover problem is NP-complete [13], the computation of a minimal set S can be done as follows.

ALGORITHM 3.1. An Algorithm for Computing the Minimal Set $S \subseteq X$ That Covers Y in a Membership Graph $G = (X \cup Y, E)$.

Input: A Membership Graph $G = (X \cup Y, E)$.

Output: The Minimal Subset S of X That Covers Y in G.

```
1) $S = \emptyset$;
2) while ($Y \neq \emptyset$) do {
3) Choose a vertex $a$ from $X$ such that
       $|N(\{a\})| = \max_{a_i \in X} |N(\{a_i\})|$;
4) $S = S \cup \{a\}$;
5) $E = E - \{(x, y) \mid x \in X \wedge y \in N(\{a\})\}$;
6) $Y = Y - N(\{a\})$;
7) $X = X - \{a\}$;
8) }
9) Output($S$);
```

In Example 3.1, to find $\nu(count(Employees.salary))$, we can construct the membership graph $G = (X \cup Y, E)$ as depicted in Fig. 1, where $X = \{20k, 60k, 75k, 80k, 90k\}$ and $Y = \{[20k\ 80k], [60k, 75k], [20k, 90k]\}$. Thus, one of the minimal subsets of X that covers Y is $\{20k, 60k\}$ (which is also a minimum cover set in this example, shown as the two shaded nodes) and one of the maximum matching in G is $M = \{(20k, [20k, 80k]), (60k, [60k, 75k]), (90k, [20k, 90k])\}$ (the three pairs associated with the lines of double-ended arrows). Therefore, $\nu(count(Employees.salary)) = \{2, 3\}$.

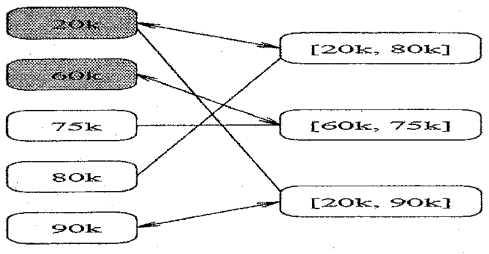


Fig. 1. The membership graph for Example 3.1.

An $O(n^{5/2})$ algorithm, where n is the number of vertices, for the computation of maximum matching in a bipartite graph has been developed by Hopcroft and Karp [17]. The time complexity for computing maximum matching dominates that for computing a minimal subset of X that covers Y in G. Therefore, the time complexity of our algorithm for *count* is $O(n^{5/2})$.

3.2.2 Properties of the Scalar Aggregates—Max and Min

In this subsection, we develop two dual properties for the scalar aggregates—*Max* and *Min*, respectively. These properties state that their corresponding scalar aggregates can be solved in linear time.

THEOREM 3.3. *Let T be a partial relation on attributes $\{A_1, A_2, \ldots, A_m\}$ containing tuples $\{t_1, t_2, \ldots, t_n\}$.*

$$\nu(max(T.A_j)) = \left\{ a \,\middle|\, a \in \bigcup_{t_i \in T} \nu(\eta_{ij}), a \geq \max_{t_i \in T} \min \nu(\eta_{ij}) \right\}.$$

PROOF. By Definition 3.4,

$$\nu(max(T.A_j)) = \left\{ \max_{t_i \in R_k} a_{ij_k} \,\middle|\, R_k \in rep(T) \right\}.$$

Let $\mathcal{A} = \{\max_{t_i \in R_k} a_{ij_k} \mid R_k \in rep(T)\}$ and

$$\mathcal{B} = \left\{ a \,\middle|\, a \in \bigcup_{t_i \in T} \nu(\eta_{ij}), a \geq \max_{t_i \in T} \min \nu(\eta_{ij}) \right\}.$$

We want to show $\mathcal{A} = \mathcal{B}$.

("$\subseteq$") For any $\max_{t_i \in R_k} a_{ij_k} = a \in \mathcal{A}$, we obtain

$a =$

$\max\{a_{ij_k} \mid a_{ij_k} \in \nu(\eta_{ij}),\ t_i \in T,$ for interpretation $\alpha_k\}$

$= \max\{a_{ij_k} \mid a_{ij_k} \in \nu(\eta_{ij}),\ a_{ij_k} \geq \min \nu(\eta_{ij}),$
$t_i \in T,$ for interpretation $\alpha_k\}$

$= \max\{a_{ij_k} \mid a_{ij_k} \in \nu(\eta_{ij}), a_{ij_k} \geq \max_{t_i \in T} \min \nu(\eta_{ij}),$
$t_i \in T,$ for interpretation $\alpha_k\}$

$\in \{a \mid a \in \bigcup_{t_i \in T} \nu(\eta_{ij}),\ a \geq \max_{t_i \in T} \min \nu(\eta_{ij})\} = \mathcal{B}.$

("$\supseteq$") Assume $a \in \mathcal{B}$, then we obtain

$$a \geq \max_{t_i \in T} \min \nu(\eta_{ij})$$

and $a \in \bigcup_{t_i \in T} \nu(\eta_{ij})$. Hence, there exists an $\eta_{lj} \in T[A_j]$ such that $a \in \nu(\eta_{lj})$. Therefore, there exists α_k subject to

$$a_{ij_k} = \begin{cases} a & \text{if } i = l, \\ \min \nu(\eta_{ij}) & \text{otherwise.} \end{cases}$$

By

$$a \geq \max_{t_i \in T} \min \nu(\eta_{ij})$$

we obtain $a = \max_{t_i \in R_k} a_{ij_k}$. By $\max_{t_i \in R_k} a_{ij_k} \in \mathcal{A}$, we obtain $a \in \mathcal{A}$. □

In Example 3.1, max{min{20k, 80k}, min{60k, 75k}, min{20k, 90k}} = 60k. Therefore, $\nu(max(Employees.salary)) = \{60k, 75k, 80k, 90k\}$.

Similarly, we can obtain the dual property for $\nu(min(T.A_j))$ as follows.

THEOREM 3.4. *Let T be a partial relation on attributes $\{A_1, A_2, \ldots, A_m\}$ containing tuples $\{t_1, t_2, \ldots, t_n\}$.*

$$\nu(\min(T.A_j)) = \left\{ a \,\middle|\, a \in \bigcup_{t_i \in T} \nu(\eta_{ij}),\ a \leq \min_{t_i \in T} \max \nu(\eta_{ij}) \right\}.$$

PROOF. The proof can be obtained in an analogous way as that of Theorem 3.3 by replacing "max," "min," and "$\geq$" with "min," "max," and "$\leq$," respectively. □

In Example 3.1, min{max{20k, 80k}, max{60k, 75k}, max{20k, 90k}} = 75k. Therefore, $\nu(min(Employees.salary)) = \{20k, 60k, 75k\}$.

Let $n_\Sigma = \sum_{t_i \in T} |\nu(\eta_{ij})|$. By Theorems 3.3 and 3.4, we can compute *maximum* and *minimum* in $O(n_\Sigma)$. In comparison, the time complexity of the corresponding brute-force algorithms is $O(n_\Pi)$, where $n_\Pi = \prod_{t_i \in T} |v(\eta_{ij})|$.

4 Extended Aggregate Functions over Partial Values

4.1 Definitions of the Extended Aggregate Functions

Extended aggregate functions can be applied to a partial relation $T[\ldots, A_x, A_y, \ldots]$, where A_x and A_y are attributes containing partial values. As a GROUP BY clause is specified, tuples in T are partitioned into groups by attribute A_x and the aggregate operation is then applied to attribute A_y. That is, they compute aggregation on attribute A_y of partial relation $T[\ldots, A_x, A_y, \ldots]$ based on the distinct scalar values a_x of attribute A_x, and return $<a_x, \eta_y>$, where η_y is the aggregation of A_y based on a_x. For databases containing partial values, the extended aggregate functions produce η_y as a partial value. We use the same operator names as in Section 3.1, but each extended with a *by* clause.

DEFINITION 4.1. *For a partial relation $T[\ldots, A_x, A_y, \ldots]$, the extended count on $T.A_y$ by $T.A_x$, denoted count($T.A_y$ by $T.A_x$), is defined as:*

$$count(T.A_y \ by \ T.A_x) = \{< a_x, \eta_y > | \ a_x \in \bigcup_{t \in T} v(t.A_x) \wedge$$

$$v(\eta_y) = \{ |S_k[A_y]| \ | \ R_k \in \text{rep}(T) \} \}$$

where $S_k = \{t \mid t \in R_k \wedge t.A_x = a_x\}$.

DEFINITION 4.2. *For a partial relation $T[\ldots, A_x, A_y, \ldots]$, the extended sum on $T.A_y$ by $T.A_x$, denoted sum($T.A_y$ by $T.A_x$), is defined as:*

$$sum(T.A_y \ by \ T.A_x) = \{< a_x, \eta_y > | \ a_x \in \bigcup_{t \in T} v(t.A_x) \wedge$$

$$v(\eta_y) = \{ \sum_{t \in S_k} t.A_y \ | \ R_k \in \text{rep}(T) \wedge S_k \neq \emptyset \} \}$$

where $S_k = \{t \mid t \in R_k \wedge t.A_x = a_x\}$.

DEFINITION 4.3. *For a partial relation $T[\ldots, A_x, A_y, \ldots]$, the extended average on $T.A_y$ by $T.A_x$, denoted avg($T.A_y$ by $T.A_x$), is defined as:*

$$avg(T.A_y \ by \ T.A_x) = \{< a_x, \eta_y > | \ a_x \in \bigcup_{t \in T} v(t.A_x) \wedge$$

$$v(\eta_y) = \{ \frac{\sum_{t \in S_k} t.A_y}{|S_k|} \ | \ R_k \in \text{rep}(T) \wedge S_k \neq \emptyset \} \}$$

where $S_k = \{t \mid t \in R_k \wedge t.A_x = a_x\}$.

DEFINITION 4.4. *For a partial relation $T[\ldots, A_x, A_y, \ldots]$, the extended maximum on $T.A_y$ by $T.A_x$, denoted max($T.A_y$ by $T.A_x$), is defined as:*

$$max(T.A_y \ by \ T.A_x) = \{< a_x, \eta_y > | \ a_x \in \bigcup_{t \in T} v(t.A_x) \wedge$$

$$v(\eta_y) = \{ \max_{t \in S_k} t.A_y \ | \ R_k \in \text{rep}(T) \wedge S_k \neq \emptyset \} \}$$

where $S_k = \{t \mid t \in R_k \wedge t.A_x = a_x\}$.

DEFINITION 4.5. *For a partial relation $T[\ldots, A_x, A_y, \ldots]$, the extended minimum on $T.A_y$ by $T.A_x$, denoted min($T.A_y$ by $T.A_x$), is defined as:*

$$min(T.A_y \ by \ T.A_x) = \{< a_x, \eta_y > | \ a_x \in \bigcup_{t \in T} v(t.A_x) \wedge$$

$$v(\eta_y) = \{ \min_{t \in S_k} t.A_y \ | \ R_k \in \text{rep}(T) \wedge S_k \neq \emptyset \} \}$$

where $S_k = \{t \mid t \in R_k \wedge t.A_x = a_x\}$.

The following example illustrates these extended aggregate functions.

EXAMPLE 4.1. Consider the following partial relation.

Employees

name	pos	salary
a	Mgr	[40k, 45k]
b	[Mgr, Engr]	32k
c	Engr	[20k, 32k]

rep(Employees) contains the following elements:

a	Mgr	40k
b	Mgr	32k
c	Engr	20k

a	Mgr	40k
b	Mgr	32k
c	Engr	32k

a	Mgr	45k
b	Mgr	32k
c	Engr	20k

a	Mgr	45k
b	Mgr	32k
c	Engr	32k

a	Mgr	40k
b	Engr	32k
c	Engr	20k

a	Mgr	40k
b	Engr	32k
c	Engr	32k

a	Mgr	45k
b	Engr	32k
c	Engr	20k

a	Mgr	45k
b	Engr	32k
c	Engr	32k

The aggregate functions over *Employees.salary* by *Employees.pos* are as follows.

sum(Employees.salary by Employees.pos) =

pos	salary
Mgr	[72k, 77k, 40k, 45k]
Engr	[20k, 32k, 52k, 64k]

count(Employees.salary by Employees.pos) =

pos	salary
Mgr	[1, 2]
Engr	[1, 2]

avg(Employees.salary by Employees.pos) =

pos	salary
Mgr	$[\frac{72k}{2}, \frac{77k}{2}, \frac{40k}{1}, \frac{45k}{1}]$
Engr	$[\frac{20k}{1}, \frac{32k}{1}, \frac{52k}{2}, \frac{64k}{2}]$

max(Employees.salary by Employees.pos) =

pos	salary
Mgr	[40k, 45k]
Engr	[20k, 32k]

min(Employees.salary by Employees.pos) =

pos	salary
Mgr	[32k, 40k, 45k]
Engr	[20k, 32k]

Note that in this example the result of *avg* is $[\frac{20k}{1}, \frac{32k}{1}, \frac{52k}{2}, \frac{64k}{2}]$, $\frac{64k}{2}$ can be removed since $\frac{64k}{2} = \frac{32k}{1}$.

4.2 Properties and Algorithms for Evaluating Aggregate Functions over Partial Values

In the following, we develop polynomial time algorithms to compute *count, maximum* and *minimum* based on their properties. These properties are generalized from those presented in the previous section.

4.2.1 Properties of the Aggregate Function—Count

Before going through the details, we need the following definitions.

DEFINITION 4.6. *Let* $T[\ldots, A_x, A_y, \ldots]$ *be a partial relation and* $R = \bigcup_{R_k \in \text{rep}(T)} R_k$. *The* marginal membership graph of $T[A_x, A_y]$ over $R[A_x, A_y]$ with respect to (w.r.t.) a_x, *where* $a_x \in \bigcup_{t\in T} \mathcal{V}(t.A_x)$, *is a bipartite graph* $H_{a_x} = (U \cup V, E)$, *where*

$$U = \{\tau \mid \tau \in R[A_x, A_y],\ \tau.A_x = a_x\},$$
$$V = \{t \mid t \in T[A_x, A_y],\ a_x \in v(t.A_x)\}, \text{ and}$$
$$E = \{(\tau, t) \mid \tau \in \text{rep}(t),\ \tau \in U,\ t \in V\}.$$

DEFINITION 4.7. *Let* $T[\ldots, A_x, A_y, \ldots]$ *be a partial relation and* $R = \bigcup_{R_k \in \text{rep}(T)} R_k$. *The* strict marginal membership graph of $T[A_x, A_y]$ over $R[A_x, A_y]$ w.r.t. a_x, *where* $a_x \in \bigcup_{t\in T} \mathcal{V}(t.A_x)$, *is a bipartite graph* $\mathcal{H}_{a_x} = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$, *where*

$$\mathcal{U} = \{\tau \mid \tau \in R[A_x, A_y],\ \tau.A_x = a_x\},$$
$$\mathcal{V} = \{t \mid t \in T[A_x, A_y],\ t.A_x = a_x\}, \text{ and}$$
$$\mathcal{E} = \{(\tau, t) \mid \tau \in \text{rep}(t),\ \tau \in \mathcal{U},\ t \in \mathcal{V}\}.$$

According to our definitions, for an H_{a_x} and $\mathcal{H}_{a_x}$, it is always true that $\mathcal{U} = U$ and $\mathcal{V} \subseteq V$. Therefore, for each a_x, $\mathcal{H}_{a_x}$ is always a subgraph of H_{a_x}.

We now state the properties for *count* as follows.

LEMMA 4.1. *For a partial relation* $T[\ldots, A_x, A_y, \ldots]$ *and each* $a_x \in \bigcup_{t\in T} \mathcal{V}(t.A_x)$, *let* $R = \bigcup_{R_k \in \text{rep}(T)} R_k$, $\mathcal{H}_{a_x} = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$ *be the* strict marginal membership graph of $T[A_x, A_y]$ over $R[A_x, A_y]$ w.r.t. a_x, *and* l_{a_x} *denote the cardinality of the minimum* S, $S \subseteq \mathcal{U}$, *that covers* $\mathcal{V}$ *in* $\mathcal{H}_{a_x}$, *then* $<a_x, \eta_y> \in$ *count*$(T.A_y$ *by* $T.A_x)$ *where, for all* $c \in \mathcal{V}(\eta_y)$, $c \ge l_{a_x}$.

PROOF. Suppose, for a $<a_x, \eta_y> \in$ *count*$(T.A_y$ *by* $T.A_x)$, there is an element c of $\mathcal{V}(\eta_y)$ such that $c < l_{a_x}$. That is, there is an R_k with $R_k[A_x, A_y] = \mathcal{M}_k \cup \mathcal{N}_k$, where

$$\mathcal{M}_k = \{< a_x, a_{y_q} >, < a_x, a_{y_2} >, \ldots, < a_x, a_{y_c} >\},$$

and

$$\mathcal{N}_k = \{< \bar{a}_x, a_{y_{c+1}} >, < \bar{a}_x, a_{y_{c+2}} >, \ldots\}$$

where $\bar{a}_x$ denotes those elements which are not a_x. Since, for each $t \in \mathcal{V}$, $t.A_x = a_x$, we obtain there is a $< a_x, a_{y_j} > \in \mathcal{M}_k$ such that $< a_x, a_{y_j} > \in \text{rep}(t)$, which implies $\mathcal{M}_k$ covers $\mathcal{V}$ in $\mathcal{H}_{a_x}$. But $|\mathcal{M}_k| = c < l_{a_x}$, which contradicts that the cardinality of the minimum set that covers $\mathcal{V}$ in $\mathcal{H}_{a_x}$ is l_{a_x}. Hence, this lemma holds. □

LEMMA 4.2. *For a partial relation* $T[\ldots, A_x, A_y, \ldots]$ *and each* $a_x \in \bigcup_{t\in T} \mathcal{V}(t.A_x)$, *let* $R = \bigcup_{R_k \in \text{rep}(T)} R_k$, $H_{a_x} = (U \cup V, E)$ *be the* marginal membership graph of $T[A_x, A_y]$ over R w.r.t. a_x, *and* u_{a_x} *denote the cardinality of the maximum matching* M *in* H_{a_x}, *then* $<a_x, \eta_y> \in$ *count*$(T.A_y$ *by* $T.A_x)$ *where, for all* $c \in \mathcal{V}(\eta_y)$, $c \le u_{a_x}$.

PROOF. Suppose, for a $<a_x, \eta_y> \in$ *count*$(T.A_y$ *by* $T.A_x)$, there is an element $c \in \mathcal{V}(\eta_y)$ such that $c > u_{a_x}$. That is, there is an $R_k \in \text{rep}(T)$ with $R_k[A_x, A_y] = M_k \cup N_k$, where

$$M_k = \{< a_x, a_{y_1} >, < a_x, a_{y_2} >, \ldots, < a_x, a_{y_c} >\},$$

and

$$N_k = \{< \bar{a}_x, a_{y_{c+1}} >, < \bar{a}_x, a_{y_{c+2}} >, \ldots\},$$

where $\bar{a}_x$ denotes those elements which are not a_x. In other words, for each $< a_x, a_{y_j} > \in M_k$, there exists at least a distinct $t \in T[A_x, A_y]$ such that

$$< a_x, a_{y_j} > \in \text{rep}(t)$$

(i.e., $a_x \in \mathcal{V}(t.A_x)$ and $a_{y_j} \in v(t.A_y)$). That implies, for all $S \subseteq M_k$, $|S| \le |N(S)|$ in the membership graph of V over M_k, $G' = (M_k \cup V, E')$. Therefore, by Theorem 3.1, there exists a complete matching M^* from M_k to V in G'. Besides, since $M_k \subseteq U$, M^* is also a matching in H_{a_x}. But $|M^*| = |M_k| = c > u_{a_x}$, which contradicts that the maximum matching in H_{a_x} has cardinality u_{a_x}. Therefore, this lemma holds. □

LEMMA 4.3 *For a partial relation* $T[\ldots, A_x, A_y, \ldots]$ *and each* $a_x \in \bigcup_{t\in T} \mathcal{V}(t.A_x)$, let

1) $R = \bigcup_{R_k \in \text{rep}(T)} R_k$,
2) $H_{a_x} = (U \cup V, E)$ *be the* marginal membership graph of $T[A_x, A_y]$ over $R[A_x, A_y]$ *w.r.t.* a_x, *and*
3) $\mathcal{H}_{a_x} = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$ *be the* strict marginal membership graph of $T[A_x, A_y]$ over $R[A_x, A_y]$ w.r.t. a_x *(Recall that, according to our definitions,* $\mathcal{U} = U$ *and* $\mathcal{V} \subseteq V$*).*

Then, there is a complete matching from S, $S \subseteq U$, *to* V *in* *the membership graph of V over S, G′ = (S ∪ V, E′) (a subgraph of H_{a_x}), and S covers $\mathcal{V}$ in $\mathcal{H}_{a_x}$ if and only if $<a_x, \eta_y> \in count(T.A_y$ by $T.A_x)$ and $|S| \in \nu(\eta_y)$.*

PROOF. (⇐) For a tuple $<a_x, \eta_y> \in count(T.A_y$ by $T.A_x)$, suppose $|S| = c \in \nu(\eta_y)$. That is, there is an $R_k \in$ rep(T) with $R_k[A_x, A_y] = M_k \cup N_k$, where

$$M_k = S = \{< a_x, a_{y_1} >, < a_x, a_{y_2} >, \ldots, < a_x, a_{y_c} >\}$$

and

$$N_k = \{< \bar{a}_x, a_{y_{c+1}} >, < \bar{a}_x, a_{y_{c+2}} >, \ldots\}$$

where $\bar{a}_x$ denotes those elements which are not a_x.

In other words, for each $< a_x, a_{y_j} > \in S$, there exists at least a distinct $t \in T[A_x, A_y]$ such that $< a_x, a_{y_j} > \in$ rep(t) (i.e., $a_x \in \nu(t.A_x)$ and $a_{y_j} \in \nu(t.A_y)$). That implies, for all $S' \subseteq S$, $|S'| \leq |N(S')|$ in $G' = (S \cup V, E')$. Therefore, by Theorem 3.1, there exists a complete matching M^* from S to V in G'. Since, for each $t \in \mathcal{V}$, $t.A_x = a_x$, we obtain there is a $< x, a_{y_j} > \in S$, such that $< a_x, a_{y_j} > \in$ rep(t), which implies S covers $\mathcal{V}$ in $\mathcal{H}_{a_x}$.

(⇒) Assume there is a complete matching M from S to V in G' and S covers $\mathcal{V}$ in $\mathcal{H}_{a_x}$ where

$$S = \{< a_x, a_{y_1} >, < a_x, a_{y_2} >, \ldots\} \subseteq U.$$

To show that $|S| \in \nu(\eta_y)$, we can generate an interpretation

$$\alpha_k(T[A_x, A_y]) = \{< a_{1x_k}, a_{iy_k} >, < a_{2x_k}, a_{2y_k} >, \ldots < a_{nx_k}, a_{ny_k} >\}$$

such that

$$< a_{ix_k}, a_{iy_k} > = \begin{cases} < a_x, a_{y_l} > & \text{if } (< a_x, a_{y_l} >, t_i) \in M, \\ \text{any } < a_x, a_{y_l} > \in (S \cap \text{rep}(t_i)), & \text{if } (< a_x, a_{y_l} >, t_i) \notin M \text{ and } t_i \in \mathcal{V}, \\ \text{any } < a_{ix_l}, a_{iy_l} > \in \text{rep}(t_i) \text{ with } a_{ix_l} \neq a_x & \text{if } t_i \in T[A_x, A_y] - \mathcal{V} \end{cases}$$

Then, we obtain $|\{t_i.A_y \mid t_i \in R_k, t_i.A_x = a_x\}| = |S| \in \nu(\eta_y)$. □

THEOREM 4.1. *For a partial relation T[..., A_x, A_y, ...] and each $a_x \in \bigcup_{t \in T} \nu(t.A_x)$, let*

1) $R = \bigcup_{R_k \in \text{rep}(T)} R_k$,
2) $\mathcal{H}_{a_x} = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$ *be the* strict marginal membership graph of $T[A_x, A_y]$ over $R[A_x, A_y]$ w.r.t. a_x,
3) l_{a_x} *denote the cardinality of the minimum set S, $S \subseteq \mathcal{U}$, that covers $\mathcal{V}$ in $\mathcal{H}_{a_x}$,*
4) $H_{a_x} = (U \cup V, E)$ *be the* marginal membership graph of $T[A_x, A_y]$ over $R[A_x, A_y]$ w.r.t. a_x, *and*
5) u_{a_x} *denote the cardinality of the maximum matching M in H_{a_x},*

then $<a_x, \eta_y> \in count(T.A_y$ by $T.A_x)$ and

$$\nu(\eta_y) = \{c \mid l_{a_x} \leq c \leq u_{a_x}\}.$$

PROOF. By Lemmas 4.1 and 4.2, we obtain, for each $<a_x, \eta_y> \in count(T.A_y$ by $T.A_x)$, $l_{a_x} \leq c \leq u_{a_x}$, for all $c \in \nu(\eta_y)$. Now we need to show every integer between l_{a_x} and u_{a_x} (including l_{a_x} and u_{a_x}) is an element of $\nu(\eta_y)$.

We first show that $l_{a_x} \in \nu(\eta_y)$. Since S covers $\mathcal{V}$ in $\mathcal{H}_{a_x}$, we can generate an interpretation

$$\alpha_k(T[A_x, A_y]) = \{< a_{1x_k}, a_{1y_k} >, < a_{2x_k}, a_{2y_k} >, \ldots, < a_{nx_k}, a_{ny_k} >\}$$

such that

$$< a_{ix_k}, a_{iy_k} > = \begin{cases} < a_x, a_{iy_l} > \in (S \cap \text{rep}(t_i)), & \text{if } t_i \in \mathcal{V}, \\ \text{any } < a_{ix_l}, a_{iy_l} > \in \text{rep}(t_i) \text{ with } a_{ix_l} \neq a_x & \text{if } t_i \in T - \mathcal{V}. \end{cases}$$

That is, $\left|\{t_i.A_y \mid t_i \in R_k, t_i.A_x = a_x\}\right| = c \leq |S| = l_{a_x}$. By the result of Lemma 4.1, $c \geq l_{a_x}$, we conclude

$$c = l_{a_x} \in \nu(\eta_y).$$

To show that $u_{a_x} \in \nu(\eta_y)$ by employing the maximum matching M in $H_{a_x} = (U \cup V, E)$, we can generate an interpretation

$$\alpha_k(T[A_x, A_y]) = \{< a_{1x_k}, a_{1y_k} >, < a_{2x_k}, a_{2y_k} >, \ldots, < a_{nx_k}, a_{ny_k} >\}$$

such that

$$< a_{ix_k}, a_{iy_k} > = \begin{cases} < a_x, a_{iy_l} > & \text{if } (< a_x, a_{y_l} >, t_i) \in M, \\ \text{any } < a_{ix_l}, a_{iy_l} > \in \text{rep}(t_i),\ t_i \in T[A_x, A_y] & \text{otherwise.} \end{cases}$$

Then, we obtain

$$\left|\{t_i.A_y \mid t_i \in R_k, t_i.A_x = a_x\}\right| = c \geq |M| = u_{a_x}.$$

By the result of Lemma 4.2, $c \leq u_{a_x}$. Therefore, we conclude $c = u_{a_x} \in \nu(\eta_y)$.

Finally, we need to show that all integers between l_{a_x} and u_{a_x} are elements of $\nu(\eta_y)$. Let $V(M)$ be the set of vertices in M. Define $W = V(M) \cap U$, then $|W| = u_{a_x}$. For each c, $l_{a_x} < c < u_{a_x}$, there exists a set S', $S \subset S' \subset W \subseteq U$, with $|S'| = c$. Since $u_{a_x} \in \nu(\eta_y)$, by Lemma 4.3, there is a complete matching from W to V in the membership graph of V over W. By Theorem 3.1, there is also a complete matching from S' to V in the membership graph of V over S'. Moreover, since S covers $\mathcal{V}$ in $\mathcal{H}_{a_x}$, S' also covers $\mathcal{V}$ in $\mathcal{H}_{a_x}$. By Lemma 4.3, we conclude that $|S'| = c \in \nu(\eta_y)$. □

Theorem 4.1 presents an algorithm for computing the answer of the aggregate function *count* over partial values. That is, for a partial relation T[..., A_x, A_y, ...], and each $a_x \in \bigcup_{t \in T} \nu(t.A_x)$, let $R = \bigcup_{R_k \in \text{rep}(T)} R_k$, we can

1) construct
   - $\mathcal{H}_{a_x} = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$ as the strict marginal membership graph of $T[A_x, A_y]$ over $R[A_x, A_y]$ w.r.t. a_x and
   - $H_{a_x} = (U \cup V, E)$ as the marginal membership graph of $T[A_x, A_y]$ over $R[A_x, A_y]$ w.r.t. a_x, then
2) compute
   - l_{a_x}, which denotes the cardinality of the minimum set S, $S \subseteq \mathcal{U}$, that covers $\mathcal{V}$ in $\mathcal{H}_{a_x}$ and
   - u_{a_x}, which denotes the cardinality of the maximum matching M in H_{a_x}.

Then $<a_x, \eta_y> \in count(T.A_y \; by \; T.A_x)$ and

$$v(\eta_y) = \{c \mid l_{a_x} \le c \le u_{a_x}\}.$$

Although the minimum cover problem is NP-complete [13], the computation of a minimal set S can be obtained by Algorithm 3.1 with input parameter $\mathcal{H}_{a_x}$.

In Example 4.1, to find *count(Employees.salary by Employees.pos)*, we can construct $H_{\text{Mgr}} = (U \cup V, E)$ as depicted in Fig. 2a, where U = {<Mgr, 40k>, <Mgr, 45k>, <Mgr, 32k>} and V = {<Mgr, [40k, 45k]>, <Mgr, Engr], 32k>}. Besides, $\mathcal{H}_{\text{Mgr}} = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$ is as depicted in Fig. 2b, where $\mathcal{U}$ = {Mgr, 40k>, <Mgr, 45k>, <Mgr, 32k>} and $\mathcal{V}$ = {<Mgr, [40k, 45k]>}. Thus, one of the minimal subset of $\mathcal{U}$ that covers $\mathcal{V}$ is {<Mgr, 40k>} (which is also a minimum cover set in this example, shown as the shaded node in Fig. 2b) and one of the maximum matching in H_{Mgr} is M = {(<Mgr, 40k>, <Mgr, [40k, 45k]>), (<Mgr, 32k>, <[Mgr, Engr}], 32k>)} (the two pairs associated with the lines of double-ended arrows in Fig. 2a). Therefore, <Mgr, [1, 2]> ∈ *count(Employees.salary by Employees.pos)*. The other answer tuple for 'Engr' can be computed accordingly.

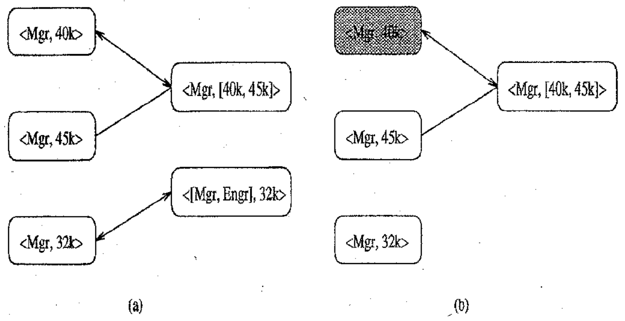


Fig. 2. (a) The marginal membership graph and (b) the strict marginal membership graph for computing *count* in Example 4.1.

Since our algorithm for *count* is also dominated by the maximum matching algorithm for each element in $\bigcup_{t\in T} v(t.A_x)$, the time complexity for computing $count(T.A_y \; by \; T.A_x)$ is $O(m \cdot n^{5/2})$, where $m = |\bigcup_{t\in T} v(t.A_x)|$ and n is the number of vertices.

4.2.2 *Properties of the Aggregate Functions—Max and Min*

In this subsection, we develop two dual properties for the aggregate functions—*Max* and *Min*, respectively. Let max $\varnothing = \infty$ and min $\varnothing = -\infty$.

THEOREM 4.2 *For a partial relation* $T[\ldots, A_x, A_y, \ldots]$,

$$\max(T.A_y \; by \; T.A_x) = \{<a_x, \eta_y> \mid a_x \in \bigcup_{t\in T} v(t.A_x) \wedge v(\eta_y) = \{a \mid a \in \bigcup_{t\in T \wedge a_x \in v(t.A_x)} v(t.A_y) \wedge a \ge \max_{t\in T \wedge t.A_x = a_x} \min v(t.A_y)\}\}.$$

PROOF. For a given $a_x \in \bigcup_{t\in T} v(t.A_x)$, let

$$\mathcal{A} = \{\max_{t\in R_k \wedge t.A_x = a_x} t.A_y \mid R_k \in \text{rep}(T)\} \text{ and}$$

$$\mathcal{B} = \{a \mid a \in \bigcup_{t\in T \wedge a_x \in v(t.A_x)} v(t.A_y),\ a \ge \max_{t\in T \wedge t.A_x = a_x} \min v(t.A_y)\}.$$

We want to show $\mathcal{A} = \mathcal{B}$.

("⊆") For any $\max_{t\in R_k \wedge t.A_x = a_x} t.A_y \in \mathcal{A}$, $R_k \in \text{rep}(T)$, we obtain

$$\begin{aligned}
&\max_{t\in R_k \wedge t.A_x = a_x} t.A_y \\
&= \max\{t_i.A_y \mid t_i \in R_k, t_i.A_x = a_x\} \\
&= \max\{a_{iy_k} \mid a_{iy_k} \in v(t_i.A_y), t_i \in T,\ a_{ix_k} = a_x\} \\
&= \max\{a_{iy_k} \mid a_{iy_k} \in v(t_i.A_y), t_i \in T, a_{ix_k} = a_x, a_{iy_k} \ge \min v(t_i.A_y)\} \\
&= \max\{a_{iy_k} \mid a_{iy_k} \in v(t_i.A_y), t_i \in T, a_{ix_k} = a_x, a_{iy_k} \ge \max_{t\in T \wedge t.A_x = a_x} \min v(t.A_y)\} \\
&\in \max\{a \mid a \in \bigcup_{t\in T \wedge a_x \in v(t.A_x)} v(t.A_y), a \ge \max_{t\in T \wedge t.A_x = a_x} \min v(t.A_y)\} \\
&= \mathcal{B}.
\end{aligned}$$

("⊇") Assume $a \in \mathcal{B}$, then we obtain

$$a \in \bigcup_{t\in T \wedge a_x \in v(t.A_x)} v(t.A_y) \text{ and a} \ge \max_{t\in T \wedge t.A_x = a_x} \min v(t.A_y).$$

Hence, there exists a $t_l \in T$ such that $a_x \in v(t_l.A_x)$ and $a \in v(t_l.A_y)$. Therefore, there exists an $R_k \in \text{rep}(T)$ such that, for all $a_{ix_k} = a_x$,

$$a_{iy_k} = \begin{cases} a & \text{if } i = l, \\ \min v(t_i.A_y) & \text{otherwise.} \end{cases}$$

By

$$a \ge \max_{t\in T \wedge t.A_x = a_x} \min v(t.A_y) \ge \min_{t\in T \wedge a_x \in v(t.A_x)} v(t.A_y),$$

we obtain $\max_{t_i \in R_k \wedge a_{ix_k} = a_x} a_{iy_k} = a$. Therefore, $a \in \mathcal{A}$. □

In Example 4.1, for 'Mgr,'

$$\max_{t\in Employees \wedge t.pos = \text{'Mgr'}} \min v(t.salary) = \max\{\min\{40k, 45k\}\} = 40k.$$

Therefore, <Mgr, [40k, 45k]> ∈ max(*Employees.salary by Employees.pos*). For 'Engr,'

$$\max_{t\in Employees \wedge t.pos = \text{'Engr'}} \min v(t.salary) = \max\{\min\{20k, 32k\}\} = 20k.$$

Therefore, <Engr, [20k, 32k, 32k> ∈ max(*Employees.salary by Employees.pos*).

Similarly, for a partial relation $T[\ldots, A_x, A_y, \ldots]$, we can obtain the dual property for $\min(T.A_y \textit{ by } T.A_x)$ as follows.

THEOREM 4.3. *For a partial relation* $T[\ldots, A_x, A_y, \ldots]$,

$$\min(T.A_y \textit{ by } T.A_x) = \{< a_x, \eta_y > \mid a_x \in \bigcup_{t\in T} v(t.A_x) \wedge$$
$$v(\eta_y) = \{a \mid a \in \bigcup_{t\in T \wedge a_x \in v(t.A_x)} v(t.A_y) \wedge$$
$$a \leq \min_{t\in T \wedge t.A_x = a_x} \max v(t.A_y)\}\}.$$

PROOF. The proof can be obtained in an analogous way as that of Theorem 4.2 by replacing "max," "min," and "≥" with "min," "max," and "≤," respectively. □

In Example 4.1, for 'Mgr,'

$$\min_{t\in Employees \wedge t.pos='\mathrm{Mgr}'} \max v(t.salary) = \min\{\max\{40k, 45k\}\} = 45k.$$

Therefore, <Mgr, [32k, 40k, 45k]> ∈ min(*Employees.salary by Employees.pos*). For 'Engr,'

$$\min_{t\in Employees \wedge t.pos='\mathrm{Engr}'} \max v(t.salary) = \min\{\max\{20k, 32k\}\} = 32k.$$

Therefore, <Engr, [20k, 32k, 32k]> ∈ min(*Employees.salary by Employees.pos*).

For the time complexity of *maximum* and *minimum*, let $n_\Sigma = \sum_{t\in T} |\, v(t.A_y) \,|$. By Theorems 4.2 and 4.3, we can compute *maximum* and *minimum* in $O(n_\Sigma)$ for each $a_x \in \bigcup_{t\in T} v(t.A_x)$. Therefore, the time complexity for our *minimum* and *maximum* are $O(m_\Sigma \cdot n_\Sigma)$, where $m_\Sigma = |\, \bigcup_{t\in T} v(t.A_x) \,|$.

5 Discussion and Future Studies

Partial values have been used to represent imprecise data in databases. Referring to the work of DeMichiel [11] on resolving domain mismatch problems in multidatabase systems by partial values, data imprecision may come from their unavailability or data/schema incompatibilities in a multidatabase system. Therefore, in addition to imprecise data manipulation, research work on partial values is also important for the interoperability of a multidatabase system.

Since, in practice, relational algebra or calculus are inadequate for many important applications involving statistical information or aggregations, modern query languages are equipped with some useful aggregate operations. In this paper, we define a set of extended aggregate operations, namely *sum, average, count, maximum,* and *minimum,* which can be applied to attributes containing partial values.

Since the number of the interpretations of a set of partial values can be very large, the properties presented in this paper can be used to speed up the computations of *count, maximum,* and *minimum.* By the properties presented in Section 3.2, we know that *maximum* and *minimum* can be solved in polynomial time. For *count,* we show that the lower bound is equal to the cardinality of the minimum cover set and the upper bound is equal to the cardinality of the maximum matching. The maximum matching can be found in polynomial time while the minimum cover problem is NP-complete. A greedy algorithm is given to find a minimal cover set. For *sum* and *average,* we point out that the cardinalities of $v(sum(T.A))$ and $v(avg(T.A))$ are equal to the cardinality of $\prod_{t\in T} |\, v(t.A) \,|$ in the worst case and no polynomial time algorithms can be found.

The *count* operation actually corresponds to those in SQL with the keyword DISTINCT. For those in SQL with the keyword ALL, *count* can be obtained by directly counting the number of tuples in the involved relation.

Although Imielinski and Vadaparty [18], [19] pointed out that if partial values are allowed to occur in databases, the data complexity of query processing jumps from polynomial time to CoNP [13], there also exist some types of queries which have polynomial time complexity [18]. Our studies on the query processing over partial values intend to discover more polynomial time algorithms from algebraic point of view. In our recent work, we found the *division* (by restricting the divisor to be definite) [31] and the *projection with redundancy elimination* [29] over partial values can be done in polynomial time. Besides, we generalized partial values into *probabilistic partial values* in [30]. Similar properties for probabilistic partial values are also being studied.

Acknowledgments

The authors would like to thank the anonymous referees for their valuable suggestions and comments.

This research was partially supported by the Republic of China National Science Council under Contract No. NSC 84-2213-E-007.

References

[1] D. Barbará, H. Garcia-Molina, and D. Porter, "The Management of Probabilistic Data," *IEEE Trans. Knowledge and Data Engineering,* vol. 4, no. 4, pp. 487-501, 1992.
[2] J.A. Bondy and U.S.R. Murty, *Graph Theory with Applications.* NewYork: Macmillan, 1976.
[3] P. Bosc, M. Galibourg, and G. Hamon, "Fuzzy Querying with SQL: Extensions and Implementation Aspects," *Fuzzy Sets and Systems,* vol. 28, pp. 333-349, 1988.
[4] B.P. Buckles and F.E. Petry, "A Fuzzy Representation of Data for Relational Databases," *Fuzzy Sets and Systems,* vol. 7, pp. 213-226, 1982.
[5] R. Cavallo and M. Pittarelli, "The Theory of Probabilistic Databases," *Proc. Very Large Data Bases Conf.,* pp. 71-81, 1987.
[6] E.F. Codd, "Understanding Relations," Installment no. 7, *ACM SIGMOD Record FDT Bulletin,* vol. 7, no. 3-4, pp. 23-28, 1975.
[7] E.F. Codd, "Extending the Database Relational Model to Capture More Meaning," *ACM Trans. Database Systems,* vol. 4, no. 4, pp. 397-434, 1979.
[8] E.F. Codd, "Missing Information (Applicable and Inapplicable) in Relational Databases," *ACM SIGMOD Record,* vol. 15, no. 4, pp. 53-78, 1986.
[9] C.J. Date, *A Guide to the SQL Standard.* Reading, Mass.: Addison-Wesley, 1989.
[10] L.G. DeMichiel, "Performing Database Operations over Mismatched Domains," PhD dissertation, Stanford Univ., 1989.
[11] L.G. DeMichiel, "Resolving Database Incompatibility: An Approach to Performing Relational Operations over Mismatched Domains," *IEEE Trans. Knowledge and Data Engineering,* vol. 1, no. 4, pp. 485-493, 1989.
[12] D. Dubois and H. Prade, *Possibility Theory: An Approach to Computerized Processing.* New York: Plenum Press, 1986.
[13] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco: Freeman, 1979.

[14] J. Grant, "Partial Values in a Tabular Database Model," *Information Processing Letters*, vol. 9, no. 2, pp. 97-99, 1979.

[15] J. Grant and J. Minker, "Answering Queries in Indefinite Databases and the Null Value Problem," *Advances in Computing Research*, P. Kanellakis, ed., vol. 3, *The Theory of Databases*, pp. 247-267. JAI Press, 1986.

[16] P. Hall, "On Representatives of Subsets," *J. London Math. Soc.*, vol. 10, pp. 26-30, 1935.

[17] J.E. Hopcroft and R.M. Karp, "An $n^{5/2}$ Algorithm for Maximum Matching in Bipartite Graphs," *SIAM J. Computing*, vol. 2, no. 4, pp. 225-231, 1973.

[18] T. Imielinski and K. Vadaparty, "Complexity of Query Processing in Databases with OR-Objects," *Proc. ACM Symp. Principles of Database Systems*, pp. 51-65, 1989.

[19] T. Imielinski, "Incomplete Deductive Databases," *Annals of Mathematics and Artificial Intelligence*, vol. 3, no. 2-4, pp. 259-294, 1984.

[20] W. Lipski, "On Semantic Issues Connected with Incomplete Information Systems," *ACM Trans. Database Systems*, vol. 4, no. 3, pp. 262-296, 1979.

[21] D. Maier, *The Theory of Relational Databases*. Rockville, Md.: Computer Science Press, 1983.

[22] A. Motro, "Accommodating Imprecision in Database Systems: Issues and Solutions," *ACM SIGMOD RECORD*, vol. 19, no. 4, pp. 69-74, 1990.

[23] A. Ola, "Relational Databases with Exclusive Disjunctions," *Proc. Eighth IEEE Int'l Conf. Data Engineering*, pp. 328-336, 1992.

[24] G. Özsoyoglu, Z.M. Özsoyoglu, and V. Matos, "Extending Relational Algebra and Relational Calculus with Set-Valued Attributes and Aggregate Functions," *ACM Trans. Database Systems*, vol. 12, no. 4, pp. 566-592, 1987.

[25] H. Prade and C. Testemale, "Generalizing Database Relational Algebra for the Treatment of Incomplete/Uncertain Information and Vague Queries," *Information Sciences*, vol. 34, pp. 115-143, 1984.

[26] E.A. Rundensteiner and L. Bic, "Aggregates in Possibilistic Databases," *Proc. 15th Int'l Conf. Very Large Data Bases*, pp. 287-294, 1989.

[27] E.A. Rundensteiner and L. Bic, "Evaluating Aggregates in Possibilistic Relational Databases," *Data & Knowledge Engineering*, vol. 7, no. 3, pp. 239-267, 1992.

[28] P.S.M. Tsai and A.L.P. Chen, "Querying Uncertain Data in Heterogeneous Databases," *Proc. IEEE Int'l Workshop Research Issues on Data Engineering (RIDE)*, pp. 161-168, 1993.

[29] F.S.C. Tseng, A.L.P. Chen, and W.P. Yang, "Searching a Minimal Semantically-Equivalent Subset of a Set of Partial Values," *The VLDB J.*, vol. 2, no. 4, pp. 489-512, 1993.

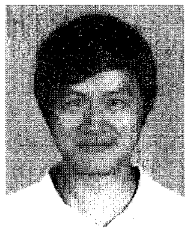
[30] F.S.C. Tseng, A.L.P. Chen, and W.P. Yang, "Answering Heterogeneous Database Queries with Degrees of Uncertainty," *Distributed and Parallel Databases: An Int'l J.*, vol. 1, no. 3, pp. 281-302, 1993.

[31] F.S.C. Tseng, A.L.P. Chen, and W.P. Yang, "Generalizing the Division Operation on Indefinite Databases," *Proc. Second Far-East Workshop Future Database Systems*, pp. 347-354, Kyoto, Japan, 1992.

[32] F.S.C. Tseng, A.L.P. Chen, and W.P. Yang, "Refining Imprecise Data by Integrity Constraints," *Data and Knowledge Engineering J.*, vol. 11, no. 3, pp. 299-316, 1993.

[33] L.A. Zadeh, "Fuzzy Sets as a Basis for a Theory of Possibility," *Fuzzy Sets and Systems*, vol. 1, no. 1, pp. 3-28, 1978.

[34] M. Zemankova and A. Kandel, "Implementing Imprecision in Information Systems," *Information Science*, vol. 37, pp. 107-141, 1985.



Arbee L.P. Chen received the BS degree in computer science from National Chiao Tung University, Republic of China, in 1977, and the PhD degree in computer engineering from the University of Southern California, Los Angeles, in 1984.

He joined National Tsing Hua University, Taiwan, as a visiting specialist in August 1990, and became a professor in the Department of Computer Science in 1991. He was a member of the technical staff at Bell Communications Research, New Jersey, from 1987 to 1990, an adjunct associate professor in the Department of Electrical Engineering and Computer Science, Polytechnic Unversity, Brooklyn, New York, and a research scientist at Unisys, Santa Monica, California, from 1985 to 1986. He is also currently director of the Computer and Communications Center of National Tsing Hua Unviersity and advisor to the Industrial Technology Research Institute and Institute for Information Industry. His research interests include heterogeneous database systems, incomplete information, object-oriented databases, and multimedia databases.

Dr. Chen is a member of the Association for Computing Machinery and the IEEE Computer Society. He has served as a program cochair for the IEEE Data Engineering Conference and a program committee member for ACM SIGMOD, VLDB, and IEEE Data Engineering Conferences. He is listed in Marquis' *Who's Who in the World.*

Jui-Shang Chiu received the MS degree and the PhD degree in computer science from National Tsing Hua University, Hsinchu, Taiwan, Republic of China, in 1989 and 1995, respectively. His research interests include databases, artificial intelligence, and algorithms.

Frank S.C. Tseng received the BS, MS, and PhD degrees, all in computer science and information engineering, from National Chiao Tung University, Taiwan, in 1986, 1988, and 1992, respectively. From 1993 to 1995, he served his military obligation at both the Chinese Air Force Academy (CAFA) and the General Headquarters of the ROC Air Force. He is currently an associate professor in the Department of Information Management, Yuan-Ze Institute of Technology, Chung-Li, Taiwan, ROC.

His current research interests include distributed database systems, schema integration in federated database systems, uncertain data manipulation, data mining, and mobile computing. He has had papers published in *The Very Large Database Journal*, *Data and Knowledge Engineering Journal*, *Distributed and Parallel Databases: An International Journal*, and *Journal of Computer and Information Science*. Dr. Tseng is a member of the IEEE Computer Society.