Incomplete Information Costs and Database Design

HAIM MENDELSON
University of Rochester
and
ADITYA N. SAHARIA
University of Washington

This paper presents a methodology for trading-off the cost of incomplete information against the data-related costs in the design of database systems. It investigates how the usage patterns of the database, defined by the characteristics of information requests presented to it, affect its conceptual design. The construction of minimum-cost answers to information requests for a variety of query types and cost structures is also studied. The resulting costs of incomplete database information are balanced against the data-related costs in the derivation of the optimal design.

Categories and Subject Descriptors: H.1.1 [**Models and Principles**]: Systems and Information Theory; H.2.1 [**Database Management**]: Logical Design; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms: Design, Economics, Theory

Additional Key Words and Phrases: Information economics, query processing, retrieval models, value of information

1. INTRODUCTION

A database may be viewed as a computer-based representation of a real-world system, which is intended to support some operational or managerial decision-making activities by responding to users' information requests. Users' requirements serve as input to the conceptual design of the database, which determines the data items available to support users' information needs. Then, a design process leads to the creation of a conceptual schema (satisfying various requirements) and a physical implementation. The result is a database which supports the specified data requirements and, in particular, the queries[1] it was planned to respond to.

[1] We give a formal definition of the term "query" in Section 2.

This work was supported in part by the IBM Program of Support for Education in the Management of Information Systems.
Authors' addresses: H. Mendelson, Graduate School of Management, University of Rochester, Rochester, NY 14627; A. N. Saharia, School of Business Administration, University of Washington, Seattle, WA 98195.

The various design steps have been studied in detail in the literature (cf. [18]). However, the "data requirements" step—which may be the most crucial of all—is still a practiced art, which has received little guidance from the theory. In this step, the costs of including various data items in the database are to be balanced against the benefits, the result being a set of selected database attributes. In fact, practicing database administrators make their living by performing such cost–benefit trade-offs intuitively when they determine the information content of the database.

This paper introduces a framework for examining this problem using a decision-theoretic approach (cf. [12]). Specifically, we study how the usage pattern, defined by the characteristics of the information requests presented to the database, affects its conceptual design. We embed the decision-theoretic approach in the operational database environment, linking users' information demands to the supply of data by the database. An important feature of this environment is the variety of decision (and operational) problems which are to be supported by a single database. In this environment, the designer has to select a set of data items that will jointly support users' diverse demands.

The design is viewed as a mapping from the input data to a database state. This mapping is selected before the input data becomes available, and before users' information requests are materialized. Thus, events take place in a sequence of three stages: (i) design, (ii) data input, and (iii) queries (or information requests). The database designer, operating in stage (i), has to cope with two types of uncertainty, which unfold at different times: input uncertainty, which is resolved at stage (ii), and interaction (or query) uncertainty, resolved at stage (iii). This process results in data uncertainty, which calls for the use of statistical query evaluation strategies.

Incomplete information databases have traditionally been modeled by the use of "null values" (see, e.g., [11] for a review; also see, e.g., [3, 5, 9, 10, 19, 20]). A pioneering article examining the role of uncertainty in the evaluation of database queries is due to Wong [22]. In [22], Wong focuses on the *query evaluation* strategy, when the database is subject to data uncertainty, using prior probabilistic information for a *given* database. Wong's study provides a useful framework which serves as a starting point for this research. Specifically, we use Wong's query classification scheme and generalize the cost structure suggested by him, as well as modify his classical ship example to illustrate the points of interest here. Our objective is, of course, different from Wong's, since we study the conceptual design issue while Wong examines the issue of statistical query evaluation. That is, Wong's work focuses on queries which are to be evaluated as part of stage (iii) of our scenario (where the problem is how to cope with *data* uncertainy), whereas this study examines the decision problem arising in stage (i), where the design mapping is to be selected.

The information retrieval literature has preceded the database literature in studying the role of uncertainty in document retrieval (cf. [13]). Swets [16, 17] introduced a probabilistic approach to the document retrieval problem; his approach has been extended by a number of authors (cf. [4, 8, 23]). The effects of term dependence on retrieval performance have been examined by Salton, et al. [14]. Chow and Yu [2] study optimal retrieval queries whose retrieval rule is equivalent to the Neyman–Pearson decision rule; they focus on the use of

feedback information to get better retrieval performance. Like Wong's work [22], these studies appropriately focus on the retrieval strategy rather than the database design strategy that is at the focus of this work.

In what follows, in Section 2 we introduce the framework for our analysis, and in Section 3 study the construction of minimum-cost answers to users' information requests. In Section 4 we discuss the problem of choosing an optimal design, and in Section 5 illustrate the issues through a comprehensive example. Our concluding remarks are offered in Section 6.

2. THE FRAMEWORK

We view the database as a compact representation of relevant real-world data. The input data consist of *information objects* which may be obtained from two possible sources:

(i) The data my be obtained directly from a real-world system reflecting real-world events. In this case, the information objects record the details of these events, which may be represented by (possibly idealized) transactions (e.g., sales records of a retail firm).

(ii) The data may be extracted from a larger existing database. In this case, the database represents a view (i.e., a logical subset of a larger "universal" database, e.g., salary data extracted from a larger "Employee" database).

We unify the two cases and use the term *input* to indicate the data source.

The purpose of the database is to provide information that reduces the uncertainty about the values of relevant variables. Most existing work in the database area focuses on the idealized cases known as "complete information databases," where the database reduces the uncertainty to complete certainty. This means that the database contains all the relevant data for all possible queries.[2] In most real-life situations, however, uncertainty is pervasive because it is sometimes infeasible and often uneconomical to completely eliminate uncertainty: the expected benefits from reducing the uncertainty to complete certainty are insufficient to offset the costs of doing so. Thus, for example, real numbers are not stored with infinite precision, historical data is not kept forever, and some relevant entities or data-items may not be included in a subschema since their added value does not justify their added cost. Also, the database often represents an aggregation of some "raw" data rather than its full details.

The data stored in the database is used to respond to various *queries*. The information provided by the answers to the queries supports various managerial or operational decisions. When the information stored in the database is incomplete, the response provided to a query may be only an estimate of the correct response. An incorrect response may bring about erroneous actions or decisions, which are costly. We call this cost "the cost of incomplete database information."

The database designer is interested in reducing the cost of incomplete database information: however, this will usually require expanding the database while increasing the data collection, manipulation, storage and retrieval costs (we term these costs "data-related costs"). One of the most important tasks of the database designer is to balance the cost of incomplete database information against the

[2] Thus, "completeness" is a property of the database and the set of all possible queries.

data-related costs. Since this balancing is to be performed at *design* time, the designer faces two types of uncertainty, which unfold at different stages:

(1) *Input uncertainty*, resulting from the fact that the realization of the input data may be unknown at design time. When the input is in the form of transactions from a real-world system, this is because the actual input data is typically absent at design time (which takes place before the system becomes operational). When the data is extracted from a larger "universal" database, the database may be updated after the "view" is defined by the designer. Consequently, the designer will have incomplete information on the state of the "universal" database at usage time.

(2) *Information-request (or interaction) uncertainty*, resulting from the fact that, at design time, the designer is unable to accurately predict what specific information requests will take place in the future; he possesses only probabilistic knowledge of the possible future requests.

To formalize these notions in more concrete terms, we introduce some notation and assumptions. We assume that the input data X may be represented by a $T \times n$ matrix,

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & & & \vdots \\ x_{T1} & x_{T2} & \cdots & x_{Tn} \end{bmatrix}$$

We call the rows of the matrix X "information objects" and denote row t by $x_t = (x_{t1}, x_{t2}, \ldots, x_{tn})$. We call the columns of X "attribute variables" (or, in short, "attributes"). Each attribute j ($j = 1, 2, \ldots, n$) is associated with a *domain* E_j such that its values $x_{tj} \in E_j$ for all $t = 1, 2, \ldots, T$. Information objects are uniquely identified by the row number t, which serves as an implicit attribute. This allows us to treat attributes $1, 2, \ldots, n$ on the same footing, rather than treat a single attribute (or a set of attributes) as a database key.[3] The information objects x_t are sampled from the composite domain $E^n \equiv E_1 \otimes E_2 \otimes \cdots \otimes E_n$, where $\otimes$ denotes Cartesian product. This terminology is clearly consistent with the usual relational database terminology, viewing the input data X as a relation defined over the domains $E_1, E_2, \ldots, E_n$.

A *database state* D (a matrix) is a representation of information derived from the input data X and is stored in the database. In general, a *design* is a procedure which transforms the input data X into a database state D. We consider designs that repeatedly apply a given transformation rule f to each information object x_t. A design is then represented by a mapping f, defined over E^n, which assigns to each information object x_t a (multidimensional) transformed object $d_t = f(x_t)$. Thus the transformation f is applied to each row of the $T \times n$ matrix X, yielding a $T \times m$ matrix D.[4] In particular, any subset of m attributes ($m \leq n$) defines a

[3] Otherwise the keys may be transformed into row numbers. We assume that all feasible designs include the database keys.

[4] Note that this means that the row number t (serving as a key value) is preserved, since row t of X is transformed into row t of D.

projection mapping obtained by projecting each information object x_t over the selected attributes. The resulting database state D consists of the corresponding columns of X. In this case we say that the database represents a *view* of X; the mapping f is then identified by the subset of attributes included in the database. For notational convenience, we let f denote, at the same time, the design (i.e., the mapping from x_t to d_t) and the set of attributes included in the view.

To illustrate these concepts, we present a (modified) version of an example due to Wong [22]. Consider a collection of T ships, each uniquely identified by a number $t \in \{1, 2, \ldots, T\}$. The attribute variables are as follows:

Attribute 1: Type (Ty), with domain E_1 = {carrier, sub};
Attribute 2: Speed (in knots) (S), with domain E_2 = [20, 40];
Attribute 3: Current Location (CL), with domain E_3 = {A, P, I, M}; and
Attribute 4: Last Week's Location (LWL), with the same domain, $E_4 = E_3$ = {A, P, I, M}.

In E_3 and E_4, A stands for the Atlantic Ocean, P for the Pacific Ocean, I for the Indian Ocean, and M for the Mediterranean Sea. An example of an information object representing ship t (the tth row of X) may be x_t = (sub, 25, A, M).

Here, any subset of the attributes {Ty, S, CL, LWL} defines a view. For example, if we select the subset f = {Ty, S}, the database will include for each ship its Type and Speed.

In general, the mapping f is not one-to-one. In the above example, if we choose the view based on f = {Ty, S} and ship 1 has x_1 = (sub, 25, A, M), then $d_1 = f(x_1)$ = (sub, 25). The same value of d_1 would be obtained if x_1 = (sub, 25, I, P), hence d_1 is less informative than x_1. This will result in incomplete information costs when a query such as "find the current location of all ships having speed less than 30 knots" is invoked. Thus the cost of incomplete database information depends on the design selected, f; the database designer has to choose a design f by balancing the anticipated cost of incomplete database information against the data-related costs (including data collection, storage, retrieval, and manipulation costs). Clearly, these costs depend on the design f.

At design time the input data realization X is not known to the designer; he has only some subjective probability distribution over $(E^n)^T$, which guides him through the design process. In what follows, we assume that the information objects $x_1, x_2, \ldots, x_T$ are independent, identically distributed (i.i.d.) random vectors, each following a joint distribution $F_x(\cdot)$ over E^n. It is important to emphasize that this specification *does* allow for probabilistic dependence across attributes, which is an important property of real-world databases. To illustrate, in the above ship example it is natural to allow for a dependence of Current Location (CL) on Last Week's Location (LWL), or for a dependence of Speed on Type.

Information is requested from the database in the form of queries. A *query* Q is an ordered triplet (π, B_π, η) where $\pi \subseteq \{1, 2, \ldots, n\}$ is a set of k attributes. $B_\pi \subseteq E^n$ specifies a retrieval criterion (a set of qualifying values for the k attributes included in π), and $\eta \subseteq \{1, 2, \ldots, n\}$ is a set of attributes whose values are to be displayed. The query $Q = (\pi, B_\pi, \eta)$ is to be interpreted as: "find all the information objects t such that $\pi(x_t) \in B_\pi$, and for each information object

selected, display t and the values $\eta(x_t)$" (where π and η are projection operators). Thus, for the above ship example, the query

Q_1: "find the ships (currently) located in the Atlantic."

will have $\pi_1 = \text{CL}$ (since retrieval is based on Current Location), $B_{\pi_1} = \{\text{A}\}$ (select only if CL is A) and $\eta_1 = \phi$ (since only the ship identification is to be displayed). The query

Q_2: "find all ships located in the Atlantic and having speed greater than 30 knots, and also display their Type."

has $\pi_2 = \{\text{CL, S}\}$, $B_{\pi_2} = \{(\text{A}, s) \mid s > 30\}$ and $\eta_2 = \{\text{Ty}\}$.

Consider a user who submits a given query $Q = (\pi, B_\pi, \eta)$. The database management system has to choose an answer a from the set A_Q of all possible answers to the query Q. The general form of such an answer is $a = \{t, h_t(\eta) \mid t \in a_r\}$, where the answer set a_r is a subset of $\{1, 2, \ldots, T\}$ specifying which information objects are included in the answer, and $h_t(\eta)$ is an estimate of the η attributes for information object t (where $t \in a_r$). When $\eta = \phi$, we say that the query is a *retrieval query*. The answer to a retrieval query is limited to the answer set a_r, identifying the information objects selected. For example, query Q_1 above is a retrieval query. When $\eta \neq \phi$, we say that the query is a *value-extracting query*; Q_2 is an example of a value-extracting query.

In general, some of the attributes included in π and η may not be available in the database (depending on the design f). For example, in our ship example consider the view $f = \{\text{S, LWL}\}$. The query Q_1 above retrieves on the basis of $\pi_1 = \text{CL}$, which is not available in the database. In this case it becomes necessary to make an inference from the values of the available attributes on the value of the missing attribute in order to decide whether to include an information object in the answer set. In the case of query Q_2, above, both CL and Ty are missing in the design f, hence both the answer set a_r and the displayed values of Ty are subject to uncertainty.

The criteria for formulating the answer a to a query Q are based on the costs associated with incorrect or inaccurate answers. These costs may depend not only on the query Q, but also on the use to be made of the information—which typically will vary between invocations of the same query Q. For example, the cost of incorrect answers to query Q_1 above may be drastically different in war time and in peace time. We model such variations using a parameter μ which may vary across invocations of the query Q. We call the pair $(Q, \mu) \equiv \theta$ an *information request*. The information request θ uniquely identifies the query component Q; thus, we can also index query parameters by θ.[5]

Consider an information request θ submitted for processing by the database management system. The system has to decide on an answer a to the information request θ given the database state D. The cost associated with incorrect (or inaccurate) answers is a function $C(a; \theta, X)$ of the information request (θ), the full input matrix (X), and the answer provided (a). However, when the answer is evaluated, only the database state D is known, and it is necessary to make an

[5] For example, A_θ is the set of all possible answers to the query Q, where $\theta = (Q, \mu)$.

inference on the input matrix X from which D has been derived. This calls for solving the following decision problem:

Find $a^* = a^*(\theta, D)$ such that

$$c(a^*; \theta, D) = \min_{a \in A_\theta} c(a; \theta, D), \tag{2.1}$$

where $c(a; \theta, D)$ is the expected cost associated with the answer a in database state D: $c(a; \theta, D) \equiv E_X[C(a; \theta, X) \mid D]$, where the conditional expectation is taken over all possible realizations of the input X (i.e., with respect to the *input* uncertainty). The minimum expected cost associated with information request θ, given that the database state is D, is given by $\bar{c}(\theta, D) = c(a^*(\theta, D); \theta, D)$, where $a^*(\theta, D)$ is the optimal answer.

Let $N(\theta)$ be the number of times an information request θ will be invoked per unit of time;[6] $N(\theta)$ is a random variable with expected value $\lambda(\theta) = E[N(\theta)] < \infty$. We call $\lambda(\theta)$ "the frequency of information request θ." Summing over all information requests, the expected cost of incomplete database information per unit of time depends on the design f, and is given by

$$I(f) = \sum_{\theta} \lambda(\theta) \underset{D}{E}\bar{c}(\theta, D). \tag{2.2}$$

With each design f, we can associate a physical implementation and a data-related cost (which includes the cost of observing the relevant data, collecting it, storing it in the database, and retrieving it from the database). These costs will depend on the design f, on the input data X, and on the information requests. At design time, however, the last two components are unknown, and we can evaluate only the *expected* data-related cost associated with design f. We denote the expected data-related cost under design f (averaged per unit of time over the relevant interval—say the lifetime of the current design) by $B(f)$. Thus the database design problem can be formulated as follows: from a given family, $\mathcal{F}$, of feasible designs, find the design f that minimizes the expected overall cost function, $I(f) + B(f)$. That is, the task of the database designer is to choose a design f^* such that

$$I(f^*) + B(f^*) = \min_{f \in \mathcal{F}} \{I(f) + B(f)\}, \tag{2.3}$$

that is, a design that minimizes the overall expected cost.

We consider the *view design* problem, where the set $\mathcal{F}$ of feasible designs consists of all the possible views (recall that a view projects each information object over a subset of the n input attributes). The view design problem is the problem of optimally selecting a subset of the input attributes to support a given set of information requests. In this case the projection mapping operates on a row-by-row basis, where each information object x_t is transformed into a projected database tuple d_t.[7] A precursor to the optimal design problem is the construction of optimal answers, which is studied in the next section.

[6] Over a relevant time interval (say, during the life of the current design).

[7] Thus preserving the row number t, which serves as a database key.

3. MINIMUM-COST ANSWERS

In this section we examine the construction of a minimum-cost answer to an information request of the form $\theta = (Q, \mu)$, where the query Q is

$$Q(\pi, B_\pi, \eta) : \text{find } \{t, \eta(x_t) \mid \pi(x_t) \in B_\pi\}. \tag{3.1}$$

As mentioned earlier, the possible answers to (3.1) are of the form $a = \{t, h_t(\eta) \mid t \in a_r\}$, where $a_r \subseteq \{1, 2, \ldots, T\}$. With each feasible answer a, we associate a cost $C(a; \theta, X)$. The cost structure is as follows:

(i) Costs are additive across information objects.

(ii) For each information object, the associated cost has two components: a *retrieval* cost component and a *value extraction* cost component. When the query is a retrieval query ($\eta = \phi$), the latter cost is zero.

(iii) A cost multiplier $s = s(\theta)$ multiplies both cost components and measures the relative importance of accurate responses to the information request.

(iv) For the retrieval cost component, we say that a *type-I* error (or omission) occurred when $\pi(x_t) \in B_\pi$, but $t \notin a_r$. We say that a *type-II* error (or false-alarm) occurred when $t \in a_r$, but $\pi(x_t) \notin B_\pi$. In the information retrieval literature, a type-I error corresponds to the case where a relevant document is not retrieved, and a type-II error to the case where a nonrelevant document is retrieved (cf. [13, ch. 6]). With each type-II error (false alarm), we associate a relative cost of α; with each type-I error (omission), we associate a relative cost of $(1 - \alpha)$. (This is consistent with the usual methodology (cf. [22]).) We allow α to depend on the parameter μ of the information request.

(v) The *value-extraction* cost component is incurred only for tuples included in the answer set a_r, and reflects the cost associated with errors in the values of the displayed attributes. It is specified by a function $c_v(h_t, x_t, \eta)$, which translates the difference between the displayed values h_t and the correct projected values $\eta(x_t)$ into cost terms (we allow the value-extraction cost function c_v to depend on μ.) When $h_t = \eta(x_t)$, no cost is incurred, hence $c_v(\eta(x_t), x_t, \eta) = 0$ (some specific structures for c_v are studied in Section 4). Note that the parameter μ associated with the information request $\theta = (Q, \mu)$ may affect the cost parameters α and s as well as the value-extraction cost function c_v.

The answer to an information request θ is the result of two decisions: a *retrieval* decision and an *estimation* decision. The retrieval decision specifies the subset of tuples to be selected, a_r (a subset of $\{1, 2, \ldots, T\}$). The estimation decision specifies the estimated values for the η attributes that will be displayed in the response. The resulting estimate may be written as $\{h_t \mid t \in a_r\}$. For purposes of discussion, it is convenient to define the result of the estimation decision by the *full* matrix $H \equiv (h_t, t = 1, 2, \ldots, T)$, regardless of the tuples actually selected. Then, only h_t values for $t \in a_r$ will be displayed in the output. Following this convention, an answer a is fully specified by a_r and H, and we can write $a = (a_r, H)$.

Our assumptions clearly imply that an optimal answer may be evaluated on a tuple-by-tuple basis. In particular, for each tuple t, regardless of whether or not it ends up being included in the answer set, we can evaluate an optimal estimate of the η attributes, which we denote by $h^*(\eta, d_t) = \{h_i^*(d_t), i \in \eta\}$. Further, we

can evaluate for each tuple t ($t = 1, 2, \ldots, T$) the minimal potential expected value-extraction cost (to be incurred only if it is selected for display):

$$\bar{c}_v(\eta, d_t) = E[c_v(h^*(\eta, d_t), x_t, \eta) \mid d_t] = \min_{h_t} E[c_v(h_t, x_t, \eta) \mid d_t]. \tag{3.2}$$

Consider now the decision whether or not to include tuple t in the answer. If the tuple is excluded, the omission cost would be incurred if $\pi(x_t) \in B_\pi$; thus the expected cost of excluding tuple t from the answer is given by

$$s \cdot (1 - \alpha) \cdot \Pr\{\pi(x_t) \in B_\pi \mid d_t\} = s(1 - \alpha)p(\pi, B_\pi, d_t), \tag{3.3}$$

where we define

$$p(\pi, B_\pi, d_t) = \Pr\{\pi(x_t) \in B_\pi \mid d_t\}. \tag{3.4}$$

If tuple t is included in the answer, the contribution of the retrieval cost component to the expected cost is given by $s\alpha(1 - p(\pi, B_\pi, d_t))$ and the contribution of the value-extraction cost component is $s \cdot \bar{c}_v(\eta, d_t)$. It follows that tuple t belongs to an optimal answer if and only if

$$p(\pi, B_\pi, d_t) \geq \alpha + \bar{c}_v(\eta, d_t), \tag{3.5}$$

and $a^*(\theta, D) = (a_r^*(\theta, D), H^*(\theta, D))$ is optimal where

$$a_r^*(\theta, D) = \{t \mid p(\pi, B_\pi, d_t) \geq \alpha + \bar{c}_v(\eta, d_t)\}$$

and

$$H^*(\theta, D) = \{h^*(\eta, d_t), \quad t = 1, 2, \ldots, T\}.$$

The procedure for responding to an information request can thus be described as follows. For each tuple t, the available data d_t is inspected and the corresponding (minimal) value-extraction cost $\bar{c}_v(\eta, d_t)$ is evaluated (the optimal estimate $h^*(\eta, d_t)$ may be evaluated either at the same time or following the decision to include tuple t in the answer). The probability $p(\pi, B_\pi, d_t)$ is then evaluated and condition (3.5) is tested. If (3.5) does not hold, the tuple is rejected. If (3.5) holds, tuple t is selected and the optimal estimate $h^*(\eta, d_t)$ is displayed.

The minimal expected cost of incomplete database information associated with information request θ in database state D is thus given by

$$\begin{aligned} \bar{c}(\theta, D) &= s \cdot \sum_{t=1}^{T} [\min\{p\{\pi, B_\pi, d_t), \alpha + \bar{c}_v(\eta, d_t)\} - \alpha p(\pi, B_\pi, d_t)] \\ &= s \cdot \sum_{t=1}^{T} g_\alpha(p(\pi, B_\pi, d_t), \bar{c}_v(\eta, d_t)), \end{aligned} \tag{3.6}$$

where $g_\alpha(p, \delta)$ is defined by

$$g_\alpha(p, \delta) = \min\{p, \alpha + \delta\} - \alpha p. \tag{3.7}$$

$g_\alpha(p, \delta)$ represents the expected (relative) cost of incomplete information for a tuple with probability of qualifying p and expected value-extraction cost δ.

Although condition (3.5) is a retrieval condition, the retrieval decision also depends on the expected value-extraction cost, $\bar{c}_v(\eta, d_t)$: a tuple will be included

in the answer if and only if the expected *combined* cost of including it is less than the expected cost of excluding it. Thus, while, from an operational viewpoint, we can separate the processing of the query to a retrieval part and a value-extraction part, the retrieval decision depends on the expected value-extraction cost. In this sense, the problem cannot be separated into two independent problems of retrieval and estimation. As a result, Wong's [22, Sect. 6] separation approach to the solution of value-extracting queries is *not* optimal in our context, since it does not take this combined cost into account.

In the case of retrieval queries (where $\eta = \phi$), the value-extraction cost component is zero and our optimality condition (3.5) reduces to

$$a_r^*(\theta, D) = \{t \mid p(\pi, B_\pi, d_t) \geq \alpha\}, \tag{3.8}$$

that is, the answer set consists of those tuples whose probability of qualifying is greater than or equal to a threshold value given by the relative cost of false alarm α. This condition is consistent with the retrieval strategy suggested by Wong [22, Sects. 4–5], as well as the optimal retrieval strategies suggested in the information retrieval literature (cf. [2, 7]).

4. OPTIMAL DESIGN

In this section we apply the results of Section 3 to derive the costs of incomplete database information and to solve the optimal design problem, (2.3). Optimal designs are then studied for increasingly complex situations. We start with single-attribute retrieval queries when attributes are independent, proceed to the case of dependent attributes, and then study the more general case of combined retrieval and value-extracting queries.

To derive the cost of incomplete database information under a given design f, we have to aggregate expression (3.6) over all information requests. Recalling that $\lambda(\theta)$ is the frequency of information request θ, the expected cost of incomplete database information under design f is given by

$$I(f) = \sum_{\theta} \lambda(\theta) s(\theta) \sum_{t=1}^{T} \underset{d_t}{E} g_\alpha(p(\pi, B_\pi, d_t), \bar{c}_v(\eta, d_t)). \tag{4.1}$$

Since $x_1, x_2, \ldots, x_T$ are i.i.d., so are $d_1, d_2, \ldots, d_T$, and Eq. (4.1) simplifies to[8]

$$I(f) = T \cdot \sum_{\theta} \Lambda(\theta) \underset{d}{E} g_\alpha(p(\pi, B_\pi, d), \bar{c}_v(\eta, d)), \tag{4.2}$$

where the i.i.d. variates $d_1, d_2, d_3, \ldots$ are distributed as d, and we define $\Lambda(\theta) = \lambda(\theta) \cdot s(\theta)$, the frequency-weighted importance of information request θ (which is independent of the design f).

Another element of (4.2) which is independent of the design f, and hence can be evaluated a priori is the functional relationship $g_\alpha(\cdot, \cdot)$, given by (3.7).[9] Further, if we assume that the values of α are a random sample from some

[8] Note that, due to the fact that $d_1, d_2, \ldots, d_t, \ldots$, are i.i.d., we can write

$$\underset{d_t}{E}\ g_\alpha(p(\pi, B_\pi, d_t), \bar{c}_v(\eta, d_t)) \quad \text{as} \quad \underset{d}{E}\ g_\alpha(p(\pi, B_\pi, d), \bar{c}_v(\eta, d)).$$

[9] The arguments of $g_\alpha(\cdot, \cdot)$, p and δ are of course dependent on the design f.

distribution $\Gamma_\alpha(\cdot)$, we can replace $g_\alpha(p, \delta)$ by

$$g(p, \delta) = \underset{\alpha}{E}\, g_\alpha(p, \delta) = \int g_\alpha(p, \delta)\, d\Gamma_\alpha(\alpha). \tag{4.3}$$

In particular, when α is uniformly distributed over the unit interval, we have

$$g(p, \delta) = \begin{cases} \frac{1}{2}p & \text{for } p \le \delta \\ \frac{1}{2}p - \frac{1}{2}(p - \delta)^2 & \text{for } p \ge \delta. \end{cases} \tag{4.4}$$

In general, $g_\alpha(p, \delta)$ is a concave function of p, satisfying $g_\alpha(0, \delta) = 0$. Thus $g(p, \delta)$ is also concave in p and satisfies $g(p, \delta) = 0$. Since $g_\alpha(p, \delta)$ defined by (3.7) may be viewed as a special case with degenerate $\Gamma_\alpha(\cdot)$, we can pursue our analysis using the function $g(p, \delta)$.

We consider first the design of a database that supports only retrieval queries (i.e., queries with $\eta = \phi$). In this case, the optimal retrieval policy given by (3.5) reduces to "a tuple t belongs to an optimal answer if and only if $p(\pi, B_\pi, d_t) \ge \alpha$." In the case of retrieval queries, the value-extraction cost is $\delta = 0$, hence $g(p, \delta) = g(p, 0)$ is a function of p only, which we denote by $G(p)$. Clearly, $G(\cdot)$ is concave and $G(0) = G(1) = 0$. In the case where α is uniformly distributed over the unit interval, (4.4) becomes

$$G(p) \equiv g(p, 0) = \tfrac{1}{2}p(1 - p). \tag{4.5}$$

We analyze first the design of a database with independent attributes which supports single-attribute retrieval queries.

4.1 Single-Attribute Retrieval Queries: Independent Database Attributes

Let each domain E_i be discrete, say $E_i = \{1, 2, \ldots, N_i\}$, with input distribution specified by

$$\Pr\{x_{ti} = j\} = p_{ij}, \quad j = 1, 2, \ldots, N_i,$$

where, for all t, $x_{t1}, x_{t2}, \ldots, x_{tn}$ are independent. Assume that the data-related costs are linear in the number of attributes in the database (i.e., the cost of adding an attribute to the database is a constant b). Consider single-attribute-project retrieval queries of the form

$$Q(i, j): \quad \text{find} \quad \{t \mid x_{ti} = j\}.$$

Here, $\pi = i$ and $B_\pi = j$. We denote $Q(\pi, B_\pi)$ more briefly by (i, j); also, $p(\pi, B_\pi, d_t)$ can be written as $p(i, j, d_t)$. Assuming that $\Lambda(\theta)$ depend only on the query $Q(i, j)$, we write $\Lambda(\theta) = \Lambda(Q(i, j)) = \Lambda(i, j)$.

Consider a given design f. To evaluate the expected cost of incomplete database information from (4.2), it is sufficient to compute $E_{d_t} G(p(i, j, d_t))$ for all i, j. But for $i \in f$, $p(i, j, d_t)$ is 1 if $d_{ti} = j$ and zero if $d_{ti} \neq j$; in either case, $G(p(i, j, d_t)) = 0$. Thus, when $i \in f$, the answer a_r will consist of the tuples t with $d_{ti} = j$, with a zero cost of incomplete database information. When $i \notin f$,

$$p(i, j, d_t) = \Pr\{x_{ti} = j \mid d_t\} = \Pr\{x_{ti} = j\} = p_{ij},$$

hence $E_{d_t} G(p(i, j, d_t)) = G(p_{ij})$. Note that since x_t are i.i.d. and the attributes are mutually independent, a_r will either be null or will include all the tuples,

depending on whether $p_{ij} \geq \alpha$ or $p_{ij} < \alpha$. The expected cost of incomplete database information is therefore given by

$$I(f) = T \cdot \sum_{i \notin f} \sum_{j} \Lambda(i, j) G(p_{ij}).$$

It follows that adding attribute i to a design f where $i \notin f$ will reduce the expected cost of incomplete database information by

$$v_i \equiv T \cdot \sum_{j} \Lambda(i, j) G(p_{ij}), \tag{4.6}$$

independent of f. We can thus call v_i "the value of attribute-i information," since this value is independent of the other attributes and depends only on i. If we number the attributes by decreasing v_i, that is, $v_1 \geq v_2 \geq v_3 \geq \cdots \geq v_n$, the optimal design will include attributes 1, 2, ..., k, where k is the first index satisfying $v_{k+1} < b$ (where we define $v_{n+1} \equiv 0$): this design includes in the database all attributes i for which the marginal benefit of inclusion v_i exceeds the marginal cost b.

4.2 Single-Attribute Retrieval Queries: Two Dependent Attributes

To gain insight into the effects of dependence among attributes, we modify the above setting by assuming that attributes 1 and 2 are (probabilistically) dependent, while all other attributes are independent. We assume that the joint distribution of attributes 1 and 2 is given by

$$\Pr\{x_{t1} = j, x_{t2} = m\} = z_{jm} \tag{4.7}$$

and define $z_{j\cdot} = \sum_m z_{jm}$, $z_{\cdot m} = \sum_j z_{jm}$. As in Section 4.1, the reduction in the cost of incomplete database information as a consequence of adding attribute i ($i \neq 1, 2$) to the database is independent of the other attributes and is given by (4.6). In contrast, the redution in the cost of incomplete information when attribute 1 (2, respectively) is added to a design f depends on whether attribute 2 (1, respectively) is already included in f. When $1, 2 \notin f$,

$$v_1 = I(f) - I(f \cup \{1\}) = I(\phi) - I(\{1\}) \tag{4.8}$$

represents the reduction in the cost of incomplete database information when attribute 1 is added to a design f that does *not* include attribute 2. Similarly, if $1, 2 \notin f$,

$$v_2 = I(f) - I(f \cup \{2\}) = I(\phi) - I(\{2\}) \tag{4.9}$$

represents the value of adding attribute 2 to a design f that does *not* include attribute 1. We also define, for $2 \in f$, $1 \notin f$,

$$v_{12} = I(f) - I(f \cup \{1\}) = I(\{2\}) - I(\{1, 2\}). \tag{4.10}$$

v_{12} represents the value of adding attribute 1 to the design when attribute 2 is present; similarly, v_{21} is defined as the value of adding attribute 2 to the database when attribute 1 is already included.

We evaluate $I(f)$ for $f = \phi$, $\{1\}$, $\{2\}$, and $\{1, 2\}$.[10] For the null design $f = \phi$, $p(1, j, d_t) = z_{j.}$ and $p(2, m, d_t) = z_{.m}$, hence

$$I(\phi) = T \cdot \sum_j \Lambda(1, j)G(z_{j.}) + T \cdot \sum_m \Lambda(2, m)G(z_{.m}) + \sum_{i=3}^{n} v_i.$$

Also, since only attributes 1 and 2 are mutually dependent,

$$I(\{1, 2\}) = \sum_{i=3}^{n} v_i.$$

We next evaluate $I(\{1\})$. For $f = \{1\}$, $p(1, j, d_t)$ is always zero or one, hence $G(p(1, j, d_t)) = 0$. As for queries involving attribute 2, such as $Q(2, m)$, if $d_{t1} = j$.

$$p(2, m, d_t) = \Pr\{x_{t2} = m \mid x_{t1} = j\} = z_{jm}/z_{j.},$$

since $\Pr\{x_{t1} = j\} = z_{j.}$. We thus obtain,

$$\underset{d_t}{E}\, G(p(2, m, d_t)) = \sum_j z_{j.}G(z_{jm}/z_{j.}),$$

and hence

$$I(\{1\}) = T \cdot \sum_{j,m} \Lambda(2, m)z_{j.}G(z_{jm}/z_{j.}) + \sum_{i=3}^{n} v_i.$$

Similar considerations lead to

$$I(\{2\}) = T \cdot \sum_{j,m} \Lambda(1, j)z_{m.}G(z_{jm}/z_{.m}) + \sum_{i=3}^{n} v_i.$$

Substituting the above expressions in (4.8)–(4.10) yields

$$v_2 = T \cdot \sum_j \Lambda(1,j)[G(z_{j.}) - \sum_m z_{m.}G(z_{jm}/z_{.m})] + T \cdot \sum_m \Lambda(2, m)G(z_{.m}), \quad (4.11a)$$

$$v_1 = T \cdot \sum_j \Lambda(1,j)G(z_{j.}) + T \cdot \sum_m \Lambda(2, m)[G(z_{.m}) - \sum_j z_{j.}G(z_{jm}/z_{j.})], \quad (4.11b)$$

and

$$v_{21} = T \cdot \sum_{j,m} \Lambda(2, m)z_{j.}G(z_{jm}/z_{j.}), \quad (4.12a)$$

$$v_{12} = T \cdot \sum_{j,m} \Lambda(1, j)z_{.m}G(z_{jm}/z_{.m}). \quad (4.12b)$$

We now show that $v_{12} \leq v_1$ and $v_{21} \leq v_2$, namely: the existence of a dependent attribute in the database always *reduces* the value of information associated with an attribute, as might be expected. To see this, compare (4.11) with (4.12); clearly,

[10] For other values of f, the value of each attribute $i \neq 1, 2$, v_i may simply be added, due to independence.

it is sufficient to prove that

$$\sum_m z_{\cdot m} G(z_{jm}/z_{\cdot m}) \leq G(z_{j\cdot}), \tag{4.13a}$$

and

$$\sum_j z_{j\cdot} G(z_{jm}/z_{j\cdot}) \leq G(z_{\cdot m}). \tag{4.13b}$$

Now the left-hand side of (4.13a) may be viewed as the expected value of the concave function $G(\cdot)$, applied to a random variable taking on the values $z_{jm}/z_{\cdot m}$ with probabilities $z_{\cdot m}$; the expected value of this random variable is $z_{j\cdot}$. But then (4.13a) clearly follows from Jensen's inequality (cf. [6, p. 249]), which states that the expected value of a concave function of a random variable is bounded by the value of that function at the expectation of the random variable. Inequality (4.13b) follows in a similar fashion.

As a simple example, consider the case where both domains consist of two values, say $E_1 = E_2 = \{1, 2\}$, and

$$Z = \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix} = \begin{bmatrix} \rho & 1/2 - \rho \\ 1/2 - \rho & \rho \end{bmatrix}$$

where $0 \leq \rho \leq 1/2$. Note that x_{t1} and x_{t2} are perfectly negatively correlated when $\rho = 0$, perfectly positively correlated when $\rho = 1/2$, and independent when $\rho = 1/4$. It is straightforward to derive

$$\upsilon_1 = T \cdot \Lambda_{1\cdot} \cdot G(1/2) + T \cdot \Lambda_{2\cdot} \cdot [G(1/2) - 1/2G(2\rho) - 1/2G(1-2\rho)],$$
$$\upsilon_2 = T \cdot \Lambda_{2\cdot} \cdot G(1/2) + T \cdot \Lambda_{1\cdot} \cdot [G(1/2) - 1/2G(2\rho) - 1/2G(1-2\rho)],$$
$$\upsilon_{12} = T \cdot \Lambda_{1\cdot} \cdot [1/2G(2\rho) + 1/2G(1-2\rho)],$$

and

$$\upsilon_{21} = T \cdot \Lambda_{2\cdot} \cdot [1/2G(2\rho) + 1/2G(1-2\rho)],$$

where $\Lambda_{1\cdot} = \Lambda(1, 1) + \Lambda(1, 2)$ and $\Lambda_{2\cdot} = \Lambda(2, 1) + \Lambda(2, 2)$. Assuming that α is uniformly distributed over $[0, 1]$, $G(p) = 1/2p(1-p)$ (see (4.5)). We demonstrate the behavior of υ_2 and υ_{12} as functions of ρ in Figure 1 (the figure depicts the case where $\Lambda_{2\cdot} > \Lambda_{1\cdot}$). Note that

$$\upsilon_2 + \upsilon_{12} = T \cdot G(1/2) \cdot (\Lambda_{1\cdot} + \Lambda_{2\cdot})$$

is a constant, independent of ρ. When $\rho = 0$ or $\rho = 1/2$, the value of attribute sets $\{1\}$, $\{2\}$, and $\{1, 2\}$ is the same, since any one of them determines the other. Clearly, the assumption $\Lambda_{2\cdot} > \Lambda_{1\cdot}$ implies that, for all ρ, $\upsilon_2 > \upsilon_1$. The (gross) value of attribute 2 (υ_2) is always greater than υ_{12}. Assuming that the cost of adding any attribute to the database is b, we add attribute 2 to a design f (satisfying $1, 2 \notin f$) if and only if $\upsilon_2(\rho) > b$, and then add attribute 1 to the design $f \cup \{2\}$ if and only if $\upsilon_{12}(\rho) > b$. Note that υ_2 depends on ρ even when the initial design f does *not* contain attribute 1; the reason is that in adding attribute 2 to the database, not only do we reduce the cost associated with queries on this attribute, but we also improve the quality of inference on attribute 1. Thus the database designer must also take into account the *indirect* benefits of adding an attribute to the database, associated with improved quality of inference on related (unavailable) data items.

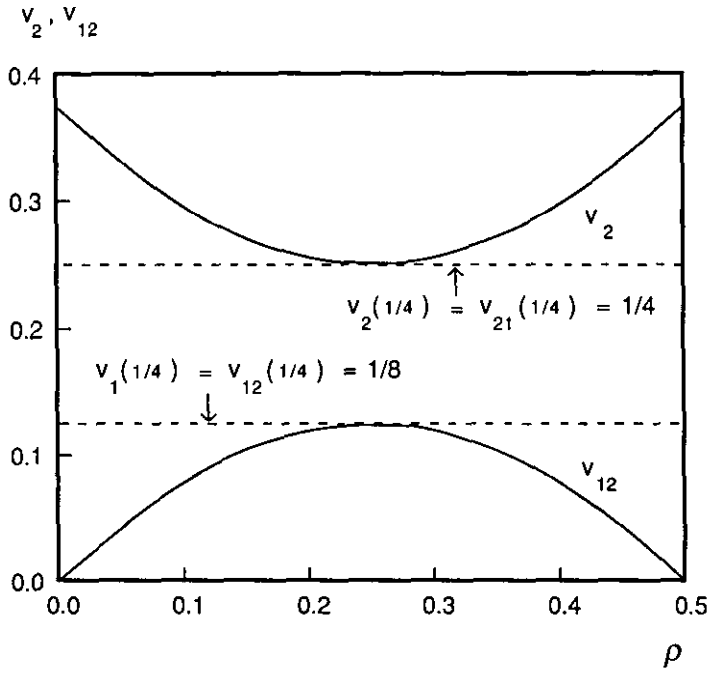

Fig. 1. v_2 and v_{12} as functions of ρ (Section 4.2). ($\Lambda_2 = 2$, $\Lambda_1 = 1$).

4.3 Value-Extracting Queries: General Considerations

Next we consider the optimal design of databases that support more general queries, involving both retrieval and value-extraction. Here, the optimal answer, a^*, and therefore also the optimal design, f^*, depend on the retrieval cost as well as the value-extraction cost. Hence, we start our analysis of value-extracting queries by examining the form of the value-extraction cost function, $c_v(h_t, x_t, \eta)$.

Several functional forms are useful representations of $c_v(h_t, x_t, \eta)$; some of the possibilities are as follows:

(i) For each tuple t included in the answer, a cost β_i is incurred if the value of the ith attribute is incorrect (i.e., whenever $h_{ti} \neq x_{ti}$), and these costs are additive. Thus,

$$c_v(h_t, x_t, \eta) = \sum_{i \in \eta} \beta_i \cdot \Phi_{\{h_{ti} \neq x_{ti}\}},$$

where Φ is an indicator function (equal to 1 if $h_{ti} \neq x_{ti}$, and zero otherwise). This cost structure is meaningful when all the attributes included in η have discrete domains.

(ii) For each tuple included in the answer a, a cost β is incurred if *any* of the attributes is incorrect (i.e., whenever the *whole* vector h_t is not identical to the true value $\eta(x_t)$). Here, $c_v(h_t, x_t, \eta) = \beta \cdot \Phi_{\{h_t \neq \eta(x_t)\}}$. Again, this cost function is meaningful only when all the attributes included in η have discrete domains.

(iii) For each tuple t included in the answer, the cost is quadratic in the magnitude of the error, namely, a cost of $\beta_i(h_{ti} - x_{ti})^2$ is incurred for the ith

attribute, and these costs are additive. Hence,

$$c_v(h_t, x_t, \eta) = \sum_{i \in \eta} \beta_i \cdot (h_{ti} - x_{ti})^2.$$

This cost structure is meaningful when the attributes included in η have continuous domains.

In Section 4.4 we study the problem of choosing an optimal design when the attributes in η have discrete domains and the cost function c_v is given by (i)—the treatment of cost functions (ii) is similar to (i). In Section 4.5 we outline the procedure when the attributes whose values are extracted have continuous domains, and the cost function c_v is given by (iii).

4.4 Value-Extracting Queries: Discrete Attributes

In this section we consider the cost structure (i) and its implications on database design. We fix a design f and consider a query $Q = (\pi, B_\pi, \eta)$. Under the assumed cost structure, the minimum-cost estimate $h_i^*(d_t)$ for the value x_{ti} of attribute $i \in \eta$ in tuple t is the *mode*[11] of the conditional distribution of x_{ti} given d_t, that is, letting $J (J \in E_i)$ be the solution of

$$\Pr\{x_{ti} = J \mid d_t\} = \max_{j \in E_i} \Pr\{x_{ti} = j \mid d_t\},$$

we have $h_i^*(d_t) = J$. The minimal expected value-extraction cost for tuple t is given by

$$\bar{c}_v(\eta, d_t) = \sum_{i \in \eta} \beta_i m_i(d_t),$$

where we have defined

$$m_i(d_t) = 1 - \Pr\{x_{ti} = h_i^*(d_t) \mid d_t\}.$$

Thus, the retrieval condition (3.5) becomes

$$p(\pi, B_\pi, d_t) \geq \alpha + \sum_{i \in \eta} \beta_i m_i(d_t); \tag{4.14}$$

for each tuple t satisfying (4.14), the displayed values will be the modes $\{h_i^*(d_t), i \in \eta\}$. Note that, in particular, tuples with $p(\pi, B_\pi, d_t) < \sum_{i \in \eta} \beta_i m_i(d_t)$, where the expected cost due to inaccurate value-extraction exceeds the expected cost of omission, will be excluded.

Using (4.2), the expected total cost of incomplete database information for a given design f is given by

$$I(f) = T \cdot \sum_{\theta} \Lambda(\theta) \cdot \underset{d}{E}\, g\left(p(\pi, B_\pi, d), \sum_{i \in \eta} \beta_i m_i(d)\right). \tag{4.15}$$

The optimal design is now obtained by balancing the reductions in this cost function against the data-related costs.

[11] This is because at the mode the probability of an error is minimized (and hence the expected cost is minimized).

4.5 Value-Extracting Queries: Continuous Attributes

In this section we study the case where all the attributes included in η have continuous domains, and the cost structure is given by (iii) of Section 4.3. It is well known (cf. [21, p. 619]) that the parameter estimator which minimizes a quadratic loss function is the mean (conditional on all available information). Thus, for a design f and given database state D,

$$h_i^*(d_t) = E[x_{ti} \mid d_t],$$

that is, x_{ti} is estimated by its conditional mean given d_t. The resulting value-extraction cost for tuple t is given by

$$\bar{c}_v(\eta, d_t) = \sum_{i \in \eta} \beta_i E[(x_{ti} - h_i^*(d_t))^2 \mid d_t] = \sum_{i \in \eta} \beta_i \sigma_i^2(d_t),$$

where $\sigma_i^2(d_t)$ denotes the conditional variance of x_{ti} given d_t. Now condition (3.5) becomes

$$p(\pi, B_\pi, d_t) \geq \alpha + \sum_{i \in \eta} \beta_i \sigma_i^2(d_t) \tag{4.16}$$

and, for each tuple t satisfying (4.16), the displayed values will be the conditional means $\{(h_i^*(d_t), i \in \eta\}$. Finally, the expected cost of incomplete database information for a design f is given by

$$I(f) = T \cdot \sum_{\theta} \Lambda(\theta) \underset{d}{E}\, g\left(p(\pi, B_\pi, d), \sum_{i \in \eta} \beta_i \sigma_i^2(d)\right). \tag{4.17}$$

5. A COMPREHENSIVE EXAMPLE

As an example which demonstrates the application of the foregoing ideas, we consider a generalized (and modified) version of Wong's [22] classical ship example introduced in Section 2. Recall that in this example an information object is a ship with attributes Type (Ty), Speed (S), Current Location (CL), and Last Week's Location (LWL).

Here, a major data-related cost is the data collection (i.e., tracking) cost. We assume that the data-related costs depend on the subset of attributes included in the design as follows: The cost of including LWL is 1 monetary unit per ship, the cost of including Ty is 3 monetary units per ship, and the cost of including CL *and* Speed, or CL only, or Speed only, is 8 monetary units per ship (since the *gross* value of information is nonnegative, this implies that either *both* S and CL will be included in the database, or both will be excluded).

We assume that the prior knowledge is summarized by the probability distributions in Tables I–II, which are taken from Wong's example. The prior probability of types is given by Pr{carrier} = Pr{sub} = 1/2; we also assume that the prior distribution of LWL is the (unique) stationary distribution corresponding to the transition probability matrix of Table I. This distribution (which, due to stationarity, is also the distribution of CL) is given in Table III. Finally, using the conditional distribution Pr{Speed $<$ s | Type} and the type probabilities, we derive the unconditional distribution of S (Table IV) and the conditional distribution of Ty given S (Table V).

Table I. Conditional Probability Distribution of Current Location Given Last Week's Location: Pr{CL | LWL}

		CL			
		M	I	P	A
LWL	M	0.8	0.1	0	0.1
	I	0.2	0.8	0	0
	P	0	0.15	0.8	0.05
	A	0.1	0	0.1	0.8

Table II. Conditional Distribution of Speed Given Type, Pr{S < s | Ty} (See Wong [22], Figure 2)

s (Knots)	Carrier	Sub
20	0	0
25	0.4	0
30	1	0.2
35	1	0.8
40	1	1

Table III. The Prior Distribution of LWL (and CL)

CL (or LWL)	Probability
M	7/18
I	5/18
P	1/9
A	2/9

Table IV. The Distribution of Speed, Pr{S < s}

s (Knots)	Pr{Speed < s}
20	0
25	0.2
30	0.6
35	0.9
40	1

Table V. The Conditional Distribution of Type Given Speed, Pr{Ty | S}

		Speed		
		20 ≤ S < 25	25 ≤ S < 30	30 ≤ S < 40
Type	Carrier	1	3/4	0
	Sub	0	1/4	1

Table VI. Expected Cost of Incomplete Database Information per Information Request—Tuple for Query Q_1 (See Appendix A)

Design f	Expected cost of incomplete database information (per information request—tuple)	
	(i) $\alpha = 1/2$	(ii) $\alpha \sim U(0, 1)$
$f = \phi$	0.07778	0.06568
$f = \{LWL\}$	0.07778	0.0568
$f = \{Ty\}$	0.07778	0.05358
$f = \{Ty, LWL\}$	0.05056	0.03582
$f \supseteq \{S, CL\}$	0	0

Table VII. Expected Cost of Incomplete Database Information per Information Request—Tuple for Query Q_2 (See Appendix A)

Design f	Expected cost of incomplete database information (per information request—tuple)	
	(i) $\alpha = 1/2$	(ii) $\alpha \sim U(0, 1)$
$f = \phi$	0.1944	0.1944
$f = \{LWL\}$	0.1944	0.1769
$f = \{Ty\}$	0.1944	0.1188
$f = \{Ty, LWL\}$	0.07778	0.06333
$f = \{S, CL\}$	0.038889	0.03403
$f = \{S, CL, Ty\}$	0	0

We assume that the database is designed to support two queries:

Q_1: Find all the ships (currently) in the Mediterranean with Speed greater than or equal to 30 knots; and

Q_2: Find all the ships (currently) in the Mediterranean and display their types.

Q_1 is a retrieval query characterized by $\pi_1 = \{CL, S\}$ and $B_{\pi} = \{M\} \times (30, 40)$; Q_2 is a value-extracting query with $\pi_2 = CL$, $B_{\pi_2} = \{M\}$, and $\eta_2 = Ty$.

We consider two alternative constructions of information requests based on these queries, which differ in the distribution $\Gamma_\alpha(\cdot)$ of the relative retrieval-cost parameter α (for both queries):

(i) $\alpha_1 = \alpha_2 = 1/2$ (i.e., the cost of omission is equal to the cost of false alarm).

(ii) The parameter α is uniformly distributed over the unit interval. We assume that a cost of $\beta_{Ty} = 1$ monetary unit is incurred whenever the Type displayed for a ship in response to Q_2 is incorrect, and denote the frequency-weighted importance of information requests involving query i by Λ_i $(i = 1, 2)$.

Next we examine the costs of incomplete database information. Clearly, the specification of the problem allows us to compare costs per tuple (i.e., per ship). The designs to be considered are $f = \phi$ (the null design), $f = \{Ty\}$, $f = \{LWL\}$, $f = \{CL, S\}$, $f = \{Ty, LWL\}$, and $f = \{Ty, CL, S\}$ (the remaining designs are obviously suboptimal). Tables VI and VII (respectively) summarize the costs of

Table VIII. Values and Costs of Alternative Designs (per Tuple) as a Function of (Λ_1, Λ_2). The First Two Columns Correspond to $\alpha = 1/2$, and the Latter to $\alpha \sim U(0, 1)$

	(i) $\alpha = 1/2$		(ii) $\alpha \sim U(0, 1)$	
Design f	Cost $B(f)/T$	Value $V(f)$	Cost $B(f)/T$	Value $V(f)$
ϕ	0	0	0	0
{LWL}	1	0	1	$0.0088\ \Lambda_1 + 0.0175\ \Lambda_2$
{Ty}	3	0	3	$0.0121\ \Lambda_1 + 0.0756\ \Lambda_2$
{Ty, LWL}	4	$0.02722\ \Lambda_1 + 0.11662\ \Lambda_2$	4	$0.02986\ \Lambda_1 + 0.13107\ \Lambda_2$
{S, CL}	8	$0.07778\ \Lambda_1 + 0.155511\ \Lambda_2$	8	$0.06568\ \Lambda_1 + 0.16037\ \Lambda_2$
{S, CL, Ty}	11	$0.07778\ \Lambda_1 + 0.01944\ \Lambda_2$	11	$0.06568\ \Lambda_1 + 0.1944\ \Lambda_2$

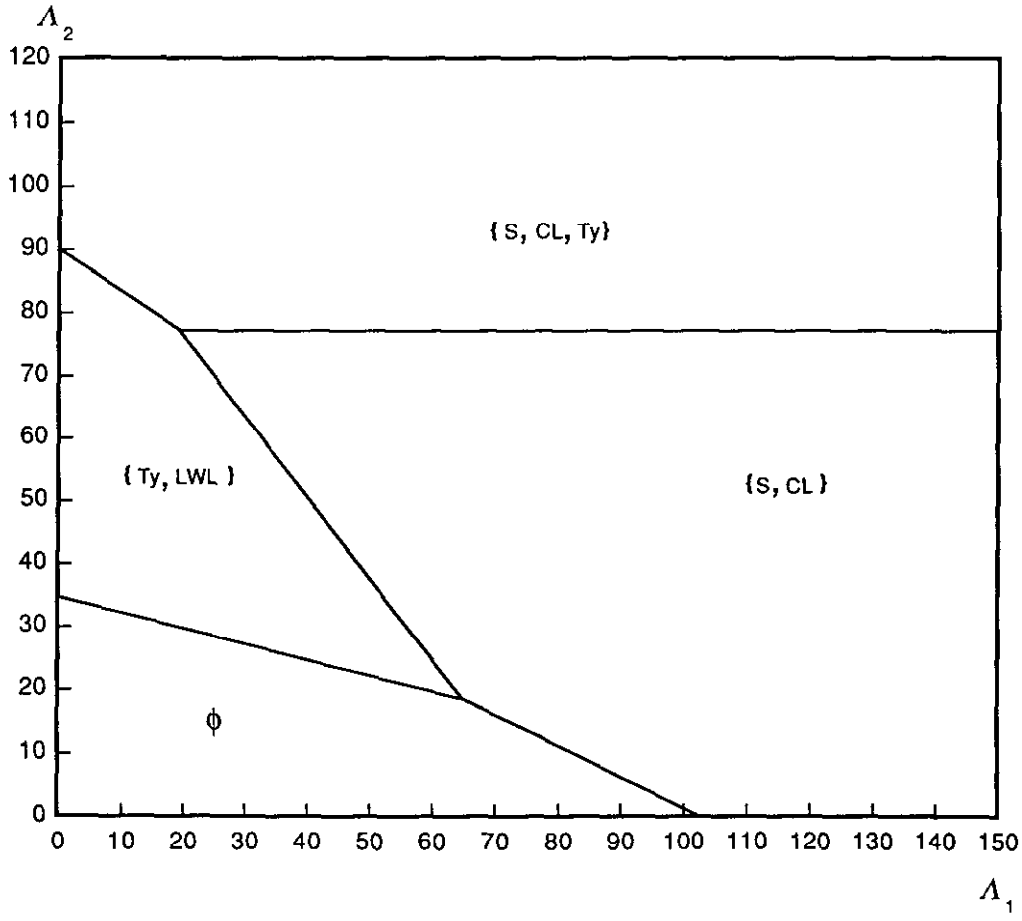

Fig. 2. Optimal design f^*, as a function of (Λ_1, Λ_2) for the ship example; $\alpha = 1/2$.

incomplete database information for each query under assumptions (i) and (ii) above; the detailed computations are given in Appendix A.

We define the *value* of a design f per tuple by $V(f) = [I(\phi) - I(f)]/T$ (i.e., the difference between the expected costs of incomplete database information per

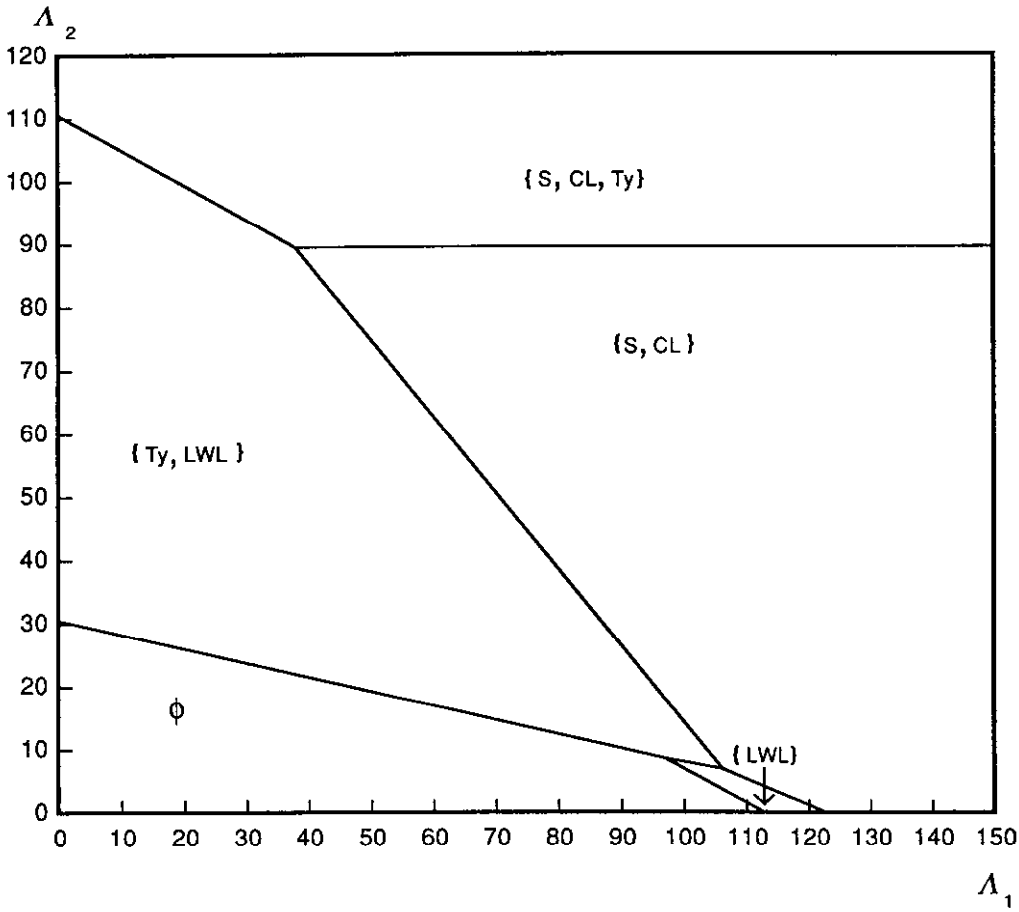

Fig. 3. Optimal design f^*, as a function of (Λ_1, Λ_2) for the ship example; $\alpha \sim U(0, 1)$.

tuple under the null design and under design f). The *net* value of design f per tuple is, obviously, $V(f) - B(f)/T$, and the optimal design f^* is the one which maximizes the net value function. Clearly, the optimal design f^* depends on the parameters Λ_1 and Λ_2, as well as on whether $\alpha = 1/2$ or $\alpha \sim U(0, 1)$. We examine for each of the latter cases the dependence of f^* on (Λ_1, Λ_2).

Table VIII summarizes the value and cost per tuple, $V(f)$ and $B(f)/T$, for the designs $f \in \mathcal{F}$ as a function of Λ_1 and Λ_2. We consider first case (i) where $\alpha = 1/2$, depicted in the first two columns of the table. Clearly, all the single-attribute designs are dominated by the empty design; optimal designs may consist of zero, two, or three attributes. Furthermore, if $\Lambda_2 = 0$, the optimal design is empty as long as $\Lambda_1 \leq 102.85$, and becomes {S, CL} (which are the relevant attributes for query Q_1) for all $\Lambda_2 \geq 102.85$. If $\Lambda_1 = 0$, the optimal design is empty for $\Lambda_2 \leq 34.3$; it becomes {Ty, LWL} for $34.3 \leq \Lambda_2 \leq 90$, and then becomes {S, CL, Ty} for $\Lambda_2 \geq 90$: in the range $34.3 \leq \Lambda_2 \leq 90$, indirect estimation of CL via LWL is more efficient due to the lower data-related cost of LWL (compared to CL). Ty is included in either case; this results from the desirability of avoiding the value-extraction error cost, β_{Ty}, associated with Q_2.

The optimal designs f^* as a function of (Λ_1, Λ_2), when $\alpha = 1/2$, are shown in Figure 2. When both Λ_1 and Λ_2 are very large, the design {S, CL, Ty}, including all the relevant attributes, is optimal. For smaller values of Λ_1 and Λ_2, we obtain one of the aforementioned designs with a preference for {Ty, LWL} when Λ_1 is small (since, for evaluating Q_2, it is more efficient to obtain the exact value of Ty while erring on CL, which is estimated via LWL), and a preference for {S, CL} when Λ_2 is small (since Ty is not required for the evaluation of Q_1).

The last two columns of Table VIII correspond to the case where $\alpha \sim U(0, 1)$; the optimal designs f^* as a function of (Λ_1, Λ_2) are depicted in Figure 3. Comparing Figure 3 to Figure 2, we observe that the general pattern is roughly the same, with ϕ, {Ty, LWL} and then {S, CL, Ty} being the optimal designs when $\Lambda_1 = 0$ and as Λ_2 increases. When $\Lambda_2 = 0$, the optimal designs are first ϕ, then {LWL}, and then {S, CL}. The design {LWL} becomes relevant since, for small values of α, we may obtain—under this design—$p(\pi, B_\pi, d_t) > \alpha$, whereas for $\alpha = 1/2$ we obtained the null answer throughout. Thus, the variability of α makes the results more sensitive to the specific design. Aside from this difference, the general pattern of behavior is similar to that of Figure 3, although the numerical results are, of course, different.

6. CONCLUDING REMARKS

This work has presented a methodology for approaching the conceptual design of databases using a decision theoretic framework. We suggest that the conceptual design problem inevitably involves a trade-off between the costs of incomplete database information and the data-related costs; our methodology provides systematic means for studying this trade-off in the design process. We illustrate our approach through a number of examples, which provide insight to the nature of this trade-off.

We believe that our approach has the potential of developing into an important design tool, but this agenda calls for further analysis based on the results presented here. First, our results clearly indicate some of the qualitative properties of the optimal design and its dependence on various query parameters and characteristics. These properties call for generalization into explicit design rules which would guide database administrators in performing this function. The results could have a major impact on the prevailing approach to database design, which starts from a given "universal relation" database which includes all the attributes of interest and seeks an equivalent representation (that is "better" in some design sense; (cf. [18])). We suggest that the decision on database attributes should be incorporated into this process rather than being performed separately on an ad-hoc basis. Further, since the physical design also has important implications on the data-related costs (cf. [1, 15]), this problem may also be treated in the same context.

A more immediate direction for further research focuses on the algorithmic aspects of the design problem, namely—on its efficient solution for large databases. A major part of this algorithmic task is the optimization problem itself, which may be formulated as a subset selection problem. The task then is to exploit the special structure of the database design problem and to suggest efficient computational procedures for its solution; this will be undertaken in a forthcoming paper.

APPENDIX A. Detailed Derivation of the Costs of Incomplete Database Information (Section 5)

A.1. Query Q_1

A.1.1. The null design $f = \phi$: Here, for each tuple t,

$$p(\pi, B_\pi, d_t) = \Pr\{\text{CL} = \text{M}\} \cdot \Pr\{\text{Sp} \geq 30\} = 7/45$$

(using Tables III and IV).

For $\alpha = 1/2$, we obtain

$$\underset{d}{E}\, g_{1/2}(p(\pi, B_\pi, d), 0) = 7/45 \cdot (1 - 1/2) = 7/90;$$

when $\alpha \sim U(0, 1)$, we have

$$\underset{d}{E}\, g(p(\pi, B_\pi, d), 0) = 1/2 \cdot 7/45 \cdot 38/45 = 133/2025 = 0.06568.$$

A.1.2. $f = \{\text{Ty}\}$: Here,

$$p(\pi, B_\pi, d_t) = \Pr\{\text{CL} = \text{M}\} \cdot \Pr\{\text{S} \geq 30 \mid \text{Ty}\} = \begin{cases} 0 & \text{for Ty = carrier} \\ 14/45 & \text{for Ty = sub.} \end{cases}$$

Hence, for $\alpha = 1/2$, $g_{1/2}(p(\pi, B_\pi, d_t), 0) = 0$ for Ty = carrier, and $g_{1/2}(p(\pi, B_\pi, d_t), 0) = 7/45$ for Ty = sub. Together, $E_d g_{1/2}(p(\pi, B_\pi, d), 0) = 7/90$, with a *null* answer set. When $\alpha \sim U(0, 1)$, we obtain

$$\underset{d}{E}\, g(p(\pi, B_\pi, d), 0) = 1/2 \cdot 14/45 \cdot 31/45 = 0.05358.$$

A.1.3. $f = \{\text{LWL}\}$: Here,

$$p(\pi, B_\pi, d_t) = \Pr\{\text{CL} = \text{M} \mid \text{LWL}\} \cdot \Pr\{\text{S} \geq 30\} = \begin{cases} 0.8 \cdot 0.4 = 0.32 & \text{if LWL = M} \\ 0.2 \cdot 0.4 = 0.08 & \text{if LWL = I} \\ 0 & \text{if LWL = P} \\ 0.1 \cdot 0.4 = 0.04 & \text{if LWL = A.} \end{cases}$$

For $\alpha = 1/2$, we obtain

$$\underset{d}{E}\, g_{1/2}(p(\pi, B_\pi, d_t), 0) = 7/18 \cdot 0.16 + 5/18 \cdot 0.04 + 2/9 \cdot 0.02 = 0.07778.$$

When $\alpha \sim U(0, 1)$,

$$\underset{d}{E}\, g_{1/2}(p(\pi, B_\pi, d_t), 0) = 7/18 \cdot 0.1088 + 5/18 \cdot 0.0368 + 2/9 \cdot 0.0192 = 0.0568.$$

A.1.4. $f = \{\text{Ty, LWL}\}$: Here,

$$p(\pi, B_\pi, d_t) = \Pr\{\text{CL} = \text{M} \mid \text{LWL}\} \cdot \Pr\{S \geq 30 \mid \text{Ty}\},$$

which is 0 if Ty = carrier or if LWL = P. Otherwise,

$$p(\pi, B_\pi, d_t) = \begin{cases} 0.08 & \text{for Ty = sub, LWL = A} \\ 0.16 & \text{for Ty = sub, LWL = I} \\ 0.64 & \text{for Ty = sub, LWL = M.} \end{cases}$$

For $\alpha = 1/2$, the *answer* will include all tuples with Ty = sub and LWL = M;

also,

$$g_{1/2}(p(\pi, B_\pi, d_t), 0) = \begin{cases} 0.08 \cdot 0.5 = 0.04 & \text{for Ty = sub, LWL = A} \\ 0.16 \cdot 0.5 = 0.08 & \text{for Ty = sub, LWL = I} \\ 0.36 \cdot 0.5 = 0.18 & \text{for Ty = sub, LWL = M} \\ 0 & \text{otherwise,} \end{cases}$$

hence,

$$\underset{d}{E}\, g_{1/2}(p(\pi, B_\pi, d), 0)) = 1/2 \cdot 2/9 \cdot 0.04 + 1/2 \cdot 5/18 \cdot 0.08 + 1/2 \cdot 7/18 \cdot 0.18 = 0.05056.$$

When $\alpha \sim U(0, 1)$, a similar calculation yields

$$\underset{d}{E}\, g(p(\pi, B_\pi, d), 0) = 0.03582.$$

A.1.5. $f \supseteq \{S, CL\}$: Here, the answer will always include the ships satisfying the exact retrieval criteria, and consequently the cost of incomplete database information is zero.

A.2. Query Q_2. Consider a given design f and a tuple t. The tuple will be selected for display if and only if

$$p(\pi, B_\pi, d_t) \geq \alpha + \beta_{\text{Ty}} \cdot m_{\text{Ty}}(d_t), \tag{A.1}$$

where $1 - m_{\text{Ty}}(d_t)$ is the probability mass of the mode of the conditional distribution of Ty given d_t. The corresponding expected cost is

$$g_{1/2}(p, \delta) = \min\{p, 1/2 + \delta\} - 1/2p,$$

when $\alpha = 1/2$, or the function $g(p, \delta)$ given by (4.4) when $\alpha \sim U(0, 1)$, where

$$p = p(\pi, B_\pi, d_t) \quad \text{and} \quad \delta = \beta_{\text{Ty}} \cdot m_{\text{Ty}}(d_t) = m_{\text{Ty}}(d_t).$$

We now examine the expected cost per tuple as we vary the design f.

A.2.1. $f = \phi$: Here, $p(\pi, B_\pi, d_t) = \Pr\{\text{CL = M}\} = 7/18$ and $m_{\text{Ty}}(d_t) = 1/2$. This implies that (A.1) will not be satisfied, hence the answer set will always be empty. The corresponding expected cost is 7/36 both when $\alpha = 1/2$ and when $\alpha \sim U(0, 1)$.

A.2.2. $f = \{\text{Ty}\}$: Here, $p(\pi, B_\pi, d_t) = \Pr\{\text{CL = M}\} = 7/18$, with an accurate value of Ty. When $\alpha = 1/2$, the answer set is null and the expected cost is 7/36. When $\alpha \sim U(0, 1)$, we have $g(p(\pi, B_\pi, d_t), \delta) = g(7/18, 0) = 1/2 \cdot 7/18 \cdot 11/18 = 0.1188$.

A.2.3. $f = \{\text{LWL}\}$: Here, $p(\pi, B_\pi, d_t) = \Pr\{\text{CL = M} \mid \text{LWL}\}$, which is equal to 0.1 when LWL = A, 0 when LWL = P, 0.2 when LWL = I, and $p(\pi, B_\pi, d_t) = 0.8$ when LWL = M. For $\alpha = 1/2$, condition (A.1) cannot be satisfied since $m_{\text{Ty}}(d_t) = 1/2$, hence the answer set will be null and the expected cost will be 7/36. When $\alpha \sim U(0, 1)$, we have

$$g(p, \delta) = \begin{cases} 1/2 \cdot 0.1 = 0.05 & \text{LWL = A} \\ 0 & \text{LWL = P} \\ 1/2 \cdot 0.2 = 0.1 & \text{LWL = I} \\ 1/2 \cdot 0.8 - 1/2 \cdot (0.8 - 0.5)^2 = 0.355 & \text{LWL = M} \end{cases}$$

and the expected cost is thus:

$$2/9 \cdot 0.5 + 5/18 \cdot 0.1 + 7/18 \cdot 0.355 = 0.1769.$$

Note that the optimal answer set will be null whenever $\alpha \geq 0.3$; for $\alpha < 0.3$, the answer will include all ships with LWL = M, with an arbitrary type.

A.2.4. $f = \{\text{Ty, LWL}\}$: Here, $p(\pi, B_\pi, d_t)$ are the same as in A.2.3, but $m_{\text{Ty}}(d_t) = 0$ since Ty is known with certainty. When $\alpha = 1/2$, (A.1) is satisfied only when LWL = M ($p = 0.8$), hence only ships with LWL = M will be selected for display (with their accurate Type provided in response to the query). The corresponding expected cost per tuple is

$$2/9 \cdot 0.05 + 1/9 \cdot 0 + 5/18 \cdot 0.1 + 7/18 \cdot 0.1 = 0.07778.$$

When $\alpha \sim U(0, 1)$, since $p \geq \delta$ for all LWL, we have $g(p, \delta) = 1/2\ p - 1/2(p - \delta)^2 = 1/2\ p(1 - p)$, hence the expected cost per tuple is

$$2/9 \cdot 1/2 \cdot 0.1 \cdot 0.9 + 5/18 \cdot 1/2 \cdot 0.2 \cdot 0.8 + 7/18 \cdot 1/2 \cdot 0.8 \cdot 0.2 = 0.06333.$$

Note that here the only cost component is the retrieval cost, since there are no errors in the estimation of Ty.

A.2.5. $f = \{\text{S, CL}\}$: Since CL is known with certainty, the retrieval cost component is zero, and we can focus on the value-extraction cost. Further, S can be useful in the estimation of Ty (see Table V): if $20 \leq S_t < 25$, then Type must be carrier (with probability 1), and $m(d_t) = 0$ (i.e., there is no estimation error). If $25 \leq S_t < 30$, the modal Type is carrier, and $m(d_t) = 0.25$. If $30 \leq S_t < 40$, then Type must be sub (with probability 1), and again $m(d_t) = 0$.

Let $\alpha = 1/2$. Since CL is known with certainty, the selection criterion (A.1) becomes: CL = M and $m_{\text{Ty}}(d_t) \leq 1/2$; but $m_{\text{Ty}}(d_t)$ is at most 1/4, hence any tuple with CL = M will be selected, and $g_{1/2}(p, \delta) = \delta$, which is positive (1/4) only if $25 \leq S_t < 30$. Thus the expected cost is $\Pr\{\text{CL} = \text{M}\} \cdot \Pr\{25 \leq S < 30\} \cdot 1/4 = 0.038889$.

When $\alpha \sim U(0, 1)$, the selection criterion (A.1) becomes CL = M and $1 \geq \alpha + m_{\text{Ty}}(d_t)$. This means that any tuple with CL = M and $20 \leq S_t < 25$ or CL = M and $30 \leq S_t < 40$ will be selected; tuples with CL = M and $25 \leq S_t < 30$ will be selected if $\alpha \leq 3/4$, and rejected if $\alpha > 3/4$. The resulting expected cost is $7/18 \cdot 0.4 \cdot g(1, 1/4) = 0.03403$.

Clearly, the same result holds for $f = \{\text{S, CL, LWL}\}$.

A.2.6. $f = \{\text{S, CL, Ty}\}$ or $f = \{\text{S, CL, LWL, Ty}\}$: The expected cost of incomplete database information is clearly zero in these cases.

APPENDIX B. Summary of Notation

x_t	Information object t.
d_t	Database tuple t.
f	The design mapping; $d_t = f(x_t)$.
$Q(\pi, B_\pi, \eta)$	The query "find all information objects t such that $\pi(x_t) \in B_\pi$; display t and $\eta(x_t)$."
θ	Information request; $\theta = (Q, \mu)$.
a_r	The answer set consisting of all the tuples selected in response to a query.

$c(a; \theta, D)$	Expected cost associated with answer a to information request θ when the database state is D.
$\lambda(\theta)$	Frequency of information request θ.
$I(f)$	Expected cost of incomplete database information under design f.
$B(f)$	Expected data-related cost under design f.
$\mathscr{F}$	Set of feasible designs.
f^*	Optimal design, minimizing $I(f) + B(f)$ among $f \in \mathscr{F}$.
h_t	Attribute values displayed for tuple t in a value-extracting query.
$s = s(\theta)$	Cost multiplier for information request θ.
α	Relative cost of false alarm.
$1 - \alpha$	Relative cost of omission.
$c_v(h_t, x_t, \eta)$	Value-extraction cost for tuple t.
$\bar{c}_v(\eta, d_t)$	Minimal expected value-extraction cost for tuple t.
$p(\pi, B_\pi, d_t)$:	$\Pr\{\pi(x_t) \in B_\pi \mid d_t\}$, the probability that tuple t satisfies the retrieval criteria.
$g_\alpha(p, \delta)$	Expected cost of incomplete database information when $p(\pi, B_\pi, d_t) = p$ and the expected value-extraction cost is δ.
$g(p, \delta)$	$E_\alpha\, g_\alpha(p, \delta)$.
$G(p)$	$g(p, 0)$ (for retrieval queries).
$\Lambda(\theta)$	$\lambda(\theta)s(\theta)$, the frequency-weighted importance of information request θ.

REFERENCES

1. CHEN, P. P., AND YAO, S. B. Design and performance tools for database systems. In *Proceedings of the International Conference on Very Large Databases*, 1977, 3–15.
2. CHOW, D., AND YU, C. T. On the construction of feedback queries. *J. ACM 29* (1982), 127–151.
3. CODD, E. F. Extending the database relational model to capture more meaning. *ACM Trans. Database Syst. 4*, 4 (Dec. 1979), 395–434.
4. HEINE, M. H. Design equations for retrieval systems based on the Swets model. *J. Am. Soc. Inf. Sci. 25* (1974), 183–198.
5. IMIELINSKI, T., AND LIPSKI, W., JR. Incomplete information in relational databases. *J. ACM 31*, 4 (Oct. 1984), 761–791.
6. KARLIN, S., AND TAYLOR, H. M. *A First Course in Stochastic Processes*. Academic Press, New York, 1975.
7. KRAFT, D. H. A threshold rule applied to the retrieval decision model. *J. Am. Soc. Inf. Sci. 29* (1978), 77–80.
8. KRAFT, D. H., AND BOOKSTEIN, A. Evaluation of information retrieval systems: A decision theory approach. *J. Am. Soc. Inf. Sci. 29* (1978), 31–40.
9. LIPSKI, W. On semantic issues connected with incomplete information databases. *ACM Trans. Database Syst. 4*, 3 (Sept. 1979), 262–296.
10. LIPSKI, W. On databases with incomplete information. *J. ACM 28*, 1 (Jan. 1981), 41–70.
11. MAIER, D. *The Theory of Relational Databases*. Computer Science Press, Rockville, Md., 1984.
12. MARSCHAK, J., AND RADNER, R. *Economic Theory of Teams*. Yale University Press, New Haven, Conn., 1972.
13. SALTON, G. *Dynamic Information and Library Processing*. Prentice-Hall, Englewood Cliffs, N.J., 1976.
14. SALTON, G., BUCKLEY, C., AND YU, C. T. An evaluation of term dependence models in information retrieval. In *Research and Development in Information Retrieval, Lecture Notes in Computer Science, 146*, G. Salton and H.-J. Schneider, Eds., Springer-Verlag, New York, 1983.

15. SCHKOLNICK, M. A survey of physical database design methodology and techniques. In *Proceedings of the International Conference on Very Large Databases*, 1978, 474-487.
16. SWETS, J. A. Information retrieval systems. *Science 241* (1963), 245–250.
17. SWETS, J. A. Effectiveness of information retrieval methods. *Am. Doc. 20* (1969), 72–89.
18. ULLMAN, J. D. *Principles of Database Systems*. 2nd ed., Computer Science Press, Rockville, Md., 1982.
19. VASSILIOU, Y. Null values in data base management: A denotational semantics approach. In *Proceedings of the ACM-SIGMOD International Symposium on Management of Data* (Boston, Mass., May 30–June 1, 1979), ACM, New York, 162–169.
20. VASSILIOU, Y. Functional dependencies and incomplete information. In *Proceedings of the 6th International Conference on Very Large Databases* (Montreal, Oct. 1–3, 1980), ACM, New York, 260–269.
21. WINKLER, R. L., AND HAYS, W. L. *Statistics*. Holt, Rinehart & Winston, New York, 1975.
22. WONG, E. A statistical approach to incomplete information in database systems. *ACM Trans. Database Syst. 7*, 3 (Sept. 1982), 470–488.
23. YU, C. T., LUK, W. S ., AND SIU, M. K. On models of information retrieval. *Inf. Syst. 4* (1979), 205–218.

Received March 1985; revised October 1985; accepted November 1985