

Summary Graphs for Relational Database Schemas

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

Increasingly complex databases need ever more sophisticated tools to help users understand their schemas and interact with the data. Existing tools fall short of either providing the “big picture,” or of presenting useful connectivity information.

In this paper we define summary graphs, a novel approach for summarizing schemas. Given a set of user-specified query tables, the summary graph automatically computes the most relevant tables and joins *for that query set*. The output preserves the most informative join paths between the query tables, while meeting size constraints. In the process, we define a novel information-theoretic measure over join edges. Unlike most subgraph extraction work, we allow metaedges (i.e., edges in the transitive closure) to help reduce output complexity. We prove that the problem is NP-Hard, and solve it as an integer program. Our extensive experimental study shows that our method returns high-quality summaries under independent quality measures.

1. INTRODUCTION

Given the current trend of increasingly large enterprise systems, with explosive growth in both the data size and the complexity of the underlying schemas, database management systems provide ever more advanced tools for helping users understand and interact with the data. Current approaches include custom-made forms and applications [6, 12], database browsers [3], and clustering of tables based on their information content [20, 21, 22].

Forms and query applications provide significant assistance in accessing the underlying data, but the user must be already familiar with the schema; or else, they restrict the queries to a pre-computed set of typical tasks. Database browsers are most useful for databases with partial or missing schema information: they provide tools for discovering primary and foreign keys [3, 14, 23]. However, such detailed information can be overwhelming in an unfamiliar, complex schema.

The recent work in [20, 21, 22] focuses on clustering database tables based on semantic similarity, and presenting cluster representatives to the user. This results in small, easy to understand summaries that help new users get an overall picture of the database.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Articles from this volume were invited to present their results at The 37th International Conference on Very Large Data Bases, August 29th - September 3rd 2011, Seattle, Washington. *Proceedings of the VLDB Endowment*, Vol. 4, No. 11
Copyright 2011 VLDB Endowment 2150-8097/11/08... \$ 10.00.

However, essential information, such as the most significant foreign/primary key constraints, is not present in the summary. In addition, users who have prior experience with the database may be interested in tables that may not appear in the summary.

Our goal We propose a novel summarization technique that addresses the shortcomings of previous approaches. Our method automatically computes an *adaptive schema summary*, which includes the most relevant tables and their connections with respect to a user-specified set of query tables. We call this graph the optimal *summary graph* of the database with respect to the query set. In the following, we discuss the main challenges we must meet in order to compute succinct and informative summary graphs.

1.1 Informative Join Paths

One of the most frequent scenarios we face in querying enterprise systems is as follows: “What are good join paths connecting the tuples in tables R , S and T ?”

Consider, for example, the benchmark TPC-E schema, which simulates the On Line Transaction Processing (OLTP) workload of a brokerage firm. It contains 32 tables, grouped into 3 main categories: *Customer*, *Market* and *Broker*. The database contains information about financial transactions: customer accounts and their respective holdings, types of traded securities, information on traded companies, broker fees, etc.¹

Let tables R, S, T be *Customer*, *Security* and *Trade*. There are combinatorially many join paths connecting each pair. Figure 1(a) shows the union of three possible paths, one for each pair of tables. E.g., the join path *Customer–Status_Type–Security* returns, for a given customer id, all the securities that have the same status (*active*, *pending* etc). Clearly, an analyst is more likely to be interested in the securities *held* by a customer than in those that share the same status. Implicitly, the user wants the join path with *the most relevant information*. For this example, a much better join path is *Customer–Customer_Account–Holding_Summary–Security*, i.e., the path that returns the securities held by a customer.

Therefore, we need a measure of how informative each schema edge is. None of the measures studied in prior work is particularly appropriate for this task, as we discuss in Section 1.3. One of the contributions of this paper is the definition of an information-theoretic *weight function* over schema edges: smaller weights correspond to more informative edges.

1.2 Succinct and Informative Summaries

What constitutes a “good” summary? An obvious answer is size: the smaller the summary, the better. However, this cannot come at the expense of utility. Compare the potential summaries in Fig-

¹Figure 14 in the Appendix shows the complete details of the TPC-E schema.

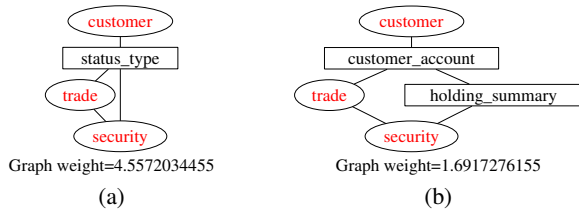


Figure 1: Possible output graphs for TPC-E. Query tables are shown as oval nodes. (a) graph of size 4; (b) more informative graph of size 5.

ures 1(a) and (b). The graph in Figure 1(a) is smaller, but it contains join paths with little information content, as discussed above. The graph in Figure 1(b) is clearly more informative. In fact, it consists of the union of shortest paths between the query tables (using the weight function mentioned above).

However, returning the union of shortest paths between query nodes is not the answer, either, as it may lead to large graphs. Figure 2(a) shows such a case: Alice must generate a report about customer costs associated with trading different types of stocks. She is unfamiliar with the TPC-E schema, but based on their names, she specifies the 4 query tables drawn as oval nodes. The union of shortest paths among these 4 tables contains 13 nodes (out of a total of 32) - which is not much help to Alice. Instead, our approach is to start from such a union of shortest paths and reduce its complexity to a user-specified level. More precisely, we restrict the number of non-query tables in the summary graph - denoted *budget tables* - to be at most a summary budget B (B is user-defined). This, of course, means that we have to drop some of the tables from the output, and replace some paths by metaedges (i.e., edges in the transitive closure of the schema graph). We define and solve an optimization problem over the set of possible summary graphs.

Figure 2(b) shows the output of our algorithm for $B = 2$. Note that the budget tables we include are “hubs” that belong to multiple join paths. As we show in Section 3, this is a consequence of the way we define our optimization problem. Alice can now deduce that, e.g., the most informative connection between tables *Broker* and *Customer_Taxrate* is via table *Customer_Account*. If necessary, she can drill down and request more nodes on this path from the original graph. This is possible because we store the underlying path of each metaedge offline. We also store all edges adjacent to summary nodes (along the shortest paths), so Alice can drill down starting from a summary node, as well. This increases the utility of the summary, while keeping it uncluttered.

By contrast, the graph in Figure 2(c) loses this hub routing information. The graph was obtained by randomly choosing 2 budget tables from among the 9 non-query nodes in Figure 2(a). For query nodes *Broker* and *Customer_Taxrate*, the summary provides no connecting information. The fact that the graph in Figure 2(b) is more informative than the one in Figure 2(c) is also reflected by their total weights: the former has only 73% of the latter’s weight.

1.3 Prior Work

Weight functions: Early work on defining edge weights for data graphs focused on instance-level graphs, where the nodes are either tuples or values: [17, 9] study graphs with homogenous edge semantics (e.g., co-authorship of papers); [13] proposes heuristic weights for RDF graphs; in the keyword search literature, edge weights are typically 1. However, graphs defined over database tuples are inherently heterogenous (edges represent different kinds of joins). It is neither trivial, nor particularly scalable, to define mean-

ingful edge weights for tuples, then aggregate them in a principled way at table level.

More recently, two table-similarity measures have been proposed in [22, 21], for clustering schema tables. The distance used by [22] is asymmetric, and can have negative values (Section 7.4 in [21]). We used the measure in [21] in our experiments, and concluded that it was less accurate and consistent than our proposed approach.

Subgraphs: A large body of work focuses on extracting “good” subgraphs that connect query nodes: In keyword search [5, 11], the goal is to compute trees connecting keyword occurrences. Connection subgraphs [4] connect two query nodes. The work on center-piece subgraphs [17, 16], and on entity-relationship subgraphs [8] is closest to our goal: for a given set of query nodes and a budget constraint, output a connecting subgraph that meets the budget and maximizes some ‘goodness’ measure. However, since the output cannot contain metaedges, this means that shortest paths are not guaranteed to be preserved (although most results try to minimize path lengths by incorporating them in the definition of the goodness measure).

Related problems are studied in graph theory, e.g., minimum spanning trees, Steiner forests, Steiner networks [19] - but none apply to our setting. Algorithms for computing t -spanners [2] come closest: t -spanners preserve all shortest paths, within a factor t . However, they may violate an order-preserving condition that is important for join paths. See details in Section 2.

1.4 Our Contributions

We propose a new relational schema summarization approach that is adaptive to user-specified tables of interest, and provides linkage information between tables. Given a set of query tables and a summary budget, we define an optimization problem that selects the most relevant budget tables as part of the summary. In addition, the summary preserves the most informative join paths between query tables. We prove that this problem is NP-Hard, and cast it as an integer program (under some simplifying assumptions). The integer program has an elegant structure, and can be solved efficiently by IP software. See Section 3.

In Section 4 we define a novel weight function over schema edges, using information theoretic measures that quantify the notion of informative joins.

Finally, we conduct an extensive experimental study in Section 5 and show that our approach computes highly relevant summaries. The relevance is measured via two independent quality measures based on query logs. We also compute summary graphs using the weight function in [21], and conclude that summaries using our information-theoretic weights are both more relevant and more consistent.

2. DEFINITIONS

The schema graph $\mathcal{G} = (\mathcal{R}, \mathcal{E})$ of a relational database is defined in the usual manner: the nodes correspond to tables $R \in \mathcal{R}$, and the edges to foreign/primary key constraints (joins). Edges are undirected.

Let $wt : \mathcal{E} \rightarrow \mathbb{R}^+$ be a weight function over the edges of \mathcal{E} , such that $wt(R, S) \geq 0$ for all $(R, S) \in \mathcal{E}$. Function wt is a dissimilarity measure, i.e., the smaller the weight $wt(R, S)$, the more important the join connection between tables R and S . A detailed discussion of how to choose wt is deferred to Section 4. We extend wt to a distance function d , defined as the shortest (weighted) distance in the graph between a pair of tables. More precisely, for any path $\Pi : R = R_0 - R_1 - \dots - R_\alpha = S$, let $wt(\Pi) = \sum_{i=0}^{\alpha-1} wt(R_i, R_{i+1})$. For any pair of tables $R, S \in \mathcal{R}$, the distance in graph \mathcal{G} between

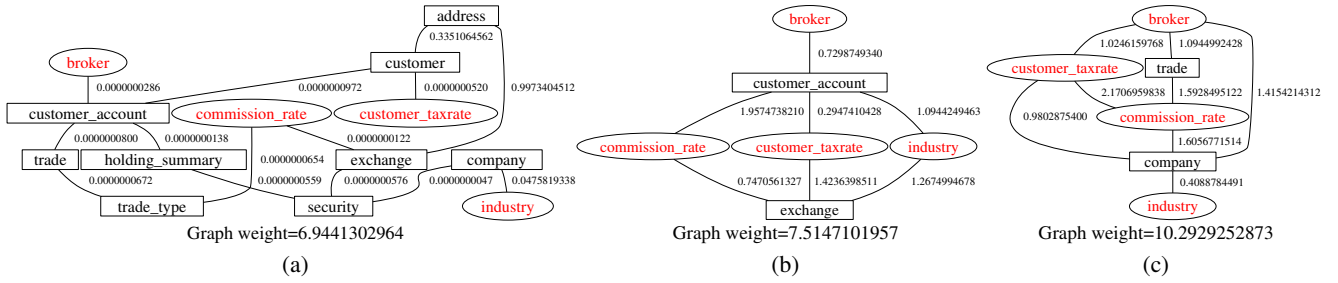


Figure 2: Summary graphs for TPC-E. Query tables are shown as oval nodes. (a) Union of shortest paths between query tables; (b) Optimal summary graph for $B = 2$; (c) Non-optimal summary graph with $B = 2$.

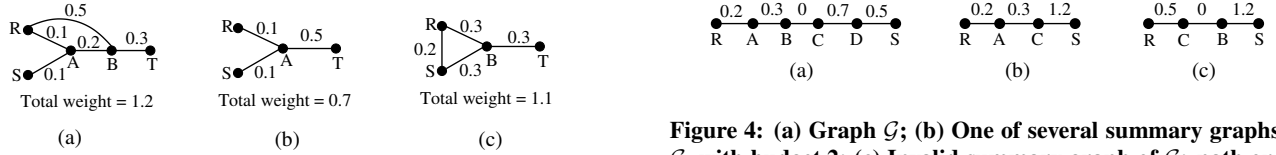


Figure 3: (a) Graph \mathcal{G} ; (b) Optimal summary graph of \mathcal{G} , with respect to query-set $\{R, S, T\}$ and budget 1; (c) A non-optimal summary graph of \mathcal{G} .

R and S is defined as:

$$d_{\mathcal{G}}(R, S) = \min_{\Pi=R \dots S} wt(\Pi).$$

In the summary graph, edges between nodes correspond to paths from the original schema graph. We refer to them as *metaedges*, to distinguish them from the join edges \mathcal{E} . Formally, $\mathcal{E}_m = \{(R, S) \mid \exists \Pi = R \dots S \text{ path in } \mathcal{G}\}$ is the set of metaedges, and $\mathcal{G}_m = (\mathcal{R}, \mathcal{E}_m)$ is the *transitive closure* of \mathcal{G} (if \mathcal{G} is connected, \mathcal{G}_m is a clique). The summary graph is a subgraph of \mathcal{G}_m . We extend the weight function to metaedges, as follows:

Definition 1. Let $\mathcal{G} = (\mathcal{R}, \mathcal{E})$ be a schema graph, and $(R, S) \in \mathcal{E}_m$ be a metaedge. We define $wt(R, S) = d_{\mathcal{G}}(R, S)$.

Example 1. Figure 3(c) shows a subset of metaedges for the graph in Figure 3(a). Note that the *metaedge* (R, B) has weight 0.3, which is the sum of weights along the shortest path $R - A - B$. This is distinct from the weight of 0.5 on the *edge* (R, B) from Figure 3(a). Hence, although tables R and B have a join edge, their most informative connection is via joins with table A .

Our goal is to compute a summary graph with minimum sum of metaedge weights (recall that smaller weights imply stronger connections between tables). We now discuss the properties that summary graphs must satisfy. Let $\mathcal{S} = (\mathcal{R}_s, \mathcal{E}_s)$ be a summary graph of \mathcal{G} with respect to a query set \mathcal{Q} and budget B . We want $\mathcal{Q} \subseteq \mathcal{R}_s \subseteq \mathcal{R}$ and $\mathcal{E}_s \subseteq \mathcal{E}_m$; also, to meet the budget, we need $|\mathcal{R}_s| \leq |\mathcal{Q}| + B$. The summary must preserve shortest paths between query tables, i.e., $d_{\mathcal{S}}(R, S) = d_{\mathcal{G}}(R, S)$.

In addition, we also need an order-preservation condition, as explained below: For each $R, S \in \mathcal{Q}$, and for any shortest path Π in \mathcal{S} , there must exist a shortest path Π' in \mathcal{G} such that the sequence of tables in Π is a sub-sequence of the sequence of tables in Π' .

Example 2. Figure 4 illustrates why we need such a condition. Suppose that we want to summarize the graph \mathcal{G} from Figure 4(a),

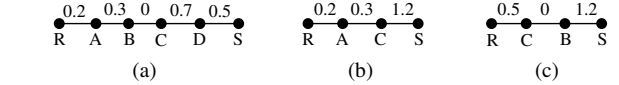


Figure 4: (a) Graph \mathcal{G} ; (b) One of several summary graphs of \mathcal{G} , with budget 2; (c) Invalid summary graph of \mathcal{G} : path order violation.

where $\mathcal{Q} = \{R, S\}$ and $B = 2$. There are many possible summaries, e.g., the path $R - A - C - S$ depicted in Figure 4(b). However, the path $R - C - B - S$ in Figure 4(c) is not a valid summary, even though it preserves the shortest distance between R and S : tables R, C, B and S appear in a different order in Figure 4(c), than in Figure 4(a). Presenting a user with the graph from Figure 4(c) would give a false idea about the semantics of the database, since the logical order of the joins is violated.

Order preservation conditions are not common in graph algorithms, and make the problem more challenging. In particular, *t*-spanners [2] do not guarantee order preservation.

The following definition summarizes the above discussion.

Definition 2. Let $\mathcal{G} = (\mathcal{R}, \mathcal{E})$ denote a schema graph with weight function wt over its edges, and let \mathcal{E}_m denote its metaedges. Let $\mathcal{Q} \subseteq \mathcal{R}$ be a *query set* and let $B \geq 0$ be the *summary budget*. A graph $\mathcal{S} = (\mathcal{R}_s, \mathcal{E}_s)$ is a *summary graph* of \mathcal{G} with respect to \mathcal{Q} if it satisfies the following conditions:

- (i) $\mathcal{Q} \subseteq \mathcal{R}_s \subseteq \mathcal{R}$ and $|\mathcal{R}_s| \leq |\mathcal{Q}| + B$.
- (ii) For any pair of query nodes $R, S \in \mathcal{Q}$, $d_{\mathcal{G}}(R, S) = d_{\mathcal{S}}(R, S)$.
- (iii) For any shortest path Π between R, S in \mathcal{S} , there exists a shortest path Π' between R, S in \mathcal{G} such that the sequence of tables in Π is a sub-sequence of the sequence of tables in Π' .

Given \mathcal{Q} and B , a summary graph \mathcal{S} is *optimal* if it minimizes $wt(\mathcal{S}) = \sum_{e \in \mathcal{E}_s} wt(e)$ over all summary graphs of \mathcal{G} with respect to \mathcal{Q} and B .

Example 3. Figure 3 illustrates the importance of choosing an optimal summary graph. Both graphs in Figures 3(b) and (c) are valid summary graphs of the graph in Figure 3(a), with respect to the query set $\mathcal{Q} = \{R, S, T\}$ and for budget $B = 1$. However, by retaining node A , which lies on all three shortest paths (i.e., between pairs (R, S) , (R, T) and (S, T)), the graph in Figure 3(b) has significantly lower weight. Thus, there is a direct connection between optimizing the weight of a summary graph and choosing those nodes that are “hubs” in the summary.

3. COMPUTING SUMMARY GRAPHS

Computing the optimal summary graph is NP-Hard, as we show in Section 3.1. Moreover, even approximate solutions seem elusive because of the order preservation condition. The most common case of violating the order occurs because of 0-weight edges, as in Figure 4(c). Other cases may occur because of multiple shortest paths between two nodes. We have encountered both cases in our experiments with the TPC-E database. To reduce the complexity of the problem, we add small random noise to each edge, so that all weights are positive and ties between multiple shortest paths are broken. Details on generating noise values are in Appendix A, along with an explanation of why we expect this heuristic not to affect the quality of results.

3.1 Properties of Summary Graphs

For the remainder of this paper, we therefore assume that all weights are non-zero, and that for every pair of nodes $R, S \in \mathcal{R}$, there is a unique shortest path connecting them in \mathcal{G} . Even with these restrictions, the problem remains NP-Hard. The following result follows by reduction from CLIQUE IN $(n - 4)$ -REGULAR GRAPHS; see proof in the Appendix B.

THEOREM 1. *Let \mathcal{G} be a schema graph. Computing the optimal summary graph of \mathcal{G} with respect to \mathcal{Q} and B , where \mathcal{Q} is a set of query nodes from \mathcal{G} and B is the summary budget, is NP-Hard.*

However, we show that it is sufficient to compute the optimal summary graph for a smaller graph $\mathcal{P} \subseteq \mathcal{G}$, which consists only of the shortest paths between nodes of \mathcal{Q} . We also prove a nice property of the optimal solution. The proof is in Appendix B.

PROPOSITION 1. *Let \mathcal{Q} be a set of query nodes in a schema graph \mathcal{G} . Let \mathcal{P} be the union of shortest paths between pairs of nodes in \mathcal{Q} . Then:*

- (i) *Any optimal summary graph of \mathcal{G} with respect to \mathcal{Q} is also an optimal summary graph of \mathcal{P} with respect to \mathcal{Q} , and vice versa.*
- (ii) *Every metaedge (A, B) in an optimal summary graph of \mathcal{G} (with respect to \mathcal{Q}) satisfies the property that A and B appear together on at least one of the shortest paths in \mathcal{P} (i.e., there are no cross-over metaedges connecting nodes from different paths).*

This allows us to formulate an elegant integer program over \mathcal{P} , for which existing IP solvers like CPLEX can quickly compute a solution. Moreover, the size of shortest paths in \mathcal{P} is often constant (hence, independent of the size of \mathcal{G}). Whenever this holds, $|\mathcal{P}| = O(|\mathcal{Q}|^2)$ and our solution is highly scalable. (Recall that \mathcal{Q} is a query set reflecting user interest and/or display capacity, so we expect $|\mathcal{Q}| \leq 10$.)

3.2 Integer Program

Let \mathcal{P} denote the union of shortest paths between nodes in \mathcal{Q} , as above. We formulate the integer program over graph \mathcal{P} .

Let x_{uv} , $u, v \in \mathcal{R}$, be metaedge variables, and y_u , $u \in \mathcal{R}$, be node variables; $x_{uv} = 1$, resp. $y_u = 1$, if and only if metaedge (u, v) , resp. node u , is in the summary graph. For technical reasons, we define the metaedges to be directed. However, we require $x_{uv} = x_{vu}$ for all (u, v) , i.e., the summary graph contains either both directed metaedges, or neither. This allows us to get rid of directionality at the end, and return an undirected summary graph.

For each pair of query nodes i and j , we denote by $\Pi_{ij} \in \mathcal{P}$ the shortest path between i and j . For each node $u \in \Pi_{ij}$, let $Pred_{ij}(u)$, resp. $Succ_{ij}(u)$, denote the set of nodes that precede, resp. succeed, node u along path Π_{ij} , when traversed from i to j . For example, in Figure 3(a), $Pred_{RT}(B) = \{R, A\}$ and

$Succ_{RT}(B) = \{T\}$. Let $MetaNbrs(u) = \bigcup_{i,j \in \mathcal{Q}} (Pred_{ij}(u) \cup Succ_{ij}(u))$ denote the set of all nodes preceding or succeeding u on any shortest path of \mathcal{P} . We define the following integer program:

$$\begin{aligned}
 & \text{Minimize} && \sum_{u,v} x_{uv} wt(u, v) \\
 & \text{Subject to:} && \\
 (1) & \sum_{w \in Succ_{ij}(v)} x_{vw} \geq x_{uv}, && \forall \Pi_{ij} \in \mathcal{P}, \forall u, v \in \Pi_{ij} \\
 (2) & \sum_{w \in Pred_{ij}(u)} x_{uw} \geq x_{uv}, && \forall \Pi_{ij} \in \mathcal{P}, \forall u, v \in \Pi_{ij} \\
 (3) & \sum_{u,v \in \Pi_{ij}} x_{uv} \geq 1, && \forall \Pi_{ij} \in \mathcal{P} \\
 (4) & \sum_{v \in MetaNbrs(u)} x_{uv} \leq y_u |MetaNbrs(u)|, && \forall u \\
 (5) & \sum_{u \notin \mathcal{Q}} y_u \leq B \\
 (6) & x_{uv} = x_{vu}, && \forall u, v \\
 (7) & y_u = 1, && \forall u \in \mathcal{Q} \\
 (8) & x_{uv}, y_u \in \{0, 1\}, && \forall u, v
 \end{aligned}$$

The following result states that the integer program is correct; the proof is in Appendix B.

PROPOSITION 2. *The optimal solution of the above integer program is an optimal summary graph of \mathcal{G} with respect to query set \mathcal{Q} and summary budget B .*

4. EDGE WEIGHTS

In this section, we propose a weight function over schema edges that quantifies how informative they are using well-known concepts from information theory.

To arrive at a principled definition for a weight function, let us first consider a subtle distinction. Let $S - R - T$ be a path in the schema graph. There are two possible paths corresponding to it at the column level, depending on whether or not the two join edges of R involve the same column. For example, Figure 5(a) shows a join path $S - R - T$ that involves two different columns of R , i.e., $R.B$ and $R.C$. In this case, there is at least one more (implicit) relationship that, while not depicted at schema level, is nevertheless used for computing the join result: a co-occurrence relationship for columns $R.B$ and $R.C$. More precisely, a tuple of values (x, y) satisfies such a relationship if the tuple appears in the projection of R on $\{R.B, R.C\}$. Figure 5(a) depicts, in fact, two such relationships inside table R (edges $(R.B, R.A)$ and $(R.A, R.C)$) for reasons that we explain below. By contrast, if path $R - T - S$ enters and exits table T on the same column, so there is no need to process any co-occurrences.

Our goal is to define the weights of schema edges in a way that accurately reflects which of the two cases applies at column-level.

4.1 Column-Level Graph

We define the column-level graph corresponding to a schema graph, as follows. All nodes in the graph are either primary keys in their respective tables, or foreign keys with respect to primary keys of other tables. (Joins between tables are issued over columns for which a foreign/primary key constraint is specified in the schema.) There is an edge between any foreign key and its primary key - denoted *inter-table edge*; and between the foreign key and the primary key of the table to which it belongs - denoted *intra-table edge*. For example, in Figure 5(a), the inter-table edge $(R.B, S.D)$ is included because there is a foreign/primary key constraint between the two columns; while the intra-table edge $(R.A, R.B)$ is included because $R.A$ is the primary key of R . However, the edge $(R.B, R.C)$ is not in the graph.

Note that, for the sequence of joins $S.D - R.B; R.C - T.F$, the path between $S.D$ and $T.F$ in the column-level graph contains two intra-table edges: $(R.A, R.B)$ and $(R.A, R.C)$. Why not connect

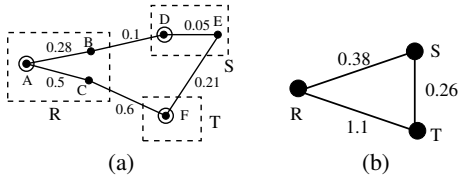


Figure 5: (a) Example column graph: dashed rectangles represent tables; circled nodes are primary keys; (b) Corresponding edge weights of the schema graph.

$R.B$ and $R.C$ directly? Recall that an intra-table edge represents a co-occurrence relationship. Since $R.A$ is the primary key of R , the semantics of the join $S.D - R.B; R.C - T.F$ remains the same, whether we regard it as traversing the single edge $(R.B, R.C)$, or the two edges $(R.A, R.B)$ and $(R.A, R.C)$. However, the combinatorial structure of the two possible paths in the column-level graph is different. Moreover, the edge weights can be impacted by whether or not one endpoint is a primary key; see Appendix C. For a fair comparison with inter-table edges (which always have one endpoint a primary key), we require all edges to have one endpoint a primary key.

Multi-column keys Every multi-column key is a node in the column level graph, with a multi-column foreign key connected to the primary key of its table. Multi-column foreign/primary key constraints are represented as edges between the respective nodes.

We define edge weights in the column-level graph using information-theory measures, then extend them to weights in the schema graph.

4.2 Defining Weights

We assume familiarity with information theory concepts. See Appendix C for a quick primer. For a joint distribution (X, Y) , let $I(X, Y)$ and $H(X, Y)$ denote its mutual information, resp. its entropy. Then $D(X, Y) = \frac{H(X, Y) - I(X, Y)}{H(X, Y)} = 1 - \frac{I(X, Y)}{H(X, Y)}$ is a distance function [10].

Column-level graph We define the *weight* of an edge (C_1, C_2) in the column-level graph to be $D(C_1, C_2)$, where $D(\cdot, \cdot)$ is the distance function defined above. Columns C_1 and C_2 are regarded as distributions of values (or distributions of tuples, if they are multi-columns). However, to apply function D , we also need to specify the *joint distribution* of C_1 and C_2 .

When C_1 and C_2 belong to the same table R , we define their joint distribution (C_1, C_2) to be the projection of R on $\{C_1, C_2\}$, consistent with prior work [7, 15]. For the case when C_1 and C_2 belong to different tables, we are not aware of any prior result that defines a joint distribution based on their join. We propose using the output of the *full outer join* as their joint distribution. The reason is that, unlike the inner join, the outer join contains values that do not match, i.e., pairs of type $(val, NULL)$. We want our weight function to be aware of such pairs, and penalize those joins with excessive numbers of unmatched values.

We discuss two examples of joint distributions. For convenience, we also show their pointwise mutual information $i(\cdot, \cdot)$; see Appendix C for the definition.

Example 4. Suppose that in Figure 5, the projection of R on (A, B) is as follows:

A	1	2	3	4	5	6	7	8
B	2	2	1	2	3	5	5	6
$i(a,b)$	$\log \frac{8}{3}$	$\log \frac{8}{3}$	3	$\log \frac{8}{3}$	3	2	2	3

Then $I(A, B) = \sum_{(a,b) \in (A,B)} p(a,b) i(a,b) = \frac{1}{8}(13 + 3 \log \frac{8}{3}) \approx 2.155$, and $H(A, B) = - \sum_{(a,b) \in (A,B)} p(a,b) \log p(a,b) = \log 8 = 3$. We have $D(A, B) \approx 0.28$.

Example 5. Suppose that in Figure 5, the foreign key column E has values $\{1, 2\}$, and the primary key column F has values $\{1, 2, 3, 4\}$. The result of the full outer join is:

F	1	1	2	3	4
E	1	1	2	NULL	NULL
$i(e,f)$	$\log \frac{5}{2}$	$\log 5$	$\log \frac{5}{2}$	$\log \frac{5}{2}$	$\log \frac{5}{2}$

Then $I(E, F) = \sum_{(e,f) \in (E,F)} p(e,f) i(e,f) = \frac{1}{5}(2 \log \frac{5}{2} + \log 5 + \log \frac{5}{2} + \log \frac{5}{2}) \approx 1.52$, and $H(E, F) = \frac{1}{5}(2 \log \frac{5}{2} + 3 \log 5) \approx 1.922$. We have $D(E, F) \approx 0.21$.

Schema graph We are now ready to define the weight function for table-level edges, as follows. Let (R, S) be an edge in the schema graph. Then $wt(R, S)$ is the (minimum) sum of weights between the primary keys of tables R and S , along the column-level path that contains only columns from R and S . For example, in Figure 5(a), we compute $wt(R, S) = wt(R.A, R.B) + wt(R.B, S.D) = 0.38$. In general, there may be another column-level path between R and S - say, if a foreign key column of S was connected to $R.A$. In that case, $wt(R, S)$ would be the minimum of the two path weights. Figure 5(b) shows the weights of schema edges corresponding to Figure 5(a).

5. EXPERIMENTAL EVALUATION

We conduct our experimental evaluation over the TPC-E [18] benchmark database, which simulates the On-Line Transaction Processing (OLTP) workload of a brokerage firm. We chose a benchmark database for two reasons: First, it has a well-specified schema, including all foreign/primary key constraints. Second, it has a transaction log, which we use to define independent quality measures for our summary graphs.

Experimental Setup We generate two database instances for the TPC-E schema, using EGen.v3.14², with parameters as listed in Table 1, Appendix D. The instances, which we call TPCE-1 and TPCE-2, have significantly different table sizes. Thus, we verify the robustness of our method over different datasets, but for the same schema. All methods are implemented in Java, and evaluated on a PC with 2.33GHz Core2 Duo CPU and 3.25G RAM.

5.1 Quality Measures

To evaluate the quality of our results, we propose two measures that quantify how well a summary graph represents the tables and edges occurring most frequently in a query log. The intuition is that the most informative tables and edges in a database are also the most frequently queried. Exceptions may occur; see, e.g., [21].

Let $f(T)$ denote the frequency with which a table T appears in the transaction log. Similarly, let $f(R, S)$ be the frequency of edge (R, S) in the log. Details about the TPC-E transaction log are in Appendix D.

Definition 3. Let $\mathcal{G} = (\mathcal{R}, \mathcal{E})$ be a schema graph and $\mathcal{S} = (\mathcal{R}_s, \mathcal{E}_s)$ be a summary graph of \mathcal{G} with respect to query set \mathcal{Q} and budget B . A table T is *covered* by \mathcal{S} , denoted $T \prec \mathcal{S}$, if $T \in \mathcal{R}_s$. An edge $(R, S) \in \mathcal{E}$ is *covered* by \mathcal{S} , denoted $(R, S) \prec \mathcal{S}$, if (R, S) appears on a shortest path between query tables in \mathcal{Q} , and at least one of the tables R or S is covered by \mathcal{S} .

²EGen is a software package provided by TPC [18].

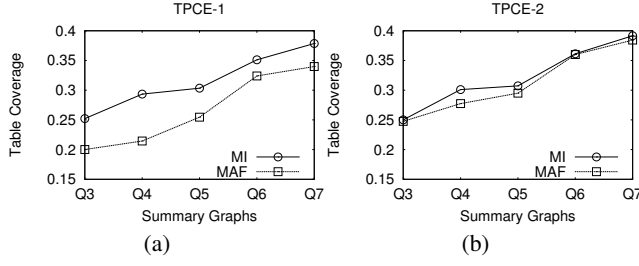


Figure 6: Table coverage of summary graphs (B=2) based on MI and MAF weights.

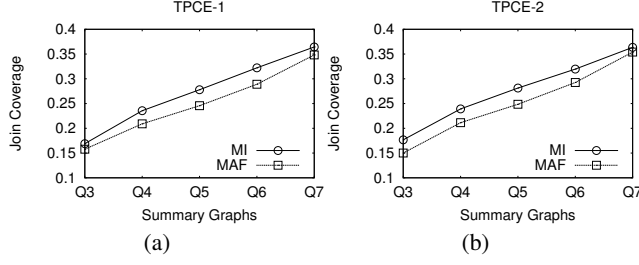


Figure 7: Join coverage of summary graphs (B=2) based on MI and MAF weights.

The reason for the above definition of covered edges is as follows. First, recall that the edges \mathcal{E}_s of \mathcal{S} are metaedges of \mathcal{G} . Some of them may coincide with edges from \mathcal{E} , but in general this is unlikely. Therefore, it would be pointless to say that an edge is covered if it appears in \mathcal{S} . The next best thing is to define an edge as covered if it appears on a shortest path between a pair of query tables in \mathcal{Q} . However, this does not help us compare different summary graphs for the same query set \mathcal{Q} . We therefore add the additional requirement that at least one endpoint of the edge appears in the summary. This is consistent with the summary usage discussed in Section 1, where users can request edges adjacent to summary nodes.

We define our independent quality measures as follows.

Definition 4. Let $\mathcal{G} = (\mathcal{R}, \mathcal{E})$ be a schema graph and $\mathcal{S} = (\mathcal{R}_s, \mathcal{E}_s)$ be a summary graph of \mathcal{G} .

- (i) The *table coverage* of \mathcal{S} is $\Phi_t = \frac{\sum_{T \prec \mathcal{S}} f(T)}{\sum_{R \in \mathcal{R}} f(R)}$.
- (ii) The *join coverage* of \mathcal{S} is $\Phi_j = \frac{\sum_{(R,S) \prec \mathcal{S}} f(R,S)}{\sum_{(A,B) \in \mathcal{E}} f(A,B)}$.
- (iii) The *density* of \mathcal{S} is $\frac{|\mathcal{E}_s|}{|\mathcal{R}_s|}$.

Assuming that the transaction log frequency of tables and edges is proportional to their importance, then the first two measures reflect the fraction of table, resp. join edge, importance captured by the summary. The last measure reflects the complexity of the output: the smaller the density, the easier the graph is to understand.

For TPC-E, the distribution of join edge frequencies is significantly more skewed than that of table frequencies, with a few edges having very high frequency. Therefore, as we see throughout this section, differences in table coverage among various summary graphs are more pronounced than for join coverage (whenever summary graphs include the same heavy edges, they have nearly identical join coverage, despite being otherwise quite different).

Given the above observation, we consider table coverage to be the more informative measure.

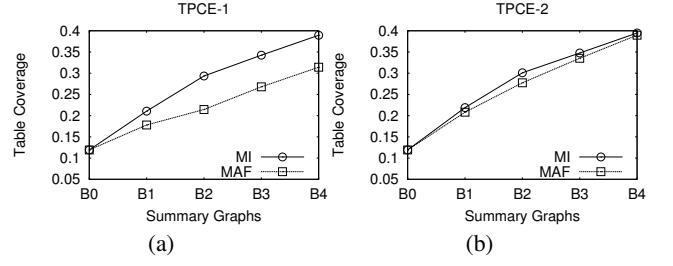


Figure 8: Table coverage of summary graphs (Q=4) based on MI and MAF weights.

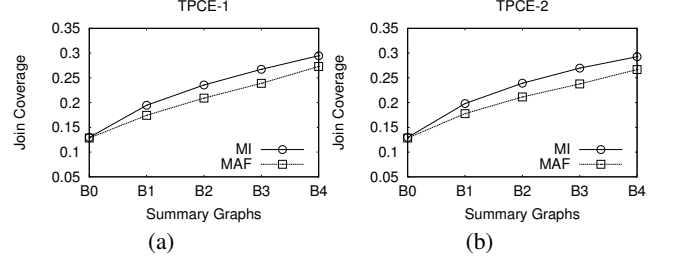


Figure 9: Join coverage of summary graphs (Q=4) based on MI and MAF weights.

5.2 Comparing Weight Functions

We compare the quality of summary graphs obtained from two different weight functions: The first function, which we refer to as MI, is the one we defined in Section 4. The second function, denoted MAF, is based on the distance measure proposed in [21] (the abbreviations stand for Mutual Information, resp. Matched Average Fanout). See the Appendix for the definition of the MAF weight.

Let $Q = |\mathcal{Q}|$. We vary Q from 3 to 7 and generate, for each value, 100 different sets of query tables, randomly chosen from the 32 tables of TPC-E. For each set of query tables, we then generate up to 5 summary graphs by varying B from 0 to 4. In some cases, the union of shortest paths between query tables contains less than 4 non-query nodes, so we generate fewer summary graphs. (We have run experiments with higher values for Q and B , but as $Q + B$ approaches the total number of tables 32, all coverage values converge to 1 and differences between different settings become insignificant. We therefore report results for $Q + B \leq 9 \approx 28\%$ of tables, which we believe is a reasonable summary size.)

For a fixed setting of Q and B , we compute the average coverage of all summary graphs generated for those values. Figures 6 and 7

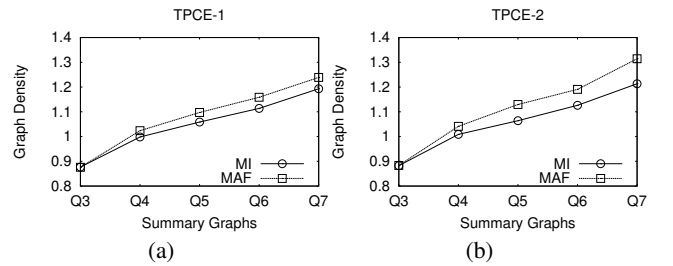


Figure 10: Density of summary graphs (B=2) based on MI and MAF weights.

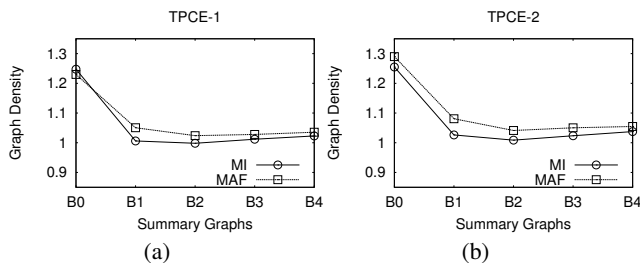


Figure 11: Density of summary graphs ($Q=4$) based on MI and MAF weights.

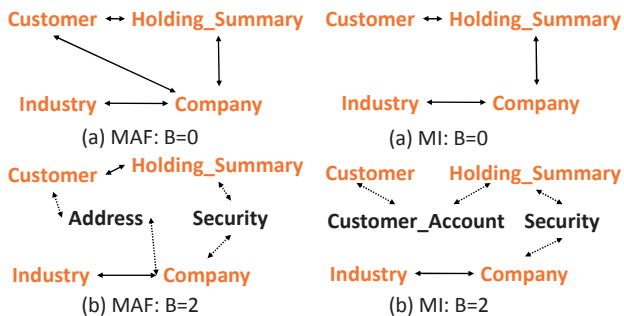


Figure 12: Summary graphs based on MAF, resp. MI weights.

show how the table and join coverage vary for different values of Q and for $B = 2$. We then fix $Q = 4$ and report the coverage when varying B from 0 to 4, in Figures 8 and 9. Clearly, summary graphs based on MI weights achieve higher table and join coverage than those based on MAF weights, for both TPCE-1 and TPCE-2. Note that we use the same query sets \mathcal{Q} in both TPCE-1 and TPCE-2, and for both MI and MAF.

Not only is the quality of summary graphs higher under the MI weights, but their complexity is smaller. Figures 10 and 11 show that the density of summary graphs for MI weights is consistently smaller than the density of summary graphs under MAF weights. The densities in Figure 10 are computed for the graphs in Figure 6, while those in Figure 11 correspond to graphs in Figure 8. Since graph density is directly correlated to ease of understanding, we conclude that MI-based summary graphs are both more informative and more concise than their MAF-based counterparts.

Clearly, the differences in the quality of results are due to the fact that the input values for the corresponding integer programs are different. The most striking difference is that for some sets \mathcal{Q} , the shortest paths between query tables are different under the two weight functions. We illustrate such an example in Figure 12, which shows several summary graphs obtained for query tables *Customer*, *Holding_Summary*, *Company* and *Industry*.

When $B = 0$, the MAF-based summary graph has one more edge between *Customer* and *Company*. This is because the shortest path between these tables is different from that based on MI. The difference becomes more clear when $B = 2$: For MI-based weights, the shortest path between *Customer* and *Company* goes through tables *Customer_Account*, *Holding_Summary* and *Security*. Hence, when $B = 0$, this path is summarized as *Customer*–*Holding_Summary*–*Company*. By contrast, for MAF-based weights, the shortest path between *Customer* and *Company* goes through table *Address*. For $B = 0$, it can only be summarized as the metaedge *Customer*–*Company*. This example also illustrates how a user could drill down for details: e.g., requesting details on the

metaedge *Customer*–*Company* in the summary graph for MAF and $B = 0$ brings up the path *Customer*–*Address*–*Company*.

However, the main difference in these graphs is qualitative. The MAF-based shortest path *Customer*–*Address*–*Company* is almost meaningless. It connects customers and companies that share an address - which may be coincidental. By contrast, the MI-based shortest path between *Customer* and *Company* conveys significant information about the operations of the brokerage firm simulated by TPCE-E.

Consistency The values reported in Figures 6 and 8 indicate that summary graphs based on the MI weight achieve consistent coverage for both TPCE-1 and TPCE-2. By contrast, results based on MAF weights are more instance-dependent, with graphs in TPCE-1 having worse table coverage than those in TPCE-2.

For the remainder of this section, we focus only on MI weights. Given their consistency, and due to space constraints, we report only results for TPCE-1.

5.3 Choosing Budget Tables

In this section, we evaluate the effect that our strategy for choosing budget tables has on the coverage and density of summary graphs. Clearly, one strategy we could compare against is choosing the budget nodes randomly from graph \mathcal{P} , which is the union of shortest paths that connect query tables. However, for many of the query sets we have generated, the corresponding graphs \mathcal{P} contain fewer than 5 nodes. Choosing, e.g., 2 nodes randomly out of 5 is a high variance experiment, which cannot be mitigated by reporting the average over many trials (there are only 10 possible outcomes, one of which coincides with that of the integer program).

Coordinated Summary Graphs Instead, our experiments are conducted under the following scenario. Let the vertex size $S = Q + B$ of the summary graph be fixed. A baseline strategy we compare with is to randomly choose all S nodes in the output. On the other hand, we can choose only the Q query nodes randomly, for some fixed Q , and the remaining $S - Q$ budget nodes based on our integer program method. Varying Q from some minimum value Q_{min} up to the maximum value S results in summary graphs with increasingly random vertex sets. To control the effects of randomness and have consistent comparisons, we enforce the condition that the query set of size Q is a subset of the query set of size $Q + 1$, for $Q_{min} \leq Q \leq S$. We say that such query sets, and their corresponding summary graphs, are *coordinated*.

In our experiments, we set $S = 7$ and $Q_{min} = 4$. For $Q = Q_{min}$, we generate 100 different query sets of size Q , each chosen independently at random from the tables in TPC-E. For each such query set in turn, we generate a series of coordinated query sets for $Q = 5, 6, 7$, and compute coordinated summary graphs using budget $B = S - Q$. Some of these series are incomplete; e.g., there may be no summary graph with $Q = 4$ and $B = 3$ because there are only 2 non-query vertices in the union of shortest paths. We dropped the incomplete series from our experiments, leaving us with 85 series of coordinated graphs. (This is the reason we set $Q_{min} = 4$ and $S = 7$: over all the choices we tried, this resulted in the largest number of complete series).

We report the quality measures on the resulting summary graphs in Figure 13. We show the averages over the 85 summary graphs corresponding to each value Q . Figure 13(a) shows the trend in graph density. Because $Q + B = 7$, the total number of nodes in all summary graphs is the same. However, the graph density (and therefore, the number of edges) increases with decreasing budget nodes. The average graph density for $Q = 7, B = 0$ is almost twice that for $Q = 4, B = 3$. As the total number of nodes remains the same, the total number of edges in the summary increases. This

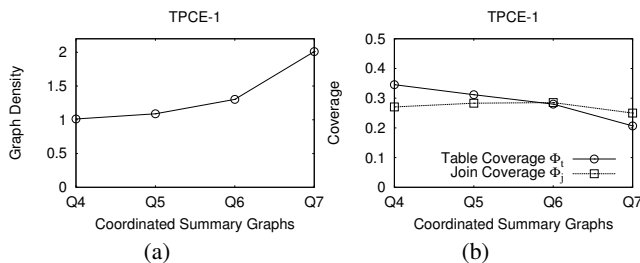


Figure 13: Density and coverage for coordinated summary graphs, for $Q + B = 7$.

indicates that the budget nodes we choose help reduce the complexity of summary graphs considerably.

Next, we evaluate the effect of budget nodes over the table and join coverage of summary graphs. Increasing the number of randomly chosen tables from $Q = 4$ to $Q = 7$ makes the average table coverage Φ_t drop from 34% to 20%; see Figure 13(b). This indicates that our strategy for choosing budget nodes increases the quality of the graphs (while reducing their complexity, as discussed above). The join coverage Φ_j has a different trend: it first increases slightly with Q , then decreases. We believe this can be explained as follows: Adding a new query table to the set means adding more shortest paths connecting it to other query tables. Thus, the set of edges over which the integer program is defined is larger, and we have a higher chance of increasing the join coverage. However, this chance is dampened by the fact that for higher Q we have lower B , and therefore fewer choices of budget nodes (recall that only edges adjacent to nodes in the summary count towards the join coverage). Of these two opposite forces, the former seems to have slightly higher impact, resulting in the trend in Figure 13(b). However, for $Q = 7$, i.e., for the case when we have no choice in budget nodes, the join coverage drops to the lowest value in the graph.

We conclude that our strategy for choosing budget nodes has meaningful effects on both the complexity and the quality of the resulting summary graphs.

5.4 Choosing Query Tables

We also studied how the quality of the summary graphs is impacted by several deterministic strategies for choosing query sets. These choices can be regarded as the system’s recommendations for users who do not have any specific query plans. Rather, such users may want a quick understanding of the most significant functionality of the database.

Computing the summary graphs for such system-chosen query sets can be regarded as an extension of the results from [22, 21]. More precisely, we choose the query sets based on the important tables and cluster outputs from [21]. However, by computing summary graphs based on them, we provide significant additional information on the schema.

We studied four deterministic strategies for query table selection, based on the notions of *table importance*, *cluster centers*, and *weighted cluster distance* developed in [21]. All four strategies achieve similar coverage, which is significantly higher than the coverage for a random choice of query tables. The improvement is more than 100%, for all $3 \leq Q \leq 7$. For example, for $Q = 7$, deterministic strategies achieve table coverage of $\approx 50\%$, with summary graphs that contain only $\approx 22\%$ of the tables. Due to space constraints, the graphs are in Appendix D.1.

We conclude that the deterministic strategies are extremely efficient in generating summary graphs with high relevance.

6. CONCLUSION

We have introduced a novel concept for summarizing complex schema graphs, called *summary graph*. The summary is adaptive to user-specified query tables, and preserves the most informative join paths between them. We proved that the problem of computing an optimal summary graph is NP-Hard, and solved it as an integer program. We have also defined an information-theoretic weight for join edges, which may prove of independent interest. Our extensive experimental study validates our weight definition as well as our summary model, and shows that our method returns high-quality summaries under independent quality measures.

Acknowledgment

We wish to thank Howard Karloff, who was instrumental in proving the NP-Hardness result.

7. REFERENCES

- [1] Standard specification of TPC benchmark. pages 57–172.
- [2] E. Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. *SIAM J. Comput.*, 28(1):210–236, 1998.
- [3] T. Dasu, T. Johnson, S. Muthukrishnan, and V. Shkapienyuk. Mining database structure; or, how to build a data quality browser. In *SIGMOD*, pages 240–251, 2002.
- [4] C. Faloutsos, K. S. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In *KDD*, pages 118–127, 2004.
- [5] H. He, H. Wang, J. Yang, and P. S. Yu. Blinks: ranked keyword searches on graphs. In *SIGMOD*, pages 305–316, 2007.
- [6] M. Jayapandian and H. V. Jagadish. Expressive query specification through form customization. In *EDBT*, pages 416–427, 2008.
- [7] J. Kang and J. F. Naughton. On schema matching with opaque column names and data values. In *SIGMOD*, pages 205–216, 2003.
- [8] G. Kasneci, S. Elbassuoni, and G. Weikum. Ming: mining informative entity relationship subgraphs. In *CIKM*, pages 1653–1656, 2009.
- [9] Y. Koren, S. C. North, and C. Volinsky. Measuring and extracting proximity graphs in networks. *TKDD*, 1(3):245–255, 2007.
- [10] A. Kraskov and P. Grassberger. Mic: Mutual information based hierarchical clustering. In *Information Theory and Statistical Learning*, pages 101–123, 2009.
- [11] G. Li, J. Feng, B. C. Ooi, J. Wang, and L. Zhou. An effective 3-in-1 keyword search method over heterogeneous data sources. *Inf. Syst.*, 36:248–266, April 2011.
- [12] B. Liu and H. V. Jagadish. A spreadsheet algebra for a direct data manipulation query interface. In *ICDE*, pages 417–428, 2009.
- [13] C. Ramakrishnan, W. H. Milnor, M. Perry, and A. P. Sheth. Discovering informative connection subgraphs in multi-relational graphs. *SIGKDD Explor. Newsl.*, 7:56–63, December 2005.
- [14] A. Rostin, O. Albrecht, J. Bauckmann, F. Naumann, and U. Leser. A machine learning approach to foreign key discovery. In *WebDB*, 2009, <http://webdb09.cse.buffalo.edu/program.html>.
- [15] D. Srivastava and S. Venkatasubramanian. Information theory for data management. In *SIGMOD Conference*, pages 1255–1256, 2010.
- [16] H. Tong and C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In *KDD*, pages 404–413, 2006.
- [17] H. Tong, C. Faloutsos, and Y. Koren. Fast direction-aware proximity for graph mining. In *KDD*, pages 747–756, 2007.
- [18] TPC-E. <http://www.tpc.org/tpce/tpc-e.asp>.
- [19] V. V. Vazirani. *Approximation Algorithms*. Springer, 2003.
- [20] W. Wu, B. Reinwald, Y. Sismanis, and R. Manjrekar. Discovering topological structures of databases. In *SIGMOD*, pages 1019–1030, 2008.
- [21] X. Yang, C. M. Procopiuc, and D. Srivastava. Summarizing relational databases. In *VLDB*, pages 634–645, 2009.
- [22] C. Yu and H. V. Jagadish. Schema summarization. In *VLDB*, pages 319–330, 2006.
- [23] M. Zhang, M. Hadjieleftheriou, B. C. Ooi, C. M. Procopiuc, and D. Srivastava. On multi-column foreign key discovery. In *VLDB*, pages 805–814, 2010.

APPENDIX

A. COMPUTING EDGE NOISE

To reduce the complexity of the summary graph problem to more manageable levels, we propose the following heuristic preprocessing, which breaks ties among shortest paths by adding random noise: Let Δ denote the smallest difference, over all pairs (R, S) , between the weights of the second-shortest, resp. shortest, paths connecting R and S . Let $n = |\mathcal{R}|$ be the number of tables, and $\varepsilon = \Delta/n$. For each edge $e \in \mathcal{E}$, we define $wt'(e) = wt(e) + \nu(e)$, where $\nu(e)$ is a noise value drawn uniformly at random from $(0, \varepsilon)$. It is easy to see that any shortest path under wt' is also a shortest path under wt : the weight of a shortest path is increased by at most $(n-1)\varepsilon < \Delta$. The reverse is not true: with high probability, for each pair of nodes only one of the shortest paths under wt is a shortest path under wt' (we re-run the procedure in those rare events when this fails).

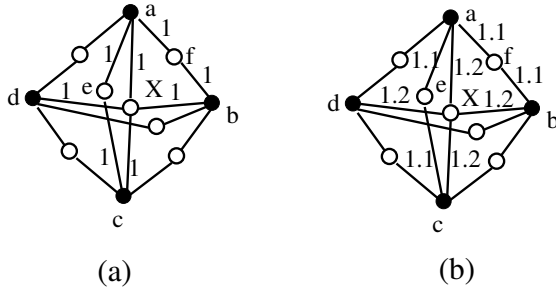


Figure 15: Impact of noise on summary graphs. Query nodes are filled circles: $Q = 4$. (a) original graph has summary of weight $\Theta(Q)$; (b) noisy graph has summary of weight $\Theta(Q^2)$.

In our experiments, $\varepsilon \approx 10^{-8}$. The resulting weights are all strictly positive. Such changes could impact the combinatorial structure of the optimal summary graph, as well as its weight. Figure 15 shows an example where unlucky choices of noise may lead to a significantly worse summary graph. For the graph in Figure 15(a), all edge weights are 1. Query nodes are filled circles, so $Q = 4$. There are 2 shortest paths between each pair of query nodes; e.g., $a - X - c$ and $a - e - c$ are shortest paths of weight 2 between a and c . Similarly, $a - X - b$ and $a - f - b$ are shortest paths between a and b . For budget $B = 1$, the optimal summary graph chooses X as the budget node, and achieves weight Q .

Suppose that after adding noise, all edges incident to X have weight 1.2, and all other edges have weight 1.1. Then X is no longer on any shortest path between query nodes. Assume that the summary graph chooses, e.g., f as the budget node (all other cases are symmetric). Then all query nodes, except a and b , are connected by direct metaedges of weight 2.2. The summary graph is almost a clique, of weight $\Theta(Q^2)$.

Despite the above example, we believe that in practice adding edge noise leads to good results, for the following two reasons:

- First, the effects of noise addition can be mitigated by conducting several independent experiments, and reporting the best summary graph overall (where the summary graphs are compared in terms of their original, noise-less weights).

- Second, the most informative connections between tables should be resilient to the addition of small noise, so that they still appear in the summary graphs. In other words, we do not expect real schema graphs to exhibit the same sensitivity to noise as our made-up example in Figure 15. This intuition is borne out by our experiments.

B. PROOFS

Proof of Theorem 1

The corresponding SUMMARY GRAPH DECISION problem is as follows: Given query set \mathcal{Q} , budget B and $\alpha > 0$, is there a summary graph \mathcal{S} of \mathcal{G} with respect to \mathcal{Q} and B such that $wt(\mathcal{S}) \leq \alpha$? The reduction is from the following NP-Hard problem, called CLIQUE IN $(n-4)$ -REGULAR GRAPHS: Given graph $J = (V, F)$ on n vertices, each vertex having degree $n-4$, and a number r , is there a clique in J of size r ?

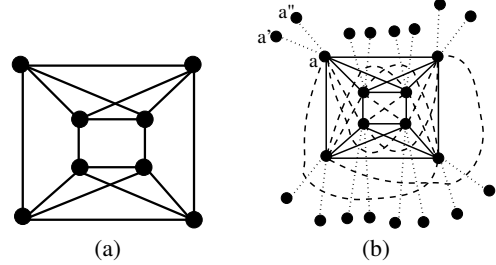


Figure 16: The reduction in the proof of Theorem 1: (a) an $(n-4)$ -regular graph J ; (b) the corresponding graph \mathcal{G} : solid edges have weight 1, dashed edges have weight 1.5, and dotted edges (e.g., $a - a'$ and $a - a''$) have weight 0.

We rephrase the summary graph decision problem as a choice of budget nodes, which induces a summary graph. For any budget set $\mathcal{B} \subseteq \mathcal{R} \setminus \mathcal{Q}$, $|\mathcal{B}| \leq B$, let $\mathcal{S}(\mathcal{B}) = (\mathcal{Q} \cup \mathcal{B}, \mathcal{E}_1)$ be the minimum weight summary graph induced by $\mathcal{Q} \cup \mathcal{B}$. We now define $f(\mathcal{B}) = wt(\mathcal{S}(\mathcal{B}))$. The decision problem is to determine if there exists $\mathcal{B} \subseteq \mathcal{R} \setminus \mathcal{Q}$, $|\mathcal{B}| \leq B$, such that $f(\mathcal{B}) \leq \alpha$.

Reduction: Let $J = (V, F)$ be an arbitrary instance of CLIQUE IN $(n-4)$ -REGULAR GRAPHS. Build a clique K on V , with edge weights $wt(e) = 1$ if $e \in F$, and 1.5 otherwise. Now triple the size of the vertex set by adding, for each node $u \in V$, two new nodes u', u'' , and edges $\{u, u'\}$ and $\{u, u''\}$, both of weight 0. See Figure 16. Let the schema graph $\mathcal{G} = (\mathcal{R}, \mathcal{E})$ be the resulting graph (which is not a clique) on $3n$ vertices, with edge weights 0, 1, or 1.5. Choose $\mathcal{Q} = \cup_{u \in V} \{u', u''\}$, i.e., all the $2n$ newly-added vertices. Choose budget $B = r$ (r is the parameter of the clique problem), and define the threshold $\alpha = (\beta - (n-4)(n-r)) - (1/2) \binom{r}{2}$, where $\beta = 1.5 \binom{n}{2} + 3r(n-r) + 6 \binom{n-r}{2}$.

We show that solving the summary decision problem for parameters \mathcal{Q} , B and α leads to a solution of Clique in $(n-4)$ -Regular Graphs. For all $i, j \in \mathcal{Q}$, Π_{ij} is as follows:

Case 1. If $\{i, j\} = \{u', u''\}$ for some u , then $\Pi_{ij} = u' - u - u''$ and $wt(\Pi_{ij}) = 0$.

Case 2. Otherwise, $(i, j) \in \{(u', v'), (u', v''), (u'', v'), (u'', v'')\}$, for some $u \neq v$, and Π_{ij} is the unique 3-edge path from i to j ; its first and last edges have cost 0 and its middle edge has cost 1 or 1.5, according to whether $\{u, v\}$ is in F or not, respectively.

In all cases, the Π_{ij} 's are *unique* shortest paths. Furthermore, every edge of \mathcal{G} is on one of the shortest paths.

Let \mathcal{B} be a set of vertices in J of size $B = r$. We compute $f(\mathcal{B})$: Pairs in Case 1 contribute 0. For pairs in Case 2: if neither u nor v is in \mathcal{B} , we get a contribution of $4wt(u, v)$; if exactly one is, the contribution is $2wt(u, v)$; and if both are, the contribution is $wt(u, v)$. Let a be the number of edges in J both of whose endpoints are in \mathcal{B} ; let b be the number of edges in J , exactly one of whose endpoints are in \mathcal{B} ; and let c be the number of edges in J neither of whose endpoints are in \mathcal{B} . A set \mathcal{B} that minimizes $f(\mathcal{B})$

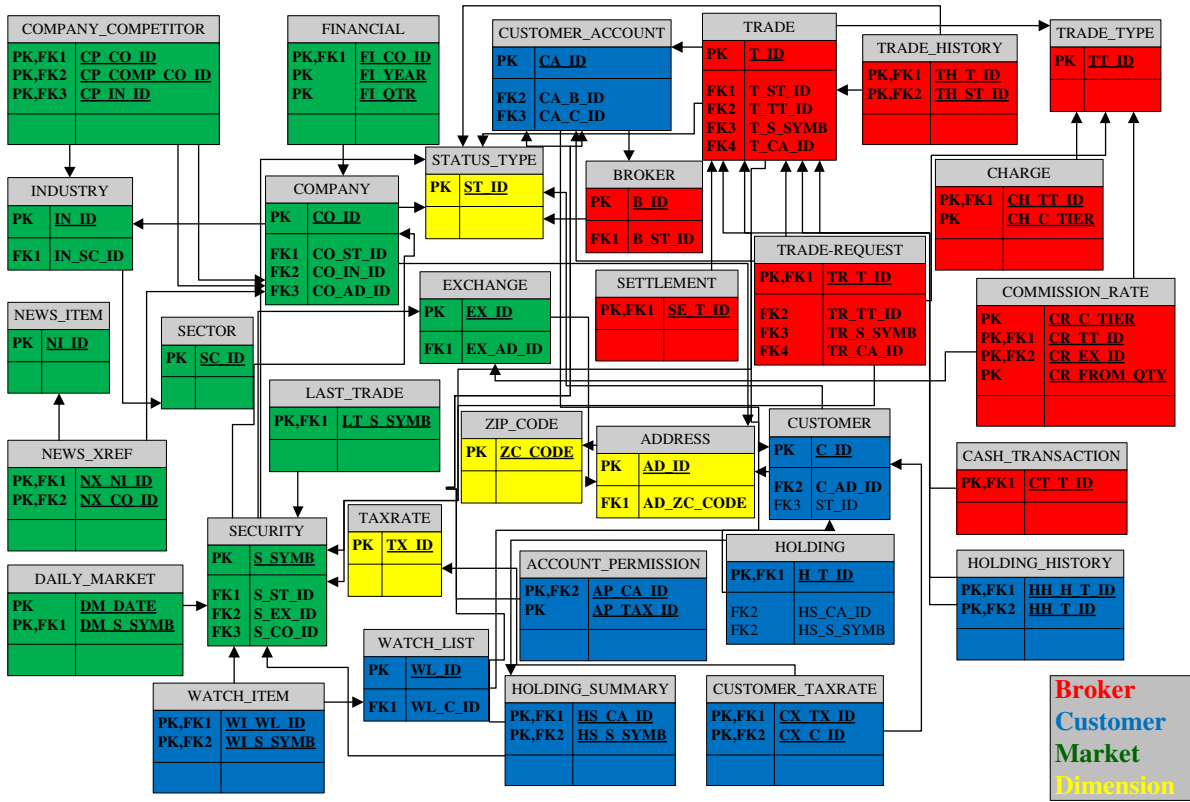


Figure 14: TPC-E Schema Graph.

is obtained by minimizing:

$$\begin{aligned}
 & [1 \cdot a + 1.5 \binom{r}{2} - a] + [2 \cdot 1 \cdot b + 2 \cdot 1.5(r(n-r) - b)] + [4 \cdot 1 \cdot c + 4 \cdot 1.5 \binom{n-r}{2} - c] \\
 & = [1.5 \binom{r}{2} + 3r(n-r) + 6 \binom{n-r}{2}] - a/2 - b - 2c.
 \end{aligned}$$

Let $\beta = 1.5 \binom{r}{2} + 3r(n-r) + 6 \binom{n-r}{2}$. Hence the goal is to minimize $\beta - (a/2 + b + 2c)$. Since the input graph is Δ -regular for $\Delta = n - 4$, $a + b + c = (\Delta/2)n$, so $c = (\Delta/2)n - a - b$. Our goal is now to minimize $(\beta - \Delta n) + (1.5a + b)$.

But $2a + b$ is the sum of degrees of the vertices in \mathcal{B} with respect to graph J , so $2a + b = \Delta r$. Therefore $1.5a + b = \Delta r - a/2$. The goal now is to minimize $(\beta - \Delta n + \Delta r) - a/2$, which means maximizing a . Therefore we can choose $a = \binom{r}{2}$, i.e., achieve cost $f(\mathcal{B}) = (\beta - \Delta n + \Delta r) - (1/2) \binom{r}{2} = \alpha$, if and only if J has a clique of size r . \square

Proof of Proposition 1

By definition of \mathcal{P} , $d_{\mathcal{P}}(R, S) = d_{\mathcal{G}}(R, S)$ for any pair of query nodes $R, S \in \mathcal{Q}$.

We prove (ii) first: Let \mathcal{S} be an optimal summary graph of \mathcal{G} with respect to \mathcal{Q} , and let (A, B) be an arbitrary edge in \mathcal{S} . Then (A, B) must appear on at least one shortest path $\Pi = R - \dots - A - B - \dots - S$ in \mathcal{S} , for some pair of nodes $R, S \in \mathcal{Q}$; otherwise, we could delete (A, B) and lower the total weight of \mathcal{S} . By definition, the shortest path Π must belong to \mathcal{P} .

To prove part (i), we first show that \mathcal{S} is a summary graph of \mathcal{P} . From the discussion above, any edge (A, B) of \mathcal{S} is a metaedge in \mathcal{P} (not just in \mathcal{G}). In addition, Definition 2, with \mathcal{P} playing the role of \mathcal{G} , is trivially verified. (Note that $d_{\mathcal{S}}(R, S) = d_{\mathcal{G}}(R, S) = d_{\mathcal{P}}(R, S)$ for any two query nodes $R, S \in \mathcal{Q}$.) Conversely,

let \mathcal{S}' be an optimal summary graph of \mathcal{P} with respect to \mathcal{Q} . We show that \mathcal{S}' is a summary graph of \mathcal{G} . Clearly, all edges in \mathcal{S}' are metaedges of \mathcal{G} and Definition 2(i) is true. Let $R, S \in \mathcal{Q}$ be two arbitrary query nodes. Then $d_{\mathcal{S}'}(R, S) = d_{\mathcal{P}}(R, S) = d_{\mathcal{G}}(R, S)$, so Definition 2(ii) holds. Let Π_1 be a shortest path in \mathcal{S}' between R and S . Since \mathcal{S}' is a summary graph of \mathcal{P} , there must exist a shortest path Π_2 in \mathcal{P} , between R and S , so that all nodes in Π_1 appear in the same order in Π_2 . By definition of \mathcal{P} , and since the shortest path between R and S is unique, Π_2 is also the shortest path in \mathcal{G} .

Since \mathcal{S} is an optimal summary graph of \mathcal{G} , and \mathcal{S}' is a summary graph of \mathcal{G} , we deduce $wt(\mathcal{S}) \leq wt(\mathcal{S}')$. Similarly, the optimality of \mathcal{S}' with respect to \mathcal{P} implies $wt(\mathcal{S}') \leq wt(\mathcal{S})$. Therefore $wt(\mathcal{S}) = wt(\mathcal{S}')$ and claim (i) is verified. \square

Proof of Proposition 2

We prove that any feasible solution of the integer program is a summary graph of \mathcal{P} with no cross-over metaedges, and viceversa. By Proposition 1(ii), the optimal solution of the integer program is an optimal summary graph of \mathcal{P} . Proposition 1(i) then implies that the optimal solution is also an optimal summary graph of \mathcal{G} .

Consider a feasible solution $[x, y]$. Inequalities (1)-(3) ensure that each pair of query nodes i, j remains connected in the summary graph, via metaedges along Π_{ij} : constraint (3) requires that we choose at least one metaedge (u, v) along each path Π_{ij} . Constraints (1) and (2) require that, once (u, v) is chosen, we must also choose some metaedges (w_1, u) and (v, w_2) along Π_{ij} . Applying constraints (1) and (2) iteratively to (w_1, u) and (v, w_2) , we obtain that i and j are connected by metaedges along path Π_{ij} . Clearly, this implies that Definition 2(ii)-(iii) is satisfied for graph \mathcal{P} .

Constraints (4), (5) and (7) ensure that Definition 2(i) is satisfied: constraint (4) ensures that, if we choose any metaedge with

endpoint u in the summary, then we also choose u in the summary, i.e., we count u towards the budget. Constraint (5) is the budget restriction, and constraint (7) ensures that Q is in the summary.

Conversely, if \mathcal{S} is a summary subgraph of \mathcal{G} with no crossover metaedges, define $x_{uv} = 1$, $y_u = 1$ and $y_v = 1$ for all metaedges (u, v) in \mathcal{S} . Inequalities (1)-(8) are clearly satisfied, so $[x, y]$ is a feasible solution. \square

C. MUTUAL INFORMATION

We briefly review the relevant definitions from information theory (for more details, see [15]). Let (X, Y) denote a joint distribution. For each tuple $(x, y) \in (X, Y)$, let $p(x, y)$ define the frequency with which the tuple (x, y) appears in the joint distribution. The *marginal probability on X* is $p_X(x) = \sum_y p(x, y)$, $\forall x \in X$, i.e., the frequency with which x appears as the first value in a tuple from (X, Y) . The marginal probability $p_Y(y)$ is similarly defined. The *pointwise mutual information* for any tuple $(x, y) \in (X, Y)$ is defined as

$$i(x, y) = \log \frac{p(x, y)}{p_X(x)p_Y(y)}.$$

The *mutual information* of X and Y , denoted $I(X, Y)$, is the expected value of the pointwise mutual information, i.e.,

$$I(X, Y) = \sum_{(x, y) \in (X, Y)} p(x, y) i(x, y).$$

For any probability distribution X , the *entropy* of X is defined as $H(X) = -\sum_{x \in X} p(x) \log p(x)$. It is well known that $I(X, Y) = H(X, Y) - H(X|Y) - H(Y|X)$. Hence, $I(X, Y) \leq H(X, Y)$. Moreover, the function

$$D(X, Y) = \frac{H(X, Y) - I(X, Y)}{H(X, Y)} = 1 - \frac{I(X, Y)}{H(X, Y)}$$

is a metric distance [10] with $D(X, X) = 0$ and $D(X, Y) \in [0, 1]$. It is easy to see that $D(X, Y) = 0$ if and only if $H(Y|X) = H(X|Y) = 0$, i.e., distribution X completely determines distribution Y and vice-versa.

Thus, if X and Y are database columns (and we define their joint distribution appropriately), distance D meets our goal for edge weights: it is a dissimilarity measure, i.e., the smaller the D , the closer the columns are. Note that if, e.g., Y is a primary key column, then $H(X|Y) = 0$ and $I(X, Y) = H(X, Y) - H(Y|X)$ is generally larger than if neither X nor Y is a primary key. For a fair comparison, we require all edges to have one endpoint a primary key.

D. ADDITIONAL EXPERIMENTS

Transaction log The TPC-E benchmark comes with 12 transactions [1], including one clean-up transaction, simulating customer and broker interactions with the system and the behavior of real market. As these transactions reflect the usage of the database, we view them as a special form of *query log* for TPC-E. The database-footprint [1] of each transaction lists all the tables and columns involved, as well as the corresponding operations over them. We conduct an analysis on all the database footprints and record the number $n(T)$ of operations performed on each table T over different columns or entire rows. Then $f(T) = n(T) / \sum_{R \in \mathcal{R}} n(R)$. We also analyze the pseudo-code of all transactions and record the number of times a join between a pair of tables R and S is issued. Let $n(R, S)$ be this number. Then $f(R, S) = \frac{n(R, S)}{\sum_{(A, B) \in \mathcal{E}} n(A, B)}$. These values are used to compute the table and join coverage as in Definition 4.

Parameters	TPCE-1	TPCE-2
Number of Customers	1,000	5,000
Initial Trade Days	20	10
Scale Factor	1,000	36,000

Table 1: Parameters of EGen for TPC-E.

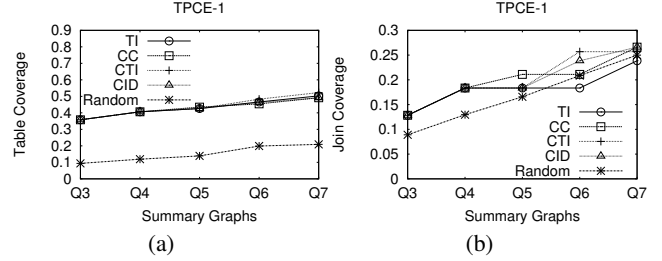


Figure 17: Coverage of summary graphs ($B=0$) for different strategies of choosing query sets.

D.1 Choosing Query Tables

Our previous experiments were conducted over randomly chosen sets of query tables. This simulates the way a large variety of queries would be asked of the system, by users with different needs or expertise.

In this section, we study how the quality of the summary graphs is impacted by several deterministic strategies for choosing the query sets. These choices can be regarded as the system’s recommendations for users who do not have any specific query plans. Rather, such users may want a quick understanding of the most significant functionality of the database. Computing the summary graphs for such system-chosen query sets can be regarded as an extension of the results from [22, 21]. More precisely, we choose the query sets based on the important tables and cluster outputs from [21]. However, by computing summary graphs based on them, we provide significant additional information on the schema.

We studied four deterministic strategies for query table selection, as follows (see [21] for related definitions):

-TI: Select tables in decreasing order of **Table Importance**.

-CC: Select tables in the sequence in which they are chosen as **Cluster Centers**.

-CTI: Group tables into k clusters. In the i th ($1 \leq i \leq Q$) iteration, select the table with current maximum **Table Importance** among clusters with fewer than $\lceil i/k \rceil$ selected tables.

-CID: Group tables into k clusters. In the first k iterations, choose the cluster centers. In the i th ($k + 1 \leq i \leq Q$) iteration, choose the table with current maximum weighted distance to its cluster center, among clusters with fewer than $\lceil i/k \rceil$ selected tables.

We study the effect of these strategies on the coverage and graph complexity of summary graphs, and compare them with the random method. For the latter, we choose 100 random query sets for each given size Q , and report the average measures over the results.

Figure 17 shows the table and join coverage, when the query sets are chosen via the 4 deterministic strategies and the random method. We report the results for $B = 0$, i.e., the quality of the graphs is determined only by the choice of query tables. Note that the 4 deterministic strategies achieve similar table coverage, which is significantly higher than the table coverage of the random method. In fact, the improvement of any deterministic strategy over the random method is more than 100% across the board; e.g., in Figure 17(a) and for $Q = 5$, the table coverage of the random

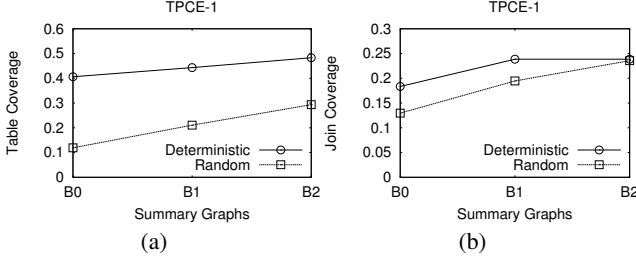


Figure 18: Coverage of summary graphs (Q=4) for different strategies of choosing query sets.

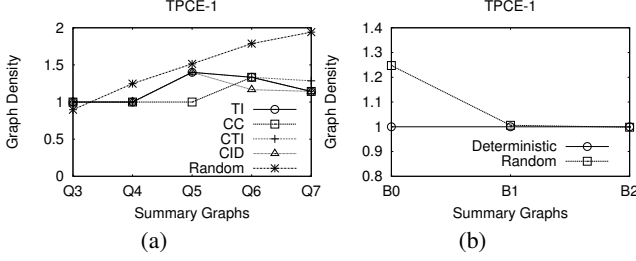


Figure 19: Graph density of summary graphs.

method is less than 20%, while the deterministic methods achieve more than 40%. Notice also that for $Q = 7$, deterministic strategies achieve table coverage of $\approx 50\%$, with summary graphs that contain only $\approx 22\%$ of the tables. We conclude that the deterministic strategies are extremely efficient in selecting the tables with highest query log frequencies.

For the join coverage, the differences between deterministic and random strategies are smaller, with the random method achieving comparable coverage for higher Q .

Finally, we compare the combined effect of query and budget choices. We fix $Q = 4$ and vary B from 0 to 2 (the largest possible value in the deterministic case). The results for all deterministic methods are virtually identical, so we report only one set, dubbed `Deterministic`, in Figure 18. As before, the deterministic methods achieve higher table coverage, as well as join coverage, compared to the random method. Note, however, that the difference in coverage decreases with increasing B . This indicates that the random method benefits from our strategy of choosing budget nodes.

The corresponding graph densities of the summary graphs in these experiments are shown in Figure 19. For $B = 0$, deterministic strategies achieve lower graph complexity than random method, with the differences becoming more significant as Q increases. For fixed $Q = 4$ and varying B , graph densities decrease with increasing B , for the random method. For the deterministic methods, the density is 1 (the smallest possible) for all B . See Figure 19(b).

D.2 Coverage Measures without Query Log

Our definitions of coverage use the frequency of tables and joins in the query log, to arrive at independent quality measures. However, for many databases, such query logs are not available. To compare summary graphs over databases in the absence of logs, we need alternative, instance-based measures.

In this section, we propose a different definition for table coverage, based on the notion of table importance from [21]. In addition, we propose a definition for join coverage based on a combination of

Query Set Choice	TI	CC	CTI	CID	Random
Δ_t	0.041	0.044	0.051	0.039	0.040
Δ_j	0.030	0.018	0.051	0.020	0.053

Table 2: Average differences between Φ and μ over query sets chosen by different strategies (for all $3 \leq Q \leq 10$ and $0 \leq B \leq 10$.)

table importance and mutual information on edges. Clearly, the latter is related to our weight function, so it is not completely independent of our methods for generating summary graphs. Nevertheless, we show that these measures are very close to the corresponding measures based on the query log, for TPC-E. Therefore, they can provide a valid picture on the quality of our outputs, and are a good proxy for the cases when no query logs are available.

Definition 5. Let $\mathcal{G} = (\mathcal{R}, \mathcal{E})$ be a schema graph and $\mathcal{S} = (\mathcal{R}_s, \mathcal{E}_s)$ be a summary graph of \mathcal{G} . Let $\mathcal{I}(T)$ denote the importance of table T (as in [21]). For any edge $(R, S) \in \mathcal{E}$, let $I(R, S)$ and $H(R, S)$ denote the mutual information, resp. the entropy, of the joint distribution along join (R, S) as defined in Section 4.2. We define $\mathcal{I}(R, S) = \frac{\mathcal{I}(R) + \mathcal{I}(S)}{2} \cdot \frac{I(R, S)}{H(R, S)}$.

- (i) The *instance-based table coverage* of \mathcal{S} is $\mu_t = \frac{\sum_{T \leftarrow \mathcal{S}} \mathcal{I}(T)}{\sum_{R \in \mathcal{R}} \mathcal{I}(R)}$.
- (ii) The *instance-based join coverage* of \mathcal{S} is $\mu_j = \frac{\sum_{(R, S) \leftarrow \mathcal{S}} \mathcal{I}(R, S)}{\sum_{(A, B) \in \mathcal{E}} \mathcal{I}(A, B)}$.

The above definition of table coverage is straightforward: the most relevant tables are those with highest table importance (instead of highest query frequency). For the join coverage, the intuition is as follows. It is natural to define the relevance of an edge as proportional to the average relevance of its endpoints, i.e., $\mathcal{I}(R, S) \sim \frac{\mathcal{I}(R) + \mathcal{I}(S)}{2}$. However, we dampen this value by the strength of the connection along the join edge. Note that $\frac{I(R, S)}{H(R, S)} = 1 - D(R, S)$ is a similarity measure, and $0 \leq \frac{I(R, S)}{H(R, S)} \leq 1$ (recall the definition of $D(\cdot, \cdot)$ from Section 4.2).

For any summary graph, we define $\Delta_t = |\Phi_t - \mu_t|$ and $\Delta_j = |\Phi_j - \mu_j|$, where Φ_t and Φ_j are the query log-based measures from Definition 4. Table 2 shows the average Δ_t and Δ_j over all summary graphs computed in Section D.1. All differences are small, both in absolute and in relative terms; i.e., $\Delta_t / \Phi_t \leq 15\%$. Hence, the instance-based coverage measures are a valid alternative in the absence of query logs.

E. MAF WEIGHTS

The following notions were defined in [21]. Let R be a table with tuples τ_1, \dots, τ_n and $e = (R, S)$ be an adjacent join edge. For each τ_i , $fanout_e(\tau_i)$ is the number of tuples in S that τ_i joins with along e . Let q be the number of tuples in R with $fanout_e(\tau_i) > 0$. 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 $maf_e(R) = \frac{\sum_{i=1}^n fanout_e(\tau_i)}{q}$. The *strength* of edge (R, S) was defined as $strength(R, S) = \frac{f_e(R)f_e(S)}{maf_e(R)maf_e(S)}$. It is a similarity measure.

In this paper, we define the *MAF weight* as $wt_{MAF}(R, S) = \log(1/strength(R, S))$, which is a dissimilarity measure. This extends to a distance function in \mathcal{G} that is different from the one in [21], but the shortest paths are identical under both distances: For a path $\pi : A = R_0 - R_1 - \dots - R_\alpha = B$, $wt_{MAF}(\pi) = -\sum_{i=1}^{\alpha} \log(strength(R_{i-1}, R_i))$. Then $d_G(A, B)$ is achieved on the path that maximizes the product $\prod_i strength(e_i)$ over its edges - the same as in [21].