Summarizing Relational Databases

Xiaoyan Yang
National Univ. of Singapore
Republic of Singapore, 117543
yangxia2@comp.nus.edu.sg

Cecilia M. Procopiuc
AT&T Labs-Research
Florham Park, NJ 07932 USA
magda@research.att.com

Divesh Srivastava
AT&T Labs-Research
Florham Park, NJ 07932 USA
divesh@research.att.com

ABSTRACT

Complex databases are challenging to explore and query by users unfamiliar with their schemas. Enterprise databases often have hundreds of inter-linked tables, so even when extensive documentation is available, new users must spend a considerable amount of time understanding the schema before they can retrieve any information from the database. The problem is aggravated if the documentation is missing or outdated, which may happen with legacy databases.

In this paper we identify limitations of previous approaches to address this vexing problem, and propose a principled approach to summarizing the contents of a relational database, so that a user can determine at a glance the type of information it contains, and the main tables in which that information resides. Our approach has three components: First, we define the importance of each table in the database as its stable state value in a random walk over the schema graph, where the transition probabilities depend on the entropies of table attributes. This ensures that the importance of a table depends both on its information content, and on how that content relates to the content of other tables in the database. Second, we define a metric space over the tables in a database, such that the distance function is consistent with an intuitive notion of table similarity. Finally, we use a Weighted k-Center algorithm under this distance function to cluster all tables in the database around the most relevant tables, and return the result as our summary. We conduct an extensive experimental study on a benchmark database, comparing our approach with previous methods, as well as with several hybrid models. We show that our approach not only achieves significantly higher accuracy than the previous state of the art, but is also faster and scales linearly with the size of the schema graph.

1. INTRODUCTION

Enterprise databases often consist of hundreds of inter-linked tables, which makes them challenging to explore and query by new users. Even when extensive documentation is available (and this is by no means the rule), there is a steep learning curve before users can interact with the system.

Recent database work has recognized the importance of developing tools that can help users significantly reduce the time they spend understanding an unfamiliar system. A strong research direction considers so-called *structure-free query models*, that allow the user to pose queries without having to understand the schema structure. The simplest such model is *keyword search* [1]. Increasingly more sophisticated mechanisms, such as *query relaxation* [3] and *meaningful summary query* [12], allow users to pose more complex queries, that include non-trivial structure requirements (e.g., joins). While the majority of the work in this area is for XML schemas, recent results have also specifically addressed relational databases [7]. An excellent overview of many other approaches can be found in [6].

However, the ability to pose queries is not the only end-goal of users interacting with unfamiliar systems. While tools that improve the usability of a system have clear and immediate benefits, their very goal is to shield users from the complexity of the underlying structure. This is perfectly acceptable for casual users, but insufficient for users who wish to familiarize themselves with a complex schema. Such users may include new database administrators, analysts trying to extract useful information out of the database, or developers who want to create new tools and forms on top of that database. While they may eventually need to acquire detailed knowledge of the system, their ability to do so would be greatly improved if they could start with a simplified, easy-to-read schema. (We ourselves have benefited from such an approach while becoming acquainted with the TPCE benchmark we use in this paper: we started by understanding the schema in Figure 2, before reading several dozen pages of documentation). Early work in this area has focused on ER model abstraction [2]. However, unlike ER models, XML and relational schemas do not have semantic information attached to the schema edges. Recognizing the importance of *schema summarization*, Yu and Jagadish [11] proposed the first algorithms for automatically creating database summaries. Further discussion of related work appears in Section 3.

The method of [11] was developed for XML schemas, but, as the authors state, it can also be applied to relational databases. However, relational schemas come with specific challenges that are not usually encountered in XML schemas. In fact, the two main assumptions on which the model of [11] is based can both fail, even on relatively simple relational schemas. The reasons for such failures are intrinsic to the design and functionality of database systems, so they cannot be avoided. We defer a detailed discussion of this issue to Section 3, as we need to introduce several notations and definitions in order to better explain it. In addition to these intrinsic differences between XML and relational schemas, there are additional challenges that arise from inconsistent or missing data in relational databases. In particular, the existence of schema-level join relationships does not necessarily imply that all instances in a table satisfy that join. As our experiments show in Section 7, the method of [11] does not work well on the benchmark relational schema over which we conduct our study. Therefore, we propose a novel approach for summarizing relational schemas, that addresses the specific challenges associated with them in a principled manner. Our contributions are as follows:

- We propose a new definition for the importance of tables in a relational schema. The definition is based on information theory and statistical models, and reflects the information content of a table, as well as how that content relates to the content of other tables.
- We define a metric space over tables in a relational schema, by introducing a novel distance function that quantifies the similarity between tables. We believe this distance may be of independent interest.
- We propose using a Weighted k-Center algorithm for schema summarization, and conduct an extensive experimental study using the TPCE benchmark data to justify our approach.

2. NOTATIONS

The schema graph $G = (\mathcal{R}, \mathcal{E})$ is defined in the usual way for relational databases: The nodes correspond to tables $R \in \mathcal{R}$, and the (undirected) edges to database joins. Each table R is a $n \times m$ matrix, where the columns are the attributes $A_1, \dots, A_m$ associated with R, and the rows are the table tuples $\tau_1, \dots, \tau_n$. We denote by $R.A_i$ the column corresponding to an attribute A_i in table R. If there is a join relationship between table R on attribute A and table S on attribute B, then the corresponding edge in the schema graph is labeled $R.A - S.B$ (note that in general, there may be multiple edges between the same pair of nodes, labeled by different join attributes).

For example, all four cases in Figure 1 correspond to the same schema graph that consists of two nodes and one edge $R.A - S.B$. However, while the schema graph is the same, different cases arise at the tuple level. We call the graphs in Figure 1 *instance-level graphs*: they are obtained by drawing the join edges between tuples, rather than tables. If a pair of tuples $\tau \in R$ and $\tau' \in S$ satisfies the join condition $R.A - S.B$, we say that the tuples τ, τ' *instantiate* the schema edge. For example, in Figure 1 (d), the first tuple of R and the first tuple of S instantiate the edge $R.A - S.B$. For a fixed tuple $\tau \in R$, the *fanout* of τ along edge $R.A - S.B$ is the number of different tuples $\tau' \in S$ so that (τ, τ') instantiate this edge. Thus, in Figure 1 (d), the first tuple of S has fanout 4, while the last tuple of S has fanout 0 (this tuple does not instantiate the edge).

DEFINITION 1. *Let $e = R.A - S.B$ be an edge in the schema graph $G = (\mathcal{R}, \mathcal{E})$. The* average fanout *of R with respect to e is defined as*

$$af_e(R) = \frac{\sum_{i=1}^{n} fanout_e(\tau_i)}{n},$$

where $\tau_1, \dots, \tau_n$ are all the tuples of R, and $fanout_e(\tau_i)$ is the fanout of τ_i along edge e.

Let q be the number of tuples in R for which $fanout_e(\tau_i) > 0$. Then the matching fraction *of R with respect to e is $f_e(R) = q/n$, and the* matched average fanout *of R with respect to e is*

$$maf_e(R) = \frac{\sum_{i=1}^{n} fanout_e(\tau_i)}{q}.$$

If the edge e is clear from the context, we omit it from the notation. By definition, for any edge e, $f_e(R) \leq 1$ and $maf_e(R) \geq 1$. However, $af_e(R) = f_e(R) \cdot maf_e(R)$ can be either larger or smaller than 1; see, for example, $af(S)$ in Figures 1 (c) and (d). This observation will be used in Section 5, and underscores the computational efficiency of our approach.

If the edge $R.A - S.B$ is instantiated as in Figures 1 (a) and (c), it is usually called a **pk-pk edge**, since $R.A$ and $S.B$ are both primary keys. If it is instantiated as in Figures 1 (b) and (d), it is called an **fk-pk edge**, since only one of its endpoints (i.e., $S.B$) is a primary key, while the other endpoint is a foreign key.

REMARK 1. *A join relationship between two tables may involve multiple attributes, e.g., "R.A = S.B and R.C = S.D." In that case, the edge in the graph is labeled by all the attributes involved, and two tuples $\tau \in R$ and $\tau' \in S$ instantiate the edge if they satisfy all the join conditions. Definition 1 extends naturally to such edges. Our experiments handle edges between multiple attributes.*

Throughout this paper we use the TPCE benchmark schema [10], both to illustrate our ideas, and for experimental purposes. The schema graph is shown in Figure 2. It consists of 33 tables, pre-classified in four categories: *Broker, Customer, Market* and *Dimension*. The database models a transaction system in which customers trade stocks. Various additional information is stored about customers, securities, brokers, a.s.o. Join attributes are enumerated below each table name; tables also contain non-joining attributes (not depicted), which we will discuss whenever necessary. Edges are shown directed from foreign key to key. The directions are for ease of reading only: in our experiments, we consider the graph undirected.

The reason for using this schema is that it allows us to develop objective measures for the accuracy of summarization approaches, by comparing the generated summaries with the pre-defined table classification provided as part of the benchmark. Previous measures for summary accuracy were based on query logs, by relating the significance of a table to its frequency in the log. In the next section, we discuss scenarios in which this is not necessarily true for a relational database. By contrast, the human-designed pre-classification of TPCE is as close to the ground truth of summarization as we expect to find.

Figure 2 illustrates the typical approach for summarizing a relational schema: By *clustering* the tables into a few *labeled* categories (and color-coding the graph), the result gives any casual reader a rough idea about what the database represents. This classification was done manually by the designer of the benchmark. More importantly, the category labeling was also decided by the designer. In this paper, we propose a statistical model that allows us to automatically classify and label tables in a schema. We evaluate the accuracy of our approach, as well as that of competing methods, by comparing the automatically generated summaries with the one in Figure 2. Formally, we define a summary as follows.

DEFINITION 2. *Given a schema graph $G = (\mathcal{R}, \mathcal{E})$ for a relational database, a summary of G of size k is a k-clustering $\mathcal{C} = \{C_1, \dots, C_k\}$ of the tables in $\mathcal{R}$, such that for each cluster C_i a representative table $center(C_i) \in C_i$ is defined. The summary is represented as a set of* labels

$$\{center(C_1), \dots, center(C_k)\},$$

and by a function $cluster : \mathcal{R} \rightarrow \mathcal{C}$ which assigns each table to a cluster.

(This definition is consistent with the one in [11], but translated into our notations).

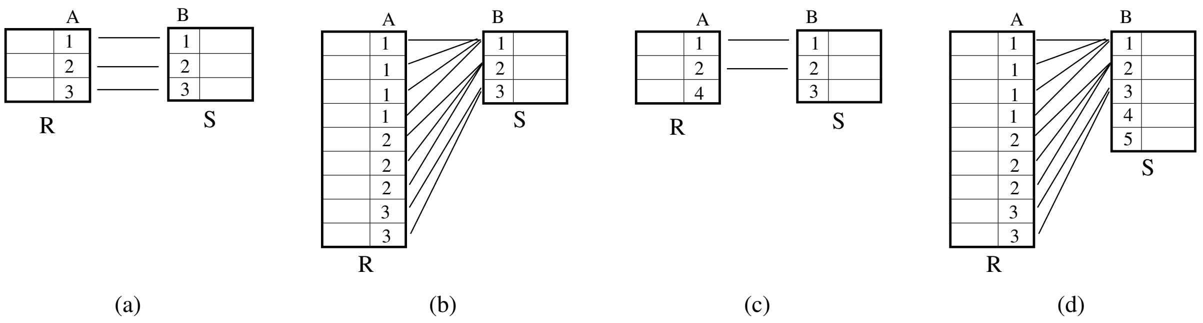

Figure 1: Four instance-level graphs for the schema graph $(\{R, S\}, \{R.A - S.B\})$: (a) $af(R) = maf(R) = f(R) = 1$, $af(S) = maf(S) = f(S) = 1$; (b) $af(R) = maf(R) = f(R) = 1$, $af(S) = maf(S) = (4+3+2)/3 = 3$, $f(S) = 1$; (c) $af(R) = af(S) = 2/3$, $maf(R) = maf(S) = 1$, $f(R) = f(S) = 2/3$; (d) $af(R) = maf(R) = f(R) = 1$, $af(S) = (4+3+2)/5 = 9/5$, $maf(S) = (4+3+2)/3 = 3$, $f(S) = 3/5$.

3. A PRINCIPLED APPROACH

Clearly, not all tables in a database have the same importance. However, it is not always easy to decide between two tables which one is more important. Different users may have different opinions. To design an automatic process for schema summarization, we propose starting from a few basic principles in order to build a complex model. The first principle is that, at the most basic level, a database table is characterized by its *attributes*, its *tuples*, and its *join relations*. We therefore need a way of integrating information about all three into a single measure of table importance.

The second principle is that there exist a few database tables for which the majority of humans can agree that they either have a lot of importance, or very little importance. For example, in the TPCE schema, table *Status_Type* has 2 attributes and five tuples, while table *Trade* contains 14 attributes and $\Theta(10^6)$ tuples. We believe that most people would readily agree that *Status_Type* has very small importance, while *Trade* is very important.

Any automatic model must distinguish between tables that humans easily classify as having high, resp. low, importance. Thus, we arrive at our third principle: compare pairs of clearly important and clearly unimportant tables, and include in the model those features that always differentiate between them. We illustrate this process on the TPCE schema. One postulate in [7, 11] is that the importance of a table is proportional to its number of tuples. However, table *Trade_History* is one of the largest, with $\Theta(10^7)$ tuples, but a casual examination would convince users that it is not a particularly relevant table: it contains only old transactions, which in a real-world system could also be stale. On the other hand, table *Customer*, with $\Theta(10^3)$ tuples, is quite important to the database: it contains information on the people who initiate all the transactions in the system. More precisely, *Customer* has 23 attributes, the most of any table; by comparison, *Trade_History* only has 2 attributes. Using our third principle, it appears that the table importance should be proportional to the number of attributes. However, a more in-depth analysis shows that among the 23 attributes, table *Customer* contains a 'status id' attribute which has only 5 distinct values, and a 'customer id' attribute which has 1000 distinct values. Clearly, they shouldn't count equally towards the importance of the table. Rather, the amount of information contained in each attribute should count towards table importance. Since entropy is the well-known measure for information [4], we propose that *a table's importance should be proportional to the sum of its attribute entropies*. However, we do not wish to completely ignore the contribution of the number of tuples to the importance of a table, as that would be unreasonable. We propose to strike a balance between tuples and attributes by allowing the number of tuples, dampened by the log function, to be added to the importance. The log function insures that the number of tuples does not dominate the entropy values. Formal definitions are in Section 4.

A second postulate of previous work is that the importance of a table is proportional to its connectivity in the schema graph. However, table *Status_Type* has 6 join edges - the second most in the schema - yet it is arguably the least significant of the entire database. But, just as with the first postulate, we do not wish to completely negate this principle, as we do need to quantify how join relations contribute to table importance. Ignoring their contribution altogether would lead to a situation in which table *Trade_History*, due to its very large number of tuples and high attribute entropy, would rank in the top third. Since this table has only one join edge, it becomes clear that its connectivity must play a role in dampening its importance. By contrast, a table such as *Company*, which also ranks in the top third based on its tuples and attributes, should gain importance via its connectivity. How can we then automatically distinguish between the high connectivity of *Company*, which should allow it to increase its importance, and the high connectivity of *Status_Type*, which should play only a minor role? Previous methods distinguished edges based on their average fanout. As we discuss in the following sections, and show via experiments, tables such as *Status_Type* significantly increase their importance in this model: not the desired result. Instead, we propose that it is not the number, but the quality, of join instantiations that counts. Thus, we assign weights to join edges proportional to the entropies of the participating attributes.

Stable distribution in random walks The discussion above illustrates the need to take into account both the information content of a table, and its (weighted) join edges, to come up with a single value for the importance of the table. Intuitively, join edges can be viewed as vehicles for information transfer between tables, since their weights depend on the entropies of their respective attributes. It is therefore natural to define a random walk process on the schema graph, whereby each table starts with its information content, and then repeatedly sends and receives information along its join edges, proportional to their weight. If the underlying schema graph is connected and non-bipartite, it is well known that this process converges to a stable distribution. We will define the *importance* of a table as the value of this stable distribution for that table.

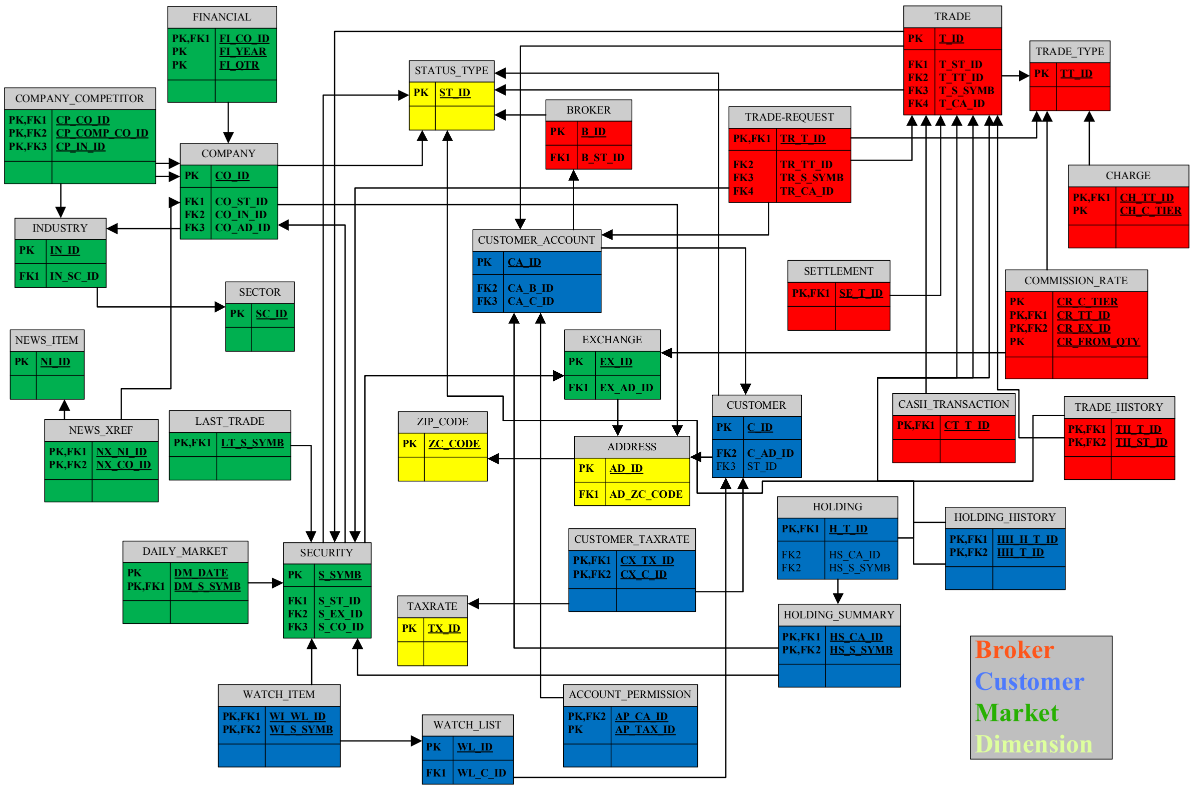

Figure 2: TPCE schema graph.

Previous Work The most closely related work is the schema summarization proposed in [11], in the context of XML schemas. Although the authors discuss how the approach extends to relational database schemas, their model makes the two crucial assumptions we mentioned above: that the importance of a table is proportional to its number of tuples, and to its number of join edges. These are reasonable assumptions for XML schemas, but they can both fail in relational databases: In data warehouse systems, the largest tables are often those containing historical, possibly stale, data (such as *Table_History*). Moreover, enterprise systems tend to have many so-called *dimension tables* (such as *Status_Type* and *Zip_Code*), i.e., tables with only a few attributes that contain companion information for other tables in the schema. Dimension tables are usually highly connected, but have little relevance for a summary. As a side note, the definition of table importance in this model is equivalent to the stable distribution of a random walk (different than the one we propose), though it is not recognized as such in [11].

Another recent result [7] applies the same two assumptions specifically to relational schemas, in the context of computing the so-called querability of a table or attribute; i.e., how likely the table/attribute is to appear in a representative query log. This value is then used to automatically design forms for querying the database. We do not know how likely these assumptions are to succeed or fail in this context. However, we note that the querability of a table is not necessarily correlated to its relevance in a schema summary. For example, one might frequently compute statistics on the transactions of customers in a specific town, requiring all such queries to include a join with table *Zip_Code*. However, we believe few people would consider this table important enough to include in a summary.

An important contribution of this paper is a definition of a metric space over database tables, so that the distance function is consistent with an intuitive notion of table similarity. Other graph-based notions of similarity have been proposed in the IR community. Koren et al. [9] define the similarity between two nodes s and t to be proportional to the probability that a random walk starting at s reaches t without revisiting s. Such a definition of similarity works well when all edges represent the same kind of relation (e.g., number of phone calls between two customers). However, in the case of schema graphs, different join edges represent different conceptual relations, so the definition is less useful. We have implemented the method of [9], and include it as part of our extensive comparison study.

4. TABLE IMPORTANCE

The entropy of an attribute A in table R is defined as $H(R.A) = \sum_{i=1}^{k} p_i \log(1/p_i)$, where $R.A = \{a_1, \ldots, a_k\}$ are all the values of attribute $R.A$, and p_i is the fraction of tuples in R that have value a_i on attribute A. For example, in Figure 1(d), $H(R.A) = (4/9)\log(9/4) + (1/3)\log 3 + (2/9)\log(9/2) \approx 1.53$.

For each table R we create a primary key $R.Key$ consisting of all the attributes in the table, and add a self-loop $R.Key - R.Key$ in the schema graph (we do this even if R already has a primary key, for consistency). Intuitively, this self-loop serves to keep a certain amount of information in the table. It also serves to add, in a uniform way, the contribution of the number of tuples $|R|$ to table importance; see below.

DEFINITION 3. *We define the* information content *of a table* R

as

$$IC(R) = \log|R| + \sum_{R.A} H(R.A),$$

where $|R|$ is the number of tuples in R, and the sum ranges over all attributes $R.A$ in table R. Hence, $IC(R)$ is the sum of entropies of all the attributes $R.A$, plus the entropy of $R.Key$, which is $\log|R|$.

Intuitively, the value $IC(R)$ is the importance of table R in the absence of any join information. As discussed in the previous section, we must also take into account the information transfer between tables in a connected schema graph. More precisely, we define an $n \times n$ probability matrix Π, where n is the number of tables in the database (by probability matrix we mean a matrix of non-negative numbers so that each row sums up to 1). There is a non-zero value on position $\Pi[R, S]$ if and only if there is at least one join edge between tables R and S in the schema. The value $\Pi[R, S]$ reflects the "information transfer" along such an edge. If there are multiple join edges between R and S, the information transfer along all edges is summed up. The exact definition is as follows.

DEFINITION 4. *Let G denote a schema graph. The* entropy transfer matrix Π *associated with G is defined as follows: Let $e = R.A - S.B$ be a join edge in G. Let q_A denote the total number of join edges involving attribute R.A, including the edge $R.Key - R.Key$. We define*

$$\text{(VE)}\ \ Pr(R.A \to S.B) = \frac{H(R.A)}{\log|R| + \sum_{R.A'} q_{A'} \cdot H(R.A')} = \frac{H(R.A)}{IC(R) + \sum_{R.A'}(q_{A'} - 1) \cdot H(R.A')}$$

(where the sum ranges over all attributes $R.A'$ of table R). We refer to this model as the variable transfer entropy *model, or (VE), for reasons that we explain later in this section. We then define*

$$\Pi[R, S] = \sum_{R.A - S.B} Pr(R.A \to S.B),$$

where the sum ranges over all edges between R and S in the graph. In addition, we define

$$\Pi[R, R] = 1 - \sum_{S \neq R} \Pi[R, S].$$

to account for the edge $R.Key - R.Key$.

We now define the importance of a table R as the stable-state value of a random walk on G, using probability matrix Π. More precisely,

DEFINITION 5. *Let $\mathcal{I}$ denote the stationary distribution of the random walk defined by the probability matrix Π, i.e., $\mathcal{I}$ is the (row) vector with the property that $\mathcal{I} \times \Pi = \mathcal{I}$. The* importance *of a table $R \in G$, denoted $\mathcal{I}(R)$, is the value of $\mathcal{I}$ on table R.*

For example, consider the triangle graph obtained by restricting TPCE to 3 tables: *Trade*, *Trade_Request*, and *Security* (*T, TR* and *S* for short). The edges are $S.S\_Symb{-}T.T\_S\_Symb$, $S.S\_Symb{-}TR.TR\_S\_Symb$, and $T.T\_ID - TR.TR\_T\_ID$. Let α, β, γ, δ, ε be the entropies of the attributes $S.S\_Symb$, $T.T\_S\_Symb$, $TR.TR\_S\_Symb$, $T.T\_ID$ and $TR.TR\_T\_ID$, resp. Then the entropy transfer matrix is:

$$\begin{array}{c|ccc|}
 & S & T & TR \\
\hline
S & \frac{IC(S)}{IC(S)+2\alpha} & \frac{\alpha}{IC(S)+2\alpha} & \frac{\alpha}{IC(S)+2\alpha} \\
T & \frac{\beta}{IC(T)+\beta+\delta} & \frac{IC(T)}{IC(T)+\beta+\delta} & \frac{\delta}{IC(T)+\beta+\delta} \\
TR & \frac{\gamma}{IC(T)+\gamma+\varepsilon} & \frac{\varepsilon}{IC(T)+\gamma+\varepsilon} & \frac{IC(T)}{IC(T)+\gamma+\varepsilon}
\end{array}$$

It is well known that, for any connected, non-bipartite graph G, and for any probability matrix Π over G, there exists a unique stationary distribution $\mathcal{I}$; see, e.g., [8]. Thus, the importance of a table is well-defined. Vector $\mathcal{I}$ can be computed using classical eigenvector methods, or by the following iterative approach: Start with an arbitrary non-zero vector V_0, and repeatedly compute $V_{i+1} = V_i \times \Pi$ until $dist(V_i, V_{i+1}) \leq \varepsilon$ ($dist$ is usually defined by the L_∞-metric). Setting $\varepsilon = 0$ means that the process stops when the stationary distribution is reached. Although the vector $\mathcal{I}$ does not depend on the initial values V_0, it is useful to start the process with $V_0(R) = IC(R)$ for each table R. This helps our intuitive understanding of information transfer. Moreover, if the values $IC(R)$ and $\mathcal{I}(R)$ are not too different, the number of iterations required to converge is small. In our experiments, we use this iterative approach.

Comparison with previous work The definition of table importance in [11] is equivalent to the stationary distribution of a random walk process. The crucial difference from our approach is that the probability matrix is defined by

$$\Pi[R, S] = af_e(R) / \sum_{e'} af_{e'}(R),$$

where e is the edge between R and S, and the sum ranges over all join edges e' incident to R. (If there are multiple edges between R and S, the numerator is the sum of their respective af values.) As a result, dimension tables like *Status_Type* gain large importance, which is not desirable. To understand why this happens, refer to the example in Figure 1 (d), and assume that it is part of a larger graph, in which S has several other join edges: a typical scenario when S is a dimension table. For simplicity, assume that R has no other edges. When the random walk starts, the information in table R is defined to be $|R|$, and this entire information is transferred to S in the next step, since $\Pi[R, S] = 1$. However, two steps later, R gets less importance back from S, since $\Pi[S, R] < 1$. As the process continues, S becomes a net gainer of importance from R, and possibly from its other neighbors. The more tables S is connected to, the more likely it is to increase its final value.

We note that in [11], the authors allow, in fact, a fraction p of the current information of R to remain inside R: for our example, this means $\Pi[R, R] = p$ and $\Pi[R, S] = 1 - p$. Thus, adjusting the value of p might appear to alleviate the problem. However, this is false: the stationary distribution of matrix Π is the same as that of matrix $pI + (1 - p)\Pi$, for any connected graph (I is the identity matrix). Therefore, parameter p is irrelevant for the final value of table importance, and only influences the number of iterations. Different values of p must be used in each table for the outcome to be altered. However, fine-tuning so many parameters is very difficult.

By contrast, in our model, $\Pi[R, R]$ is different for each R, but depends on the information content of R, rather than on an arbitrary parameter. Moreover, for the example in Figure 1 (d), the total information that R gives to S in the first step is slightly less than $H(R.A) \approx 1.53$, instead of $|R| = 9$. This is consistent with the intuitive perception of 'information transfer' along an edge: one cannot transfer more information via a join attribute than the total information content of that attribute, no matter what the information of the entire table is.

A second important distinction from previous models is that we explicitly take into account the information content of attributes that do not participate in any join edges, other than the self-loop $R.Key - R.Key$. Thus, two tables of equal size and connectivity in the schema graph may have very different information content, if one of the tables contains significantly more attributes than the other. By defining the primary key $R.Key$, and the self loop $R.Key - R.Key$, we insure that the information content of all non-joining attributes of R remains in R, and thus contributes to the final value $\mathcal{I}(R)$.

Alternative Models We discuss alternative models to those in Definition 4, obtained by changing the formula for $Pr(R.A \rightarrow S.B)$ (the definitions for $\Pi[R,S]$ and $\Pi[R,R]$ remain the same). The most intuitive formula would be $Pr(R.A \rightarrow S.B) = \frac{H(R.A)}{IC(R)}$. However, since $R.A$ may participate in many join edges, such a formula could result in $\sum_S \Pi[R,S] > 1$, violating the property of a probability matrix. Therefore, we inforce this property by using the values q_A in the denominator. An alternative formula is

$$\text{(CE)} \quad Pr(R.A \rightarrow S.B) = \frac{(1/q_A)H(R.A)}{IC(R)}.$$

This, however, reflects to a lesser extent the information transfer that occurs via database joins. Intuitively, in Definition 4 we assume that each attribute $R.A$ can transfer its entire information to any attribute it joins with, no matter how many such attributes there are. By contrast, in the above formula, we assume that $R.A$ has a constant amount of information it can transfer, and it divides it among its q_A incident edges. It is often the case that, when $R.A$ is a primary key, it is connected via several fk-pk edges to attributes in other tables. The more edges incident to $R.A$, the less information $R.A$ can transfer on any one of them. On the other hand, a corresponding foreign key $S.B$ connected to $R.A$ usually has no other join edges. Hence, the transfer of entropy between $R.A$ and $S.B$ is heavily skewed in favor of $R.A$, and table R gains too much importance from its neighbors. We discuss such a case in Section 7, for $R = Address$. For clarity, we refer to the entropy transfer matrix in Definition 4 as *variable entropy tranfer* (*VE* for short), and to the matrix defined above as *constant entropy transfer (CE)*.

We also consider a conceptually different approach, as follows. If a join edge between tables R and S requires $R.A = S.B$, let $A_B = \{a_i | \exists b_j \in B, a_i = b_j\}$. Thus, A_B is the subset of values in $R.A$ that have non-empty join in $S.B$. We define

$$H(A_B) = \sum_{a_i \in A_B} p_i \cdot \log(\frac{1}{p_i}),$$

i.e., $H(A_B)$ is the entropy of A_B. Then, there are two alternative definitions for $Prob(R.A \rightarrow S.B)$, as follows:

$$\text{(VJE)} \quad Pr(R.A \rightarrow S.B) = \frac{H(A_B)}{\log|R| + \sum_{R.A'} q_{A'} \cdot H(R.A')},$$

or

$$\text{(CJE)} \quad Pr(R.A \rightarrow S.B) = \frac{(1/q_{A'})H(A_B)}{IC(R)}.$$

We call the first variant *variable joinable entropy transfer (VJE)*, and the second one *constant joinable entropy transfer (CJE)*. The problems that occur for either of these variants are more subtle. We illustrate them via table *Zip_Code*, which ends up with significant importance in the VJE and CJE models. The reason is that table *Zip_Code* contains all zip codes in US, the majority of which do not instantiate the join edge to table *Address* (which contains the addresses of customers and companies in the database). Hence, for $R.A = Zip\_Code.ZC\_Code$ and $S.B = Address.ZC\_Code$, we have $H(A_B) << H(R.A)$, but $H(B_A) = H(S.B)$. Therefore, the VJE matrix transfers much less information from *Zip_Code* to *Address* than the VE matrix, and allows *Zip_Code* to acquire higher importance (similarly for CJE versus CE). If the database required *Zip_Code* to contain only zip codes associated with addresses, we conjecture that its importance would significantly decrease. We have tested this conjecture, and found it to be true; see Table 4. However, *adding tuples that instantiate no joins to a table should not significantly alter its importance.* We have conducted experiments for all four definitions of the entropy transfer matrix, and verified that the VE matrix is indeed the best model.

5. TABLE SIMILARITY

As discussed in the Section 3, our schema summarization approach also requires a similarity measure between tables. Before defining it formally, we must clarify what properties we wish the measure to satisfy. Refer again to the examples in Figure 1.

In the first case, the edge $R.A - S.B$ is a pk-pk join, and all values in $R.A$ match with all values in $S.B$. In the second case, the edge is a fk-pk join, such that all values in $R.A$ have a non-empty join in $S.B$ and viceversa. In the third case, the join is another pk-pk, but only $2/3$ of the values in $R.A$ instantiate the join (and similarly for $S.B$). Most people would agree that the strongest connection between tables R and S occurs in the first case. This is particularly intuitive when comparing the first and third cases. Simply put, in Figure 1 (a), $R.A = S.B$, whereas in Figures 1 (c), $R.A \neq S.B$. We therefore postulate the following (refer to Definition 1).

Property 1: The similarity of tables R and S (with respect to an edge e that connects them) is proportional to the matching fractions $f_e(R)$ and $f_e(S)$.

However, just looking at the fraction of tuples in R that instantiate the join does not explain our intuition that the first case illustrates a stronger connection than the second case. The difference is that in Figure 1 (b), there are significantly more edges between tuples in the two tables. The more edges there are, the farther away the join is from a one-to-one connection.

Property 2: The similarity of tables R and S (with respect to an edge e that connects them) is inverse proportional to the matched average fanouts $maf_e(R)$ and $maf_e(S)$.

We now propose the following definition for the similarity measure between tables, which we call *strength*.

DEFINITION 6. *Let R and S be two tables in the schema graph G. If there exists a join edge e between R and S in the graph, then the strength of R and S with respect to e is*

$$strength_e(R,S) = \frac{f_e(R)f_e(S)}{maf_e(R)maf_e(S)}.$$

Let $\pi : R = R_0 - R_1 - \ldots - R_\alpha = S$ be a path of arbitrary length $\alpha \geq 1$ in G, and let $e_i \in \pi$ be the edge between R_{i-1} and R_i. Define $strength_\pi(R,S) = \Pi_{i=1}^{\alpha} strength_{e_i}(R_{i-1}, R_i)$. Then

$$strength(R,S) = \max_\pi strength_\pi(R,S).$$

Since edge $e = R.Key - R.Key$ is in the schema graph, and by definition $f_e(R) = maf_e(R) = 1$, we have $strength_e(R,R) = 1$. Below, we show that $strength(R,S) \leq 1, \forall S$, so our definition implies that $strength(R,R) = 1 \geq strength(R,S)$, $\forall S \neq R$.

PROPOSITION 1. *(i) For any two tables R and S,*
$0 \leq strength(R,S) \leq 1$.
(ii) For any three tables R, S and T, $strength(R,T) \geq strength(R,S) \cdot strength(S,T)$.

PROOF. The first claim follows from the fact that for any edge e, $f_e(R), f_e(S) \leq 1$ and $maf_e(R), maf_e(S) \geq 1$, as discussed in Section 2. For the second claim, let π_1 be a path for which $strength(R,S) = strength_{\pi_1}(R,S)$, and π_2 be a path for which $strength(S,T) = strength_{\pi_2}(S,T)$. Let $\pi_3 = \pi_1\pi_2$ be the concatenation of the two paths. Then

$$\begin{aligned} strength(R,T) &\geq strength_{\pi_3}(R,T) \\ &= strength(R,S) \cdot strength(S,T) \end{aligned}$$

□

For our clustering method, it is convenient to use a distance measure, rather than a similarity one. We define the distance between tables R and S as

$$dist_s(R,S) = 1 - strength(R,S).$$

PROPOSITION 2. $(\mathcal{R}, dist_s)$ *is a metric space. In particular, $dist_s$ has the triangle inequality.*

PROOF. Let R, S and T be three tables. For any two numbers $0 \leq x, y \leq 1$, it is readily verified that $1 - xy \leq (1-x) + (1-y)$. Substituting $x = strength(R,S)$, $y = strength(S,T)$, and using Proposition 1 (ii), we obtain that $dist(R,T) \leq dist(R,S) + dist(S,T)$. □

For a fixed table R, all the values $strength(R,S)$, $S \neq R$, can be computed in $\tilde{O}(|G|)$ using Dijkstra's shortest paths algorithm as follows. Define the weight of an edge e between tables R and S to be $wt(e) = \log(1/strength_e(R,S)) \geq 0$. For any path π as in Definition 6,

$$wt(\pi) = \sum_{i=1}^{\alpha} wt(e_i) = \log(1/\Pi_{i=1}^{\alpha} strength_{e_i}(R_{i-1}, R_i)).$$

Thus, the shortest-weight path between R and S has weight equal to $\log(1/strength(R,S))$. Let n be the number of tables in the database. Since schema graphs are sparse, $|G| = O(n)$, so we conclude that all n distances from a fixed table R can be computed in $\tilde{O}(n)$. All n^2 distances between tables can be computed in $\tilde{O}(n^2)$.

In our experiments, we also study two alternative similarity measures between tables, as proposed in previous work. The *proximity* measure defined in [9] requires $O(n^3)$ to compute all the n^2 proximity values simultaneously. We are not aware of any faster method to compute fewer proximity values. The *coverage* measure defined in [11] requires, as a first step, the computation of a so-called affinity measure along edges and paths, similar to $strength$ at first glance. However, the affinity along edge $e = R \rightarrow S$ is defined as $aff(R,S) = 1/af_e(R)$, so $\log(1/aff(R,S))$ may be negative, and Dijkstra's method does not apply. Moreover, the affinity along a path π depends on the number of edges in π, so there is no clear correspondence between affinity and shortest paths, even with negative weights. The authors compute affinity and coverage by exploring *all* cycle-free paths between each pair of tables. Since there are combinatorially many distinct paths, this step becomes very expensive for large schema graphs.

6. WEIGHTED K-CENTER

Defining good clustering criteria for a specific problem is often an overlooked step. However, as we show in our experimental study, the quality of the results may vary greatly, depending on this step. For schema summarization, we propose a min-max optimization problem, i.e., we wish to minimize the maximum distance between a cluster center and a table in that cluster. (The other option is to minimize the sum of such distances, which sometimes results in a few significantly different tables being classified together).

```
GREEDYCLUS($G = (\mathcal{R}, \mathcal{E}), k$)
 $\mathcal{C} = \{C_1\}$: current clustering;
1. $center(C_1) = R_1$ s.t. $\mathcal{I}(R_1) = \max_{R \in \mathcal{R}} \mathcal{I}(R)$;
2. $cluster(R) = C_1, \forall R \in \mathcal{R}$: assign all tables to $C_1$;
3. for $i = 2$ to $k$
     /*$\Delta(R) = \mathcal{I}(R) dist(R, center(cluster(R)))$*/
4.   $center(C_i) = R_i$ s.t. $\Delta(R_i) = \max_R \Delta(R)$;
5.   for each $R \in \mathcal{R}$
6.     if ($dist(R, center(cluster(R))) > dist(R, R_i)$)
7.       $cluster(R) = C_i$;
8.   endfor
9.   $\mathcal{C} = \mathcal{C} \cup \{C_i\}$
10. endfor
11. return ($\mathcal{C}$, $cluster(\cdot)$)
```

Figure 3: Greedy Algorithm for Weighted k-Center.

However, using a strictly distance-based approach may result in undesirable artifacts, such as grouping the top-2 most important tables in the same cluster C. Since only one table can serve as $center(C)$, the other one is excluded from the summary. To achieve a tradeoff between importance and distance, we propose using Weighted k-Center clustering [5], where the weights are the table importance values. More precisely, we want to minimize the following measure for a set of k clusters $\mathcal{C} = \{C_1, \ldots, C_k\}$:

$$\mu(\mathcal{C}) = max_{i=1}^{k} \max_{R \in C_i} \mathcal{I}(R) dist(R, center(C_i)).$$

Weighted k-Center is NP-Hard, but admits fast approximation algorithms. The most popular one is the greedy approach described in Figure 3. It starts by creating one cluster and assigns all tables to it. It then iteratively chooses the table R_i whose weighted distance from its cluster center is largest, and creates a new cluster with R_i as its center. All tables that are closer to R_i than to their current cluster center are reassigned to cluster C_i.

THEOREM 1. *For any schema graph G, algorithm* GREEDYCLUS *of Figure 3 computes an α-approximate Weighted k-Center in $\tilde{O}(k|G|)$ time, where*

$$\alpha = \min\{3, \max_R \mathcal{I}(R) / \min_R \mathcal{I}(R)\}.$$

PROOF. The approximation factor for the greedy algorithm was proved in [5]. The running time follows from the observation that the double loop in steps 3–9 requires kn distance computations, where $n = |\mathcal{R}|$ is the total number of tables. More precisely, for each new cluster center R_i, we must compute $dist(R_i, R), \forall R$. By the observation in Section 5, for a fixed R_i, all n distances can be computed in $\tilde{O}(|G|)$, and the result follows. □

7. EXPERIMENTAL EVALUATION

We have conducted an extensive experimental study to compare our approach with the method of [11], as well as with several hybrid methods that are described in this section. We first validate each of the three components of our method: the model for table importance we defined in Section 4, our novel distance function $dist_s$ (Section 5), and our conjecture that Weighted k-Center is an appropriate clustering for schema summarization. We study the properties of each component, and compare it with alternatives, as follows:

- *Alternative table importance:* We study the definition of table importance proposed in [11]. Since the importance is initialized as table cardinality, we denote this approach by I_C. By contrast, the model we introduced in Definitions 4 and 5 is based on entropy, so we use the notation I_E to denote it.

- *Alternative distance functions:* We study two alternative distance functions, based on previously proposed similarity measures. The first function is

  $$dist_c = 1 - coverage_N,$$

  where $coverage_N$ is the normalized $coverage$ from [11]. More precisely,

  $$coverage_N(R,S) = \min\{coverage(R,S)/|S|, 1\}.$$

  (We also studied the distance $1/coverage(R,S)$, but concluded it was not a good alternative, so for the sake of clarity, we exclude it from the results). In addition, we study the distance function

  $$dist_p = 1 - proximity,$$

  where $proximity$ is the measure proposed in [9] to quantify the similarity of two nodes in a directed graph with specified edge weights. The sum of weights over all edges out of a node is 1. In our case, the graph is obtained from the schema graph by replacing each undirected edge $e = R - S$ with two directed edges, $R \to S$ and $S \to R$. The weights are defined as $wt[R \to S] = af_e(R)/\sum_{e'} af_{e'}(R)$ and $wt[S \to R] = af_e(S)/\sum_{e'} af_{e'}(S)$, where the first sum ranges over all (undirected) edges e' incident to R, and the second sum ranges over all (undirected) edges e' incident to S. We implemented the FastAllDAP algorithm (Table 4 in [9]) for computing the proximity between all pairs of tables in the directed graph thus obtained.

- *Alternative clustering algorithm:* The method of [11] proposes a Balanced-Summary algorithm for schema clustering, which uses a min-sum criterion for clustering. We implemented this algorithm and compared it with Weighted k-Center.

Table 1 summarizes the three dimensions along which we conduct our evaluation. We first compare the choices for importance and distance independently, using the pre-defined categories of the TPCE benchmark to define objective accuracy measures. We then study various summarization methods obtained by choosing one entry in each column. Our novel algorithm, which we propose as the method of choice, is the one that uses the first entry in each column, i.e., *Table Importance* = I_E, *Distance* = $dist_s$, and *Clustering* = Weighted k-Center. The current state-of-the-art [11] corresponds to the second entry in each column, i.e., *Table Importance* = I_C, *Clustering* = Balanced-Summary, but uses $coverage$ as a similarity measure, rather than distance $dist_c$. Finally, we consider several hybrid methods, such as *Table Importance* = I_E, *Distance* = $dist_c$, and *Clustering* = Weighted k-Center.

7.1 Experimental Setup

We conduct our study over the TPCE schema. This benchmark database is provided by TPC [10] to simulate the On-Line Transaction Processing (OLTP) workload of a brokerage firm. As mentioned before, it consists of 33 tables. However, since no active transactions are considered, table *Trade_Request* is empty. Therefore our experiments are performed only on the remaining 32 tables. Recall that the tables are pre-classified in four categories:

Table Importance	Distance	Clustering
I_E	$dist_s$	Weighted k-Center
	$dist_c$	
I_C	$dist_p$	Balanced-Summary

Table 1: Dimensions of experiments.

Parameters	TPCE-1	TPCE-2
Number of Customers	1,000	5,000
Initial Trade Days	10	10
Scale Factor	1,000	36,000

Table 2: Parameters of EGen for TPCE.

Broker, Customer, Market and *Dimension*. Category *Dimension* consists of dimension tables (hence the name) which share no join edges among themselves, except for one edge between *Address* and *Zip_Code*. For the purpose of evaluating the quality of schema summaries, we consider these dimension tables to be outliers. Thus, we are interested in discovering only the other three categories.

We use EGen[1] to generate two significantly different database instances for the schema, which we call TPCE-1 and TPCE-2. The different parameter settings for EGen are shown in Table 2. The purpose of generating two instances is to study the sensitivity of various summarization methods to significant changes at the data level. Note that the schema graph remains the same, so ideally, the summaries for TPCE-1 and TPCE-2 should be consistent with each other, and with the pre-defined categories. The main differences between the two instances are as follows.

In TPCE-2, the size of table *Customer* increases by a factor of 5. The change of the scale factor, from 1,000 to 36,000, affects the size of the majority of other tables, and therefore the values $Prob(R.A \to S.B)$ and $strength(R,S)$ for most pairs. In addition, the af and maf values change for 24 out of 86 edges. In the following sections, whenever not explicitly stated, the experimental results are for TPCE-1. All methods are implemented in Java, and evaluated on a PC with 2.33GHz Core2 Duo CPU and 3.25G RAM.

7.2 I_E Models

Before comparing the two table importance models, I_E and I_C, we first conduct a study of different definitions for I_E. Recall that in Section 4 we discussed three alternative models, *CE*, *VJE* and *CJE*, for defining the entropy transfer matrix, in addition to the *VE* model of Definition 4. Based on the semantics of each model, we conjectured that each of them has shortcomings when compared to *VE*. We now validate this intuition by a comparison study.

Table 3 shows the top-5 important tables according to each transition matrix. With the exception of *VE*, the other methods all rank at least one dimension table among the top 5. Thus, *CE* ranks *Address* as the 4th most important. In Section 5, we explained our intuition that constant entropy models lead to unbalanced transfers of importance on fk-pk edges, with the table containing the primary key being a disproportionate gainer from its neighbors. This is clearly the case for *Address*, which is the primary key table for three fk-pk edges (to *Customer*, *Company*, and *Exchange*).

As for the *VJE* and *CJE* models, they both rank *Zip_Code* as the most important table. In Section 5, we gave an intuitive explanation for why this table absorbs large amounts of entropy from its

[1]EGen is a software package provided by TPC [10].

Rank	*VE*	*CE*	*VJE*	*CJE*
1	*Trade*	*Customer*	*Zip_Code*	*Zip_Code*
2	*Security*	*Security*	*Trade*	*Address*
3	*Customer*	*Company*	*Customer*	*Customer*
4	*Financial*	*Address*	*Security*	*Company*
5	*Holding*	*Financial*	*Address*	*Security*

Table 3: Effects of entropy transfer matrix on table importance I_E.

Rank	*VE*	I_E	*VJE*	I_E
1	*Trade*	57.498	*Trade*	62.323
2	*Security*	41.191	*Customer*	44.826
3	*Customer*	36.015	*Security*	34.707
4	*Financial*	30.489	*Address*	34.307
5	*Holding*	28.717	*Financial*	26.446

Table 4: Table importance I_E after removing non-joinable tuples from *Zip_Code*.

neighbors under these models. Our reasoning was that matrices based on joinable entropy transfer allow tuples that instantiate no joins to have an inordinate influence on the final result. In particular, by varying the number of non-joining tuples in *Zip_Code*, we can greatly affect the value $H(A_B)$ from *Zip_Code* to *Address* (i.e., $R.A = Zip\_Code.ZC\_Code$ and $S.B = Address.ZC\_Code$), while the value in the opposite direction remains the same.

To test the extent to which such a local modification of $H(A_B)$ can influence the overall results, we conducted the following experiment. We removed from *Zip_Code* all the tuples that did not instantiate the edge to *Address*. This reduced the size of *Zip_Code* by about 90%. No other modifications were made to the database. We then computed the table importance, under *VE* and *VJE*, for the modified data. The results are shown in Table 4. The top-5 tables for *VE* remain the same and there are negligible changes in their I_E values. By contrast, the changes for *VJE* are significant. The importance of *Zip_Code* decreases from 127 to 16, and its rank changes from 1st to 12th. We conclude that *VJE* is highly unstable under local modifications in the data, but that *VE* is consistent. Furthermore, computing the matrix for *VJE* and *CJE* is very time-consuming, as it requires executing all data joins to calculate $H(A_B)$. Therefore, for the remainder of this section, the entropy-based importance I_E is computed using the *VE* matrix.

7.3 Table Importance

We now compare the I_E and I_C models for table importance. While both models use stable state values in a random walk over the schema graph, the respective transition matrices are conceptually quite different. For I_E, the matrix depends on the entropy of the join columns, while for I_C, it depends on the average fanouts af.

Tables 5 and 6 show the top-5 important TPCE tables obtained in each model. Recall that we compute the importance in I_E, resp. I_C, via an iterative process which starts by initializing the importance of each table to its information content, resp. its cardinality. Hence, for each of the top-5 tables, we show the values of their initial importance, their final importance, and their corresponding rank in the competing model. Although the initial importance does not influence the final value, but only the number of iterations, it is still instructive to compare the two sets of values. For example, in Table 5, *Trade* and *Security* have smaller initial values than both *Customer* and *Financial*, but end up being more important, due to their connectivity in the graph. A similar observation holds for *Trade* and *Trade_History* in Table 6. Hence, table connectivity makes a difference in both models. However, the way it influences the result in each case is quite different.

Rank	Table	Info. Content	I_E	I_C Rank
1	*Trade*	39.730	57.798	1
2	*Security*	37.350	41.405	4
3	*Customer*	45.781	36.202	17
4	*Financial*	43.575	30.647	16
5	*Holding*	26.112	28.866	11

Table 5: Top-5 important tables based on I_E.

Rank	Table	Card.	I_C	I_E Rank
1	*Trade*	576000	1805787.6	1
2	*Trade_History*	1382621	659751.7	14
3	*Status_Type*	5	503280.9	32
4	*Security*	685	487461.5	2
5	*Holding_History*	722143	321415.2	9

Table 6: Top-5 important tables based on I_C.

Notice that only 2 out of the top-5 tables are the same in Tables 5 and 6: *Trade* and *Security*. In general, I_C favors large tables, which is an expected consequence of the model. However, tables *Trade_History* and *Holding_History*, which have the highest, resp. 3rd highest, cardinality in TPCE-1, and are among the top-5 in I_C, contain only historical data and are of limited importance to the database. By contrast, I_E ranks *Trade_History* as the 14th in importance, and *Holding_History* as the 9th. We noted before that such examples of'‘history’ tables are quite typical in warehouse databases, so the ability to distinguish between the size and overall significance of a table is clearly a desirable property for a model.

On the other hand, table *Status_Type* is the smallest one in the database, yet still ends up in the top-5 in I_C. This is due to its high connectivity - 6 join edges - as well as the extremely large af values on those edges, which make it a net gainer of importance from all its neighbors. Again, this is a direct consequence of the I_C model, as discussed in Section 4. By contrast, I_E ranks this table as the least important in the schema, which is what most humans would also probably do.

Finally, we note that while both models rank in the top 5 at least one table from each of the three main categories (i.e.., *Broker*, *Customer*, and *Market*), category *Customer* is better represented in I_E (by *Customer* and *Holding*) than in I_C (by *Holding_History*). Although being among the top-5 does not automatically make a table part of the summary, it is nevertheless more likely that a summarization method using I_C will choose table *Holding_History* to summarize this category (as, indeed, is the case in our experiments).

A different type of comparison between I_C and I_E is depicted in Table 7, which shows the top-7 tables computed over TPCE-1 and TPCE-2. As discussed above, while the schema remains the same for both databases, the entries in the probability matrices change, since they depend on the data. We achieve similar results on both TPCE-1 and TPCE-2 for I_E. Although TPCE-1 differs from TPCE-2 in both table sizes and join selectivity, our entropy-based approach is quite stable in the way it ranks important tables. By contrast, I_C is more sensitive to data variations. Among its top tables, not only the ranks change, but also two out of seven tables are different.

These experiments demonstrate that our entropy-based model

Rank	I_E/TPCE-1	I_E/TPCE-2
1	*Trade*	*Trade*
2	*Security*	*Security*
3	*Customer*	*Customer*
4	*Financial*	*Financial*
5	*Holding*	*Company*
6	*Company*	*Customer_Account*
7	*Customer_Account*	*Holding*

(a) I_E

Rank	I_C/TPCE-1	I_C/TPCE-2
1	*Trade*	*Security*
2	*Trade_History*	*Daily_Market*
3	*Status_Type*	*Watch_Item*
4	*Security*	*Watch_List*
5	*Holding_History*	*Trade*
6	*Daily_Market*	*Trade_History*
7	*Customer_Account*	*Customer_Account*

(b) I_C

Table 7: Comparison of Top-7 Important Tables for I_E and I_C.

	All tables		No *Dimension* tables	
	TPCE-1	TPCE-2	TPCE-1	TPCE-2
$dist_s$	0.659	0.649	0.72	0.702
$dist_c$	0.589	0.621	0.619	0.662
$dist_p$	0.5	0.557	0.529	0.601

Table 8: Accuracy of distance functions.

outperforms the previously proposed approach in both accuracy and consistency.

7.4 Distance Between Tables

In this section, we study the properties of the metric distance $dist_s = 1 - strength$, and compare it with the distances $dist_c = 1 - coverage_N$ and $dist_p = 1 - proximity$.

We first examine the accuracy with which each of the three distances reflects table similarity. To define an objective measure for this comparison, we make the following observation. Since TPCE is pre-classified into four categories, it is reasonable to assume that tables within one category are more similar to each other than tables in different categories (except for *Dimension* tables). Therefore, for each table R, its distances to tables within the same category should be smaller than its distances to tables in different categories. For any distance function d and some fixed value q, let $m(R)$ denote the number of tables in the same category as R among its top-q nearest neighbors under distance d. Let $n(R)$ denote the number of top-q nearest neighbors of R (in general, $n(R) = q$, but if there are ties, we include all relevant tables). We define the accuracy of distance d as

$$acc(d) = \frac{\sum_R \frac{m(R)}{n(R)}}{n},$$

where n is the number of database tables.

Table 8 reports the accuracy of the three distance functions over TPCE-1 and TPCE-2, computed for top-5 nearest neighbors. We report both the average over all 32 tables (first two columns), as well as the average over the 28 tables that are not in the *Dimension* category. In both cases, the distance $dist_s$ achieves the highest accuracy, on both databases. The standard deviation of the values $m(R)/n(R)$ is about 0.2 for all distance functions.

Figure 4 illustrates how the top-5 nearest neighbors of table *Customer* change for each distance function. For $dist_s$, 3 of the nearest neighbors (tables *Customer_Account, Customer_Taxrate*, and *Watch_List*) are in the *Customer* category. On the other hand, the nearest neighbors found based on $dist_c$ and $dist_p$ have only 2 tables from the *Customer* category. Moreover, $dist_c$ finds 3 of the top-5 nearest neighbors from the *Dimension* category.

Additional considerations: By the definition of $coverage$ in [11], it is possible to have $coverage(R, S) > coverage(R, R)$ for some $S \neq R$. Similarly, it is possible to have $coverage(R, S)/|S| > coverage_N(R, R) = 1$ (thus, the definition $coverage_N(R, S) = \min\{coverage(R, S)/|S|, 1\}$). This undesirable artifact, which runs counterintuitive to the notion of a similarity measure, is a consequence of the formula for $affinity$ - a component of $coverage$. From the discussion in [11], the $affinity$ between any pair of tables is intended to be at most 1. However, by definition, the $affinity$ between R and S is a maximum value over all paths connecting R and S; and the value along a specific path π from R to S contains factors of the type $1/af_e$, for edges $e \in \pi$. As we saw in Section 2, it is possible that $af_e < 1$ for some low-selectivity edges e. Thus, if tables R and S are connected by a path with low-selectivity edges, their affinity is boosted to a value larger than 1. This situation does, in fact, occur in TPCE:

$$affinity(Zip\_Code, Customer) > 1,$$

and

$$coverage(Zip\_Code, Customer) > coverage(Zip\_Code, Zip\_Code).$$

An important property of $dist_s$ is that it is a metric. Thus, it is symmetric and it satisfies the triangle inequality. Neither of these properties is satisfied by $dist_c$, nor by $dist_p$, so no approximation guarantees are likely for any clustering algorithm that uses them.

Time complexity: As discussed in Section 5, all $strength$ values can be computed in $\tilde{O}(n^2)$ using Dijkstra's algorithm, while $proximity$ requires $O(n^3)$, and $coverage$ may explore combinatorially many paths in the schema graph. In our experiments, the average time for computing the similarity between a pair of tables was $\Theta(10^{-2})$ ms for $strength$ and $proximity$, and $\Theta(10^2)$ ms for $coverage$. Hence, while all these computations are fast, the running time for computing $coverage$ is nevertheless four orders of magnitude higher, which indicates that it is unlikely to scale well for large schema graphs (many hundreds of tables) in real database systems.

7.5 Clustering Algorithms

In this section, we compare the two clustering methods, Balanced-Summary and Weighted k-Center, by fixing the other two dimensions. More precisely, we use I_C and $coverage$ for Balanced-Summary, and I_C and $dist_c$ for Weighted k-Center. To evaluate the quality of the summary, we propose a similar approach to the one we used for measuring the accuracy of distance functions. In this case, we consider that for a cluster C_i, the table $center(C_i)$ chosen as the representative of the cluster(by either Weighted k-Center or Balanced-Summary) determines the category for the entire cluster. Thus, if $center(C_i)$ belongs to, e.g., category *Market*, then all the tables in C_i are categorized as *Market* in the summary. We then measure how many tables are correctly categorized, as follows. Let $m(C_i)$ denote the number of tables in C_i that belong to the same category as $center(C_i)$, in the pre-defined labeling (including $center(C_i)$). Then the accuracy of a summary

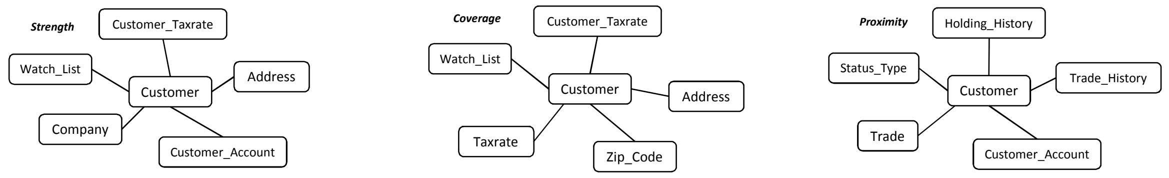

Figure 4: Nearest neighbors for table *Customer* under the three distance functions.

$\mathcal{C} = \{C_1, \ldots, C_k\}$ is

$$acc(\mathcal{C}) = \frac{\sum_{i=1}^{k} m(C_i)}{n},$$

where n is the total number of database tables. We also define the accuracy of each cluster as $acc(C_i) = m(C_i)/n(C_i)$, where $n(C_i)$ is the total number of tables in cluster C_i. This allows a more detailed view of where the inaccuracies occur.

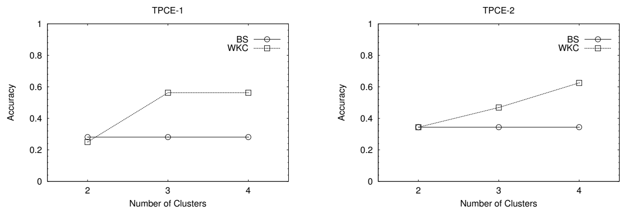

Figure 5: Comparison of Balanced-Summary and Weighted k-Center.

Balanced-Summary: We implemented the Balanced-Summary algorithm described in Figure 7 in [11], which is shown to be better than several alternatives. It chooses cluster centers in decreasing order of table importance I_C, provided that a new center is not "dominated" by existing centers. The dominance relationship is defined in terms of $coverage$, and reflects whether the clustering measure (which is a sum of $coverage$ values) would increase or decrease if a cluster center was replaced by a different table. Moreover, if a newly chosen cluster center dominates an existing center, the old center is deleted from the summary. This approach tries to balance the role of table importance and table coverage for a min-sum clustering measure, roughly analogous to what Weighted k-Center does for the min-max clustering. The algorithm stops when it picks k centers, or sooner, if all tables are dominated by existing centers. After all centers are chosen, each remaining table is assigned to the cluster whose center best covers it. Although we can set the desired number of clusters k in the algorithm, it may return fewer clusters due to dominance. This is the problem we encountered in our experiments. For TPCE-1, the method returns 2 clusters, no matter the value $k \geq 2$. The first cluster contains only table *Trade*, which has the highest I_C; and the second cluster contains all the remaining tables, with *Trade_History* as its center (table *Trade_History* dominates all other tables except *Trade*). For TPCE-2, the method only generated one cluster with center *Daily_Market*, because this table has the highest I_C in this case, and it dominates all the other tables. Clearly, the Balanced-Summary approach does not work well on TPCE.

Figure 5 plots the accuracy for the method above, as well as the alternative Weighted k-Center. Clearly, Weighted k-Center performs better. For a full comparison of accuracy, see also the graphs in Figure 6.

7.6 Summarization Algorithms

Given the results in the previous subsection, we limit our comparative study of summarization algorithms to methods based on Weighted k-Center. This includes our novel approach, as well as several hybrid methods. We report detailed results for I_E and the three distance functions from Table 1. The accuracy of methods using I_C is smaller (refer to Figure 5 for one such method). Since the distances $dist_c$ and $dist_p$ are asymmetric, we need to specify which direction we use in lines 4 and 6 of the algorithm from Figure 3. Nodes can be clustered to a center by minimizing either $dist(center(C_i), R)$ or $dist(R, center(C_i))$, which may lead to very different results. We first compare the clustering results for $dist_c$ and $dist_p$ based on the two directions for the distance function. The results are shown in Table 9. For $dist = dist_c$, although the clustering using direction $dist(R, center(C_i))$ is more balanced than the other one (in terms of cluster sizes), it picks two centers from category *Market* and none from *Customer*; while the clustering based on the direction $dist(center(C_i), R)$ picks one center from each of the 3 main categories, and even obtains a cluster with $acc(Financial) = 1$. For $dist = dist_p$, the same cluster centers are chosen for each direction. However, the clustering using $dist(R, center(C_i))$ is more balanced than the other one. Therefore, in the following experiments, we use the direction $dist(center(C_i), R)$ for $dist = dist_c$, and the direction $dist(R, center(C_i))$ for $dist = dist_p$. Recall that $dist_s$ is symmetric, so both directions are identical.

Table 10 shows the clustering results based on the three distance functions. For $k = 2$ clusters, there is not much difference among the three distance functions, although the summary for $dist_s$ is slightly more balanced, as well as more accurate. For $k = 3$ clusters, the summary for $dist_s$ is the most balanced and has the highest accuracy, while the summaries for $dist_c$ and for $dist_p$ each contain one big cluster consisting of more than 50% of all tables. We now analyze the results for $k = 4$ clusters. Notice that in this case the summary for $dist_s$ becomes unbalanced for the first time, with one cluster, centered at *Financial*, containing only one table. This trend continued for $k = 5$, which we do not show due to space constraints. Thus, the summary computed via $dist = dist_s$ gives a clear signal that there are only 3 categories in the data, and that computing k clusterings for $k \geq 4$ is meaningless. By contrast, the 4-clusterings computed for $dist_c$ and $dist_p$ are now more balanced, because they each split the biggest cluster of the previous 3-clustering into two smaller clusters. Moreover, their accuracy also goes up, as this split separates tables that belong to different categories. However, this improved overall quality comes at the cost of splitting one natural cateogry. More precisely, the pre-defined category *Market* is represented by two smaller clusters in the summary, one centered at *Security* and the other at *Financial*. We conclude that the clustering for $dist = dist_s$ is the most consistent with the predefined classification of TPCE, and the only one capable of automatically discovering the correct number of categories, i.e., $k = 3$.

The above results are from TPCE-1. Due to space limitations,

C_i	$dist(center(C_i), R)$				$dist(R, center(C_i))$			
	center(C_i)	$n(C_i)$	$m(C_i)$	$acc(C_i)$	center(C_i)	$n(C_i)$	$m(C_i)$	$acc(C_i)$
1	*Trade*	19	8	0.42	*Trade*	11	7	0.64
2	*Financial*	7	7	1.0	*Security*	8	4	0.5
3	*Customer*	6	3	0.5	*Financial*	13	7	0.54

(a) $dist = dist_c$

C_i	$dist(center(C_i), R)$				$dist(R, center(C_i))$			
	center(C_i)	$n(C_i)$	$m(C_i)$	$acc(C_i)$	center(C_i)	$n(C_i)$	$m(C_i)$	$acc(C_i)$
1	*Trade*	2	2	1.0	*Trade*	21	8	0.38
2	*Security*	1	1	1.0	*Security*	6	4	0.67
3	*Customer*	29	9	0.31	*Customer*	5	2	0.4

(b) $dist = dist_p$

Table 9: The effect of directional distance on clustering.

k	C_i	$dist_s$				$dist_c$				$dist_p$			
		center(C_i)	$n(C_i)$	$m(C_i)$	$acc(C_i)$	center(C_i)	$n(C_i)$	$m(C_i)$	$acc(C_i)$	center(C_i)	$n(C_i)$	$m(C_i)$	$acc(C_i)$
2	1	*Trade*	9	6	0.67	*Trade*	25	8	0.32	*Trade*	26	8	0.31
	2	*Security*	23	11	0.48	*Financial*	7	7	1.0	*Security*	6	4	0.67
3	1	*Trade*	9	6	0.67	*Trade*	19	8	0.42	*Trade*	21	8	0.38
	2	*Security*	13	11	0.85	*Financial*	7	7	1.0	*Security*	6	4	0.67
	3	*Customer*	10	6	0.6	*Customer*	6	3	0.5	*Customer*	5	2	0.4
4	1	*Trade*	9	6	0.67	*Trade*	13	7	0.54	*Trade*	14	8	0.57
	2	*Security*	12	10	0.83	*Financial*	7	7	1.0	*Security*	6	4	0.67
	3	*Customer*	10	6	0.6	*Customer*	6	3	0.5	*Customer*	5	2	0.4
	4	*Financial*	1	1	1.0	*Security*	4	6	0.67	*Financial*	7	7	1.0

Table 10: Comparison of Weighted k-Center clustering over the three distance functions.

we do not show detailed results for TPCE-2: they follow a similar trend. However, in Figure 6, we plot the values $acc(\mathcal{C})$ for the different clusterings $\mathcal{C}$, over both TPCE-1 and TPCE-2. The graph clearly shows that the most accurate summaries are obtained for the distance $dist_s$, on both TPCE-1 and TPCE-2. Although the differences among the three choices decrease on TPCE-2, $dist_s$ still outperforms the others.

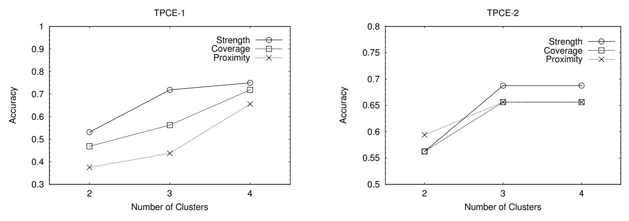

Figure 6: Summary accuracy for Weighted k-Center with I_E.

We have also performed experiments using I_C instead of I_E, and compared the results of Weighted k-Center for the three distance functions. The resulting summaries all had low acc values, and unbalanced clusters. Due to lack of space, we omit those graphs.

8. CONCLUSIONS

In this paper, we proposed a novel approach for summarizing relational schemas, justified by limitations of previous methods. We have defined a new model for table importance using well-known principles from information theory and statistics. One of our main contributions is the definition of a metric distance over schema tables, which allows us to develop a summarization algorithm with provable guarantees, and may prove of independent interest. Finally, we have conducted an extensive study on an independent benchmark schema, showing that our approach is accurate and robust under changes in the data.

9. REFERENCES

[1] B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nahke, and S. Sudarshan. Banks: Browsing and keyword searching in relational databases. In *VLDB*, 2002.

[2] J. Akoka and I. Comyn-Wattiau. Entity-relationship and object-oriented model automatic clustering. *Data Knowl. Eng.*, 1996.

[3] S. Amer-Yahia, L. Lakshmanan, and S. Pandit. Flexpath: flexible structure and full-text querying for xml. *SIGMOD*, 2004.

[4] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley& Sons, 1991.

[5] M. E. Dyer and A. M. Fireze. A simple heuristic for the p-center problem. *Op. Res. Letters*, 3(6):285–288, 1985.

[6] H. Jagadish, A. Chapman, A. Elkiss, M. Jayapandian, Y. Li, A. Nandi, and C. Yu. Making database systems usable. *SIGMOD*, 2007.

[7] M. Jayapandian and H. V. Jagadish. Automated creation of a forms-based database query interface. *PVLDB*, 1(1):695–709, 2008.

[8] R. Motwani and P. Raghavan. *Randomized algorithms*. Cambridge Univ. Press, 1995.

[9] H. Tong, C. Faloutsos, and Y. Koren. Fast direction-aware proximity for graph mining. In *KDD*, pages 747–756. ACM, 2007.

[10] TPCE. http://www.tpc.org/tpce/tpc-e.asp.

[11] C. Yu and H. V. Jagadish. Schema summarization. In *VLDB*, pages 319–330. VLDB Endowment, 2006.

[12] C. Yu and H. V. Jagadish. Querying complex structured databases. *VLDB*, 2007.