Relational link-based ranking

Floris Geerts [*] Heikki Mannila Evimaria Terzi

Laboratory for Foundations of Computer Science
School of Informatics
University of Edinburgh, UK
fgeerts@inf.ed.ac.uk

Basic Research Unit
Helsinki Institute for Information Technology
Department of Computer Science
University of Helsinki, Finland
{mannila,terzi}@cs.helsinki.fi

Abstract

Link analysis methods show that the interconnections between web pages have lots of valuable information. The link analysis methods are, however, inherently oriented towards analyzing binary relations.

We consider the question of generalizing link analysis methods for analyzing relational databases. To this aim, we provide a generalized ranking framework and address its practical implications.

More specifically, we associate with each relational database and set of queries a unique weighted directed graph, which we call the *database graph*. We explore the properties of database graphs. In analogy to link analysis algorithms, which use the Web graph to rank web pages, we use the database graph to rank partial tuples. In this way we can, e.g., extend the PageRank link analysis algorithm to relational databases and give this extension a random querier interpretation.

Similarly, we extend the HITS link analysis algorithm to relational databases. We conclude with some preliminary experimental results.

[*] Work done while at the Basic Research Unit, Helsinki Institute for Information Technology, Department of Computer Science, University of Helsinki

Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004

1 Introduction

Methods for ranking elements have been widely discussed in a variety of settings. In the context of database systems the motivation for ranking has increased along with the size of databases. In huge databases the users that pose a query would like to see the top-k partial tuples that satisfy their query rather than thousands of tuples ordered in a completely uninformative way. Additionally, the necessity of ranking the query results goes far beyond the functionality of the existing `ORDER BY` operator, which sorts the results only according to the values in the specified attributes. A variety of algorithms that efficiently handle the top-k selection [15, 19] and top-k join queries [20, 24] have been proposed.

Ranking is a notion that has appeared also in the context of Web search applications. The natural need in this context is to rank the web pages returned as a result to a user query. In this case the pages are ranked such that the more relevant the page is to the query, the higher it is ranked. Furthermore, among the web pages that are equally relevant those that are more "important" should precede the less "important" ones. Many ranking algorithms for web pages have been developed ([11, 6, 22, 9, 25]) with the most popular among them being the HITS algorithm proposed by Kleinberg [22] and the PageRank algorithm proposed by Brin et.al [11]. The latter has led to the popular Google search engine.

Web pages are categorical data, and thus the problem of ranking them as such is not trivial since they do not have an intrinsic numerical value on which a ranking could be based on. However, all the ranking algorithms developed for them exploit the hyperlink information, i.e. the structure of the Web graph, in order to assign to each web page a rank value and obtain a ranking based on these values. In contrast to web pages, the assignment of rank values to categorical data in relational databases has not yet been much investigated. In this paper, we do exactly this. More specifically, *we address the problem of automated assignment of rank values to categorical partial tuples.* Based on this assignment we produce useful rankings of partial tuples. We will construct database graphs using queries and try to exploit their structure to obtain rank values.

These rank values can be used in a variety of database applications: First, one can get ranked answers to queries. Second, they can serve as input to the existing top-k algorithms mentioned above. Until now, the top-k algorithms are mainly applied to databases with non-categorical attributes and the top-k algorithms use these values as input. The rank values we obtain for categorical data can be used in a similar way. Finally, the obtained rank values can be helpful in providing ranked keyword search results in relational databases. How exactly the obtained values are going to be used is beyond the scope of this paper. Here we only consider how such rank values can be obtained.

More specifically, we present a general framework for obtaining such rank values for partial tuples of relational databases. The goal is to define those rank scores and find the algorithms to calculate them. For this we exploit information about the interconnections of the partial tuples in the database, as these can be discovered using relational algebra queries.

To obtain rankings for partial tuples we mimic the principles of link analysis algorithms. The well-studied algorithms ([11, 6, 22, 9, 25]) for the Web show that the structure of the interconnections of web pages has lots of valuable information. For example, Kleinberg's HITS algorithm [22] suggests that each page should have a separate "authority" rating (based on the links going *to* the page) and "hub" rating (based on the links going *from* the page). The intuition behind the algorithm is that important hubs have links to important authorities and important authorities are linked by important hubs. Brin's PageRank algorithm [11], on the other hand, calculates globally the PageRank of a web page by considering a random walk on the Web graph and computing its stationary distribution. The PageRank algorithm can also be seen as a model of a user's behavior where a hypothetical web surfer clicks on hyperlinks at random with no regard towards content. More specifically, when the random surfer is on a web page, the probability that he clicks on one hyperlink of the page depends solely on the number of outgoing links the latter has. However, sometimes the surfer gets bored and jumps to a random web page on the Web. The PageRank of a web page is the expected number of times the random surfer visits that page if he would click infinitely many times. Important web pages are ones which are visited very often by the random surfer.

We now rephrase the random surfer in the relational

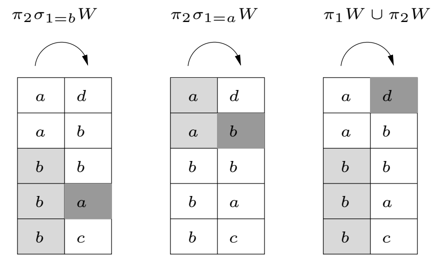

Figure 1: Random walk of random surfer using only 2 kinds of queries.

database setting. Consider the fragment of the Web shown as the binary table W in Figure 1. In the same figure we have shown the surf trail $b \to a \to b \to d$ of the random surfer. In order to walk along the partial tuples (pages) in W, the random surfer needs only two kinds of queries: The first is simply the query which returns all pages present in W. This can be expressed by the expression $\pi_1 W \cup \pi_2 W$. The second kind are queries expressed by $\pi_2\sigma_{1=v}W$, in which v is a page present in W. In other words, these queries ask for all pages reachable from a certain page v. After the random surfer has evaluated one of these queries, he selects a random tuple out of the query result and repeats this procedure again. An important restriction is that while $\pi_1 W \cup \pi_2 W$ may be asked by the random surfer independent of the current page, $\pi_2\sigma_{1=v}W$ may only be asked when the surfer is at page v. In Figure 1 we have shown which queries are asked in order to obtain the shown surf trail.

We use this observation to extend the random surfer model to the *random querier*, which generalizes random-walk based link analysis algorithms by providing the random surfer with a different set of queries at his disposal. Additionally, the model facilitates extensions that allow for using this model for ranking partial tuples.

Seeing the Web as a database allows us to see a hyperlink between two web pages to exist due to queries that connect the two web pages. E.g., in Figure 1 the link between page b and page a can be seen to exists due to the fact that a is in the query result $\pi_2\sigma_{1=b}W$. This idea generalizes to arbitrary databases D and any finite set of queries $\{q_1, \ldots, q_n\}$: There exists a link between two partial tuples $\vec{s}$ and $\vec{t}$ of a database D if there exists an $i = 1, \ldots, n$ such that $\vec{t}$ is in the query result of q_i when evaluated on D and where the selection parameters of q_i are instantiated with constants in $\vec{s}$.

We augment these links with weights relative to some preference function on the queries and frequency information of tuples in the query results. In this way we obtain a weighted directed graph which we call the *database graph*. The database graph is a natural generalization of the graphs used in link analysis.

The database graph enables any graph-based link analysis method to be used for ranking partial tuples. For example, both the PageRank and HITS algorithms can be generalized to operate on the database graph; the generalizations provide tuple ranking algorithms for relational databases.

The contributions of this paper are the following:

- We define the database graph for a given database, set of queries and preference function and explore its theoretical properties.
- We study random walks on the database graph and show that they can be interpreted as the walks of a random querier. We use the stationary distribution of the random walks to assign rank values to partial tuples.
- We show that the random querier generalizes many well-known link analysis algorithms.
- As a second application of the database graph, we extend the HITS algorithm to relational databases and use it to assign rank values as well.
- We experimentally evaluate the use of the obtained rank values to rank query results.

Related work

The problem of assigning rank values to partial tuples in the relational framework is related to the problem of ranking web pages. The latter has been extensively investigated and several link analysis algorithms have been developed for this [11, 6, 22, 9, 25]. Even some unifying frameworks for link analysis algorithms exist [12].

Interesting work on ranking elements in relational databases based on measures from Information Retrieval (IR) is described in [3], however the notion of "link" provided by the queries has not been considered there.

The representation of a database as a graph and link-based ranking appears in the context of keyword search in [5, 18, 2]. The nodes in the graph are the database tuples and the directed relationships between the nodes are induced by foreign key or other constraints. The ranking values are related to the inverse of the path distance between nodes.

Graph representations of databases and random walks on them are considered also in the context of similarity of categorical attributes. Both [26] and [21] construct a graph where the nodes are the constants in the database and two nodes are linked when they appear in the same tuple. They perform different random walks on them in order to obtain a similarity measure for the values. A related iterative approach is the idea of hyperedges connecting tuples based on values [17]. The main difference that we use partial tuples instead of tuples and that we use queries to connect them.

A random walk approach to ranking on (semi-)structured data is proposed in [4]. Although the approach to ranking is very similar to ours, the graph construction is heavily dependent on the presence of (semi-)structured data.

Organization

The rest of this paper is organized as follows. In Section 2 we define databases and query languages. In Section 3, we formally define the database graph and prove some of its properties. We then define the random walk on the database graph and the random querier in Section 4. In Section 5 we extend PageRank and HITS algorithm to relational databases using the database graph. Section 6 describes some experimental results. We conclude the paper in Section 7.

2 Preliminaries

We refer to [1, 16] for a more detailed description of basic database notions. For simplicity of exposition, we assume that the database schema S consists of a single relation name R of arity n. However, all definitions and results generalize directly to arbitrary database schemas.

Let $\mathcal{D}$ be a database instance over S. The active domain of $\mathcal{D}$, denoted by $adom(\mathcal{D})$, consists of all constants in $\mathcal{D}$. For a tuple $\vec{t} \in \mathcal{D}$ of size n, we denote the value of its i-th attribute by $t_i \in adom(\mathcal{D})$. The active domain of a tuple $\vec{t} \in \mathcal{D}$, denoted by $adom(\vec{t})$, is the set $\{t_1, \ldots, t_n\}$.

The standard query language is the relational algebra, or equivalently the relational calculus, over the database schema S. We denote this query language by RA. Relations and queries are interpreted using the bag semantics, i.e., duplicate tuples are allowed. The reason for this is that we need the notion of frequency which disappears if we do not allow for duplicates. We will not distinguish between queries and the RA expressions expressing them. We denote the query result of q on $\mathcal{D}$ by $q(\mathcal{D})$.

Let $q \in$ RA be an n-ary query and denote the set of attributes in the query result by I. We will partition I in *source attributes* $\vec{x}$ and the *target attributes* $\vec{y}$. We always assume that this partition is specified for each query q we encounter. We make this explicit by writing $q(\vec{y}|\vec{x})$ instead of simply q.

Let $\vec{s} \in adom(\mathcal{D})^k$ where $k = |\vec{x}|$ and let $\ell = |\vec{y}|$. Then we define the RA expression

$$q(\vec{y}|\vec{s}) \equiv \pi_{y_1,\ldots,y_\ell}\sigma_{x_1=s_1,\ldots,x_k=s_k} q(\vec{y}|\vec{x}).$$

We will denote the query result of $q(\vec{y}|\vec{s})$ on $\mathcal{D}$ by $q(\mathcal{D}, \vec{s})$.

We extend the RA with the duplicate elimination operator δ for transforming bags into sets if necessary.

Given a tuple $\vec{s}$ and a query $q \in$ RA the support of $\vec{s}$ in $q(\mathcal{D})$, denoted by $\mathrm{supp}(\vec{s}, q(\mathcal{D}))$, is the number of times $\vec{s}$ appears in $q(\mathcal{D})$. The frequency of $\vec{s}$ in $q(\mathcal{D})$ is defined as $\mathrm{freq}(\vec{s}, q(\mathcal{D})) = \frac{\mathrm{supp}(\vec{s}, q(\mathcal{D}))}{|q(\mathcal{D})|}$, where $|q(\mathcal{D})|$ denotes the size of $q(\mathcal{D})$.

3 The database graph

As already mentioned in the Introduction, one can consider the web as a database $\mathcal{D}$ over a binary relation W. Then following a hyperlink from a page v can be seen as first querying the database using the query $q(y|x) \equiv W(x, y)$, and then selecting a page out of $q(\mathcal{D}, v)$. Two web pages v and w are now linked by the query q iff $w \in q(\mathcal{D}, v)$. We generalize this idea to arbitrary databases and queries.

Definition 1 (Link). For a given database $\mathcal{D}$ and query language $\mathcal{L} \subseteq \mathrm{RA}$, a tuple $\vec{s} \in \mathit{adom}(\mathcal{D})^k$ is $\mathcal{L}$-*linked* to a tuple $\vec{t} \in \mathit{adom}(\mathcal{D})^\ell$ iff there exists a query $q(\vec{y}|\vec{x}) \in \mathcal{L}$ such that $|\vec{x}| = k$, $|\vec{y}| = \ell$, and $\vec{t} \in q(\mathcal{D}, \vec{s})$. □

From now on we assume that $\mathcal{L}$ consists of a finite number of queries.

Let $M = \langle \mathcal{D}, \mathcal{L}, f \rangle$ where $\mathcal{D}$ is a database, $\mathcal{L} \subseteq \mathrm{RA}$, and f is some preference function $f : \mathcal{L} \to \mathbb{Q}^+$. Here, $\mathbb{Q}^+$ denotes the set of strictly positive rational numbers.

We now define the database graph. The definition is rather technical but the intuition behind it is very natural. Indeed, the vertices of the database graph correspond to the active domain of tuples in the answers to queries in $\mathcal{L}$. The reason why we work with the active domains instead of the tuples themselves is that a constant appearing in some attribute can possibly be used in other attributes as well. So instead of storing a constant for each possible attribute separately, we store it only once. This slightly complicates the formal definition (see below) of database graph since many different tuples can correspond to the same vertex. The edge relation is based on Definition 1. Finally, we will assign weights to the edges corresponding to the preferences of the queries establishing this edge (or link) and the support of the tuples consistent with the target vertex in the query results. More formally,

Definition 2 (Database graph). Given $M = \langle \mathcal{D}, \mathcal{L}, f \rangle$ the corresponding *database graph* is the weighted directed graph $G_M = (V_M, E_M, \lambda_M)$ where,

- The set of vertices V_M is constructed as follows: For each query $q(\vec{y}|\vec{x}) \in \mathcal{L}$ we instantiate the parameters $\vec{x}$ with tuples $\vec{s} \in \mathit{adom}(\mathcal{D})^k$, where $k = |\vec{x}|$. For each $\vec{t} \in q(\mathcal{D}, \vec{s})$, we add the vertex $v = \mathit{adom}(\vec{t})$ to V_M, if not already included. Note that v is a set of constants. Thus, V_M is
$$\big\{\mathit{adom}(\vec{t}) \mid q(\vec{y} \mid \vec{x}) \in \mathcal{L}, |\vec{x}| = k \quad \vec{s} \in \mathit{adom}(\mathcal{D})^k, \vec{t} \in q(\mathcal{D}, \vec{s})\big\}.$$
For a vertex $v \in V_M$, we denote by v^k the set of all k-tuples formed from constants in v.
- The set of edges E_M is equal to all ordered pairs of vertices (v, w) such that there exists a tuple $\vec{s} \in v^k$ which is $\mathcal{L}$-linked to a tuple $\vec{t} \in \mathit{adom}(\mathcal{D})^\ell$ such that $w = \mathit{adom}(\vec{t})$; and
- The weight function $\lambda_M : E_M \to \mathbb{Q}^+$ is defined as $\lambda_M(v, w) =$
$$\sum_{q(\vec{y}|\vec{x}) \in \mathcal{L}} f(q)\Big(\sum_{\substack{\vec{s} \in v^k, k=|\vec{x}| \\ \vec{t} \in w^\ell, \ell=|\vec{y}|}} \mathrm{freq}(\vec{t}, q(\mathcal{D}, \vec{s}))\Big).$$

□

We illustrate the concept of database graph by the following examples.

Example 1. Let $\mathcal{D}$ be the database given by the table in Figure 2. The language $\mathcal{L}$ consists of the queries $q_1(y|x) \equiv \pi_{1,2}R(x, y, z)$ and $q_2(y, z|x) \equiv R(x, y, z)$. Then for any constant a appearing in the first attribute $q_1(\mathcal{D}, a)$ equals $\{b \mid (a, b) \in q_1(\mathcal{D})\}$. Similarly, for any constant a, $q_2(\mathcal{D}, a)$ consists of the pairs $\{(b, c) \mid (a, b, c) \in q_2(\mathcal{D})\}$. This shows that q_1 will link the first attribute to the second one, while q_2 links the first attribute to the second and third one, as can be seen in Figure 2. We define the preference function as $f(q_1) = f(q_2) = 1$. The complete database graph is shown in Figure 2. E.g., the weight on the edge from $\{v_2\}$ to $\{t_2, v_3\}$ is equal to $f(q_2)\mathrm{freq}((t_2, v_3), q_2(\mathcal{D}, v_2)) = 1$. □

When we disregard the weights, another example is the Gaifman graph of finite model theory [13].

Example 2. Let $\mathcal{D}$ be a database over an n-ary relation R. Consider the language $\mathcal{L}$ consisting of $q_{i,j}(x_j|x_i) \equiv \pi_{i,j}R(x_1, \ldots, x_n)$ and $f(q_{i,j}) = 1$ for all $i, j = 1, \ldots, n$. The database graph has as vertices the constants in $\mathit{adom}(\mathcal{D})$ and there is an edge between two constants iff they appear in the same tuple in $\mathcal{D}$. □

The database graph is a well-defined object. Indeed, we call $\langle \mathcal{D}, \mathcal{L}, f \rangle$ and $\langle \mathcal{D}', \mathcal{L}, f \rangle$ isomorphic, denoted by $\langle \mathcal{D}, \mathcal{L}, f \rangle \cong \langle \mathcal{D}', \mathcal{L}, f \rangle$, if there exists a bijection $b : \mathit{adom}(D) \to \mathit{adom}(D')$ such that for all $q(\vec{y}|\vec{x}) \in \mathcal{L}$ and $\vec{s} \in \mathit{adom}(\mathcal{D})^k$ for $k = |\vec{x}|$, we have for any $\vec{t} \in q(\mathcal{D}, \vec{s})$ that
$$\mathrm{freq}(\vec{t}, q(\mathcal{D}, \vec{s})) = \mathrm{freq}(b(\vec{t}), q(\mathcal{D}', b(\vec{s}))),$$
where b is extended to tuples $\vec{x}$ as $b(\vec{x}) = (b(x_1), \ldots, b(x_k))$.

Theorem 1. If $M = \langle \mathcal{D}, \mathcal{L}, f \rangle$ and $N = \langle \mathcal{D}', \mathcal{L}, f \rangle$ such that $M \cong N$, then G_M is isomorphic to G_N.

$\mathcal{D} =$

v_1	v_2	t_1
v_1	v_3	t_1
v_1	v_4	t_1
v_2	v_3	t_2
v_4	v_1	t_2
v_4	v_3	t_2

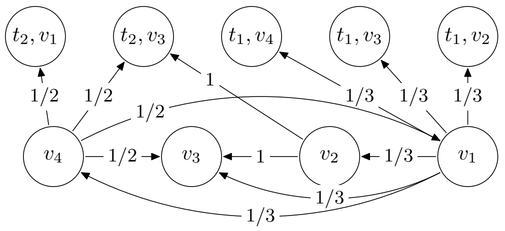

Figure 2: The database $\mathcal{D}$ (left) and the database graph G_M of $M = \langle \mathcal{D}, \mathcal{L}, f \rangle$ of Example 1 (right).

Proof. We refer for the proof to the full paper. □

We also have a monotonicity property with respect to taking sub-languages.

Theorem 2. If $M = \langle \mathcal{D}, \mathcal{L}, f \rangle$ and $N = \langle \mathcal{D}, \mathcal{L}', f' \rangle$ such that $\mathcal{L}' \subseteq \mathcal{L}$ and $f(q) = f'(q)$ for any $q \in \mathcal{L}'$, then G_N is isomorphic to a subgraph of G_M.

Proof. The proof is analogous to the proof of Theorem 1. □

In general, the reverse of Theorem 1 is not true as can be seen from the following example.

Example 3. Consider the databases $\mathcal{D}$ and $\mathcal{D}'$ shown in Figure 3. Here, different symbols denote different constants. Let $\mathcal{L}$ consist of the queries

$$\begin{aligned} q_0(x|) &\equiv \pi_1 R(x, y, z, u, v), \\ q_1(y|x) &\equiv \pi_{1,2}\sigma_{3=5} R(x, y, z, u, v), \\ q_2(y|x) &\equiv \pi_{1,2}\sigma_{3=4} R(x, y, z, u, v), \\ q_3(y|x) &\equiv \pi_{1,2}\sigma_{3\neq 4} R(x, y, z, u, v). \end{aligned}$$

The preference function f assigns weight 1 to each query. It is easily verified that the graphs G_M and G_N are isomorphic and correspond to the graph shown in Figure 3. However, there is no bijection making $\langle \mathcal{D}, \mathcal{L}, f \rangle$ and $\langle \mathcal{D}', \mathcal{L}, f \rangle$ isomorphic. Indeed, from $q_1(\mathcal{D}, s)$ and $q_1(\mathcal{D}', s)$ the bijection b should map $b(t_1) = t_1$, while from $q_2(\mathcal{D}, s)$, $q_2(\mathcal{D}', s)$, $q_3(\mathcal{D}, s)$ and $q_3(\mathcal{D}', s)$ it follows that $b(t_1) = t_2$. □

The database graph is defined without taking into account any semantic relationships between attributes or additional schema constraints. However, this can be easily incorporated in the queries used in the language $\mathcal{L}$.

In the next section we define a random walk on the database graph. In order for the random walk to have nice convergence properties (see the next section), the underlying graph should be strongly connected and non-bipartite. This property turns out to be undecidable.

Theorem 3. Given a query language $\mathcal{L}$, it is undecidable whether the database graph is strongly connected and non-bipartite for all $\mathcal{D}$ and preference functions f.

Proof. First, we remark that the topology of the graph is independent of f. So, we can disregard the preference function in what follows. We use a reduction to the undecidability of satisfiability of relational algebra expressions on binary relations [8]. We construct for each $\mathbf{q}(x_1, \ldots, x_k) \in \mathrm{RA}$ the language $\mathcal{L} = \{q_1, q_2, q_3\}$ where,

$$\begin{aligned} q_1(u, z|x, y) &\equiv \texttt{if}(\exists x_1 \cdots \exists x_k \mathbf{q}(x_1, \ldots, x_k|)) \\ &\quad\ \texttt{then} \\ &\quad\ \sigma_{1\neq 2 \wedge 1=3 \wedge 2=4} R(x, y) \times R(u, z) \\ q_2(y|) &\equiv \texttt{if}(\exists x_1 \cdots \exists x_k \mathbf{q}(x_1, \ldots, x_k|)) \\ &\quad\ \texttt{then}\ \ \pi_2 R(x, y) \\ q_3(z|) &\equiv \texttt{if not}(\exists x_1 \cdots \exists x_k \mathbf{q}(x_1, \ldots, x_k|)) \\ &\quad\ \texttt{then}\ \ \pi_1 R(z, u) \cup \pi_2 R(u, z) \end{aligned}$$

By construction, for any $\mathcal{D}$ and f, the database graph associated with $\langle \mathcal{D}, \mathcal{L}, f \rangle$ will be connected and non-bipartite iff $\mathbf{q}$ is not satisfiable.

Indeed if $\mathbf{q}$ is not satisfiable then $\mathcal{L}$ collapses to q_3. For any $\mathcal{D}$ and f, the database graph associated with $\langle \mathcal{D}, q_3, f \rangle$ is the complete graph with vertex set $adom(\mathcal{D})$. This is clearly always a strongly connected and non-bipartite graph.

For the other direction, suppose that there exists $\mathcal{D}$ and f such that the graph associated with $\langle \mathcal{D}, \mathcal{L}, f \rangle$ is disconnected or bipartite. We need to show that this implies that on $\mathcal{D}$ the query $\mathbf{q}$ is satisfiable. Therefore, we show that for any $\mathcal{D}$ and f, the database graph associated with $\langle \mathcal{D}, \{q_1, q_2\}, f \rangle$ is disconnected. W.l.o.g., we may assume that $\mathcal{D}$ only consists of tuples (s, t) such that $s \neq t$. Indeed, if $|adom(\mathcal{D})| > 1$ (The case when $|adom(\mathcal{D})| = 1$ can be disregarded), applying first the query $\pi_{14} R \times R$ ensures that $\mathcal{D}$ always contains (s, t) with $s \neq t$. We then select only those pairs (s, t) from $\mathcal{D}$ such that $s \neq t$. So, the database graph will be not connected because there is an edge in the database graph from vertex $\{s, t\}$ to vertex $\{t\}$ by q_2, but no edge exists from $\{t\}$ to $\{s, t\}$. This is because q_1 only links $\{t\}$ to vertex $\{t, t\}$, which is by construction not in $\mathcal{D}$. □

4 Random walks on databases

Let $G = (V, E, \lambda)$ be a weighted directed graph. We next define the random walk on this graph, and then show how the concept applies to database graphs.

$\mathcal{D} =$

s	t_1	α	α	α
s	t_1	α	γ	α
s	t_1	α	γ	α
s	t_2	β	γ	α
s	t_2	β	β	α
s	t_2	β	β	α

$\mathcal{D}' =$

s	t_1	α	α	α
s	t_1	α	α	α
s	t_1	α	γ	α
s	t_2	β	γ	α
s	t_2	β	γ	α
s	t_2	β	β	α

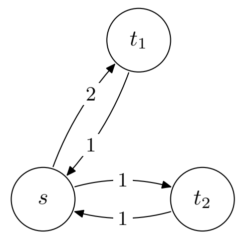

Figure 3: Two non-equivalent databases (left) giving the same database graph (right) for $\mathcal{L}$ of Example 3.

Definition 3 (RW on a graph, [7]). A simple random walk on G is the following random process: Start in a randomly selected vertex $v \in V$. Next, jump to an adjacent vertex w with probability $\lambda(v,w)/\left(\sum_{(v,w')\in E}\lambda(v,w')\right)$. This is then repeated starting from vertex w. $\square$

A random walk on G can also be seen as a Markov chain with state space V where the transition probabilities are represented by a stochastic[1] $|V| \times |V|$-matrix $\mathbf{P}_G = (P_{vw})$, where $P_{vw} = \lambda(v,w)/\left(\sum_{(v,w')\in E}\lambda(v,w')\right)$.

Theorem (Fundamental Theorem of Markov Chains, [23]). If G is strongly connected and non-bipartite, then the Markov chain given by $\mathbf{P}_G$ has the following properties. There exists a unique stationary distribution $\vec{p}$, i.e., $\vec{p} = \vec{p}\mathbf{P}_G$ and $\sum_i p_i = 1$. Moreover, let $N(v,k)$ be the number of times the Markov chain visits v in k steps, then

$$\lim_{k\to\infty}\frac{N(v,k)}{k} = p_v.$$

$\square$

The stationary distribution is a description of the steady-state behavior of the Markov Chain. The stationary distribution will be used to obtain rank values for partial tuples in the next section.

Definition 4 (RW on database). The random walk on $M = \langle\mathcal{D},\mathcal{L},f\rangle$ is the simple random walk on the database graph G_M. $\square$

We now define the random querier. When in certain vertex of the database graph, the random querier will select a query compatible with the vertex in which he currently is. The probability of selecting such query depends on a given preference function. Once the query is asked a random tuple is selected as input parameter for the query and a random tuple is selected from the output. We make this more formal in what follows.

Let $s \subseteq \mathit{adom}(\mathcal{D})$. We denote by $\mathcal{L}_s$ all queries $q(\vec{y}|\vec{x}) \in \mathcal{L}$ for which there exists $\vec{s} \in s^k$ for $k = |\vec{x}|$ such that $q(\mathcal{D},\vec{s})$ is nonempty. For a query $q(\vec{y}|\vec{x}) \in \mathcal{L}$ and $s \subseteq \mathit{adom}(\mathcal{D})$, let $\Gamma(q,s)$ be the set of tuples $\vec{s} \in s^k$ for $k = |\vec{x}|$ such that $q(\mathcal{D},\vec{s})$ is nonempty. Moreover, let $\gamma(q,s) = |\Gamma(q,s)|$.

Definition 5 (Random Querier). The $(\mathcal{L},f)$-random querier on $\mathcal{D}$ is the following random process: An initial element s is selected randomly from the set of vertices V_M from the database graph. Next, a query q is chosen from $\mathcal{L}_s$ with probability

$$\gamma(q,s)f(q)/(\sum_{q'\in\mathcal{L}_s}\gamma(q',s)f(q')),$$

and a tuple $\vec{s} \in \Gamma(q,s)$ is chosen uniformly at random. Finally, a tuple $\vec{t}$ is selected randomly from $q(\mathcal{D},\vec{s})$. This is then repeated starting from $\mathit{adom}(\vec{t})$. $\square$

Theorem 4. The random walk performed by the $(\mathcal{L},f)$-random querier on $\mathcal{D}$ is the same random walk as the random walk on the database graph G_M of $M = \langle\mathcal{D},\mathcal{L},f\rangle$.

Proof. We refer for the proof to the full paper. $\square$

Example 4. *A well-known example of a random walk is the random surfer introduced by Brin [10]. The random surfer is the same as the $(\mathcal{L},f)$-random querier on the Web database $\mathcal{D}$, with $\mathcal{L} = \{q_1(y|x) \equiv W(x,y), q_2(x|) \equiv \delta(\pi_1 W(x,y) \cup \pi_2 W(y,x))\}$ and $f(q_1) = 1-p$ and $f(q_2) = p$. Let $V = \mathit{adom}(\mathcal{D})$. The transition matrix $\mathbf{P} = (P_{vw})$ of the random surfer is given by*

$$P_{vw} = \begin{cases} \frac{p}{|V|} + \frac{(1-p)}{\text{outdeg}(v)} & \textit{if } (v,w) \in \mathcal{D} \\ \frac{p}{|V|} & \textit{otherwise.} \end{cases}$$

In the database graph corresponding to $\langle\mathcal{D},\mathcal{L},f\rangle$ we have an edge (v,w) for any pair of vertices. The weight of an edge (v,w) is given by the sum of

$$f(q_2)\text{freq}(w, q_2(\mathcal{D})) = p\frac{1}{|V|},$$

and $f(q_1)\text{freq}(w, q_1(\mathcal{D},v))$ which is equal to

$$\begin{cases} \frac{1-p}{\text{outdeg}(v)} & \textit{if } (v,w) \in \mathcal{D} \\ 0 & \textit{otherwise.} \end{cases}$$

Hence, the $(\mathcal{L},f)$-random querier on $\mathcal{D}$ has the same transition matrix as the one of the random surfer.

[1]A stochastic matrix is a matrix in which for each row the elements in the row sum up to one.

Also other link analysis algorithms fit perfectly in the random queries framework.

Example 5 (sHITS, [6]). The stochastic HITS algorithm has as transition matrix

$$P_{vw} = \frac{|\{x \in V | (x,v) \in E \wedge (x,w) \in E\}|}{\sum_{y \in V} |\{x \in V | (x,v) \in E \wedge (x,y) \in E\}|}.$$

A simple computation shows that the $(\mathcal{L}, f)$-random querier with $\mathcal{L} = \{q(z|x) \equiv \pi_{2,4}\sigma_{1=3}W(u,x) \times W(y,z)\}$ and $f(q) = 1$ results in the same random walk.

We are primarily interested in the stationary distribution of the random walks. By the Fundamental Theorem of Markov Chains, this can be obtained by computing the matrix and solving the eigenvector problem. However, for undirected graphs we have a closed form expression for the stationary distribution.

Theorem ([23]). Let $G = (V, E)$ a connected, non-bipartite, undirected and un-weighted graph and let $m = |E|$. Then the stationary distribution $(p_v)_{v \in V}$ of the simple random walk on G is given by $\deg(v)/2m$.

We have the following result.

Theorem 5. It is decidable for a given weighted directed graph $G = (V, E, \lambda)$, whether there exists an undirected multi-graph $G_u = (V_u, E_u)$ such that the simple random walks on G and G_u have the same transition matrix. Moreover, the graph G_u can be computed, if it exists.

Proof. (Sketch) Note that weighted directed graph G can be transformed in a graph with integer weights by the multiplying for each vertex V all weights of edges starting in v with the least common multiplier (l.c.m) of the denominators of weights of the edges starting in v. So, we may assume that G has integer weights. Also note that for each node $v \in V$ we are allowed to multiply the weights of all outgoing edges by the same integer n_v without affecting the random walk. We get rid of the integer weights by replacing each edge (v, w) with integer weight k, by k edges (v, w) of weight 1. We abuse notation and call this edge set also E. So, in order to decide whether G can be replaced by an undirected multigraph we need to check whether there exist integers (n_v) with $v \in V$ such that the indegree becomes equal to the outdegree for every vertex v, or for every v,

$$\sum_{w:(v,w) \in E} n_v \lambda(v, w) = \sum_{w:(w,v) \in E} n_w \lambda(w, v). \quad (1)$$

This can be decided using standard integer programming techniques [27]. So the answer to the decision problem is yes iff there exists a solution to equation (1). In case there exists a solution, we define $G_u = (V_u, E_u)$ as $V_u = V$ and E_u contains an undirected edge (v, w) for every pair of edges (v, w) and (w, v) in E. □

The importance of the previous result is that before starting computing the stationary distribution of a random walk, one can decide whether the walk corresponds to a walk on an undirected graph. In this case the stationary distribution can be computed much more efficiently, using the Fundamental Theorem of Markov Chains.

For some languages it can be shown that there exists integers n_v such that Equation (1) holds for any $\mathcal{D}$.

Example 6. Consider the database $\mathcal{D}$ consisting of a single relation R, and let $\mathcal{L} = \{q_{i,j}(x_j|x_i) \equiv \pi_{i,j}R(x_1, \ldots, x_n) \mid 1 \leq i \leq n, 1 \leq j \leq n\}$. All queries have preference 1. Let G be the database graph. Then there exists an undirected graph G_u satisfying the property stated in Theorem 5. Indeed, for each s and each t such that $t \in q_{i,j}(\mathcal{D}, s)$ we have an edge (s, t) of weight $\text{freq}(t, q_{i,j}(\mathcal{D}, s))$. The l.c.m. of the denominators for all outgoing edges from s is $|q_{i,j}(\mathcal{D}, s)|$, so we get the integer weight

$$\begin{aligned}\lambda(s,t) &= \text{freq}(t, q_{i,j}(\mathcal{D}, s))|q_{i,j}(\mathcal{D}, s)| \\ &= |\{\vec{u} \in \mathcal{D} \mid u_i = s \wedge u_j = t\}|.\end{aligned}$$

We get the same integer weight for the edge (t, s), so $\lambda(s,t) = \lambda(t,s)$ and hence Equation (1) holds for $n_s = 1$ for all s. This reasoning is clearly independent from $\mathcal{D}$. □

5 Rank algorithms

In this section we describe two methods for obtaining rank values for partial tuples. Both are based on eigenvector computations. The first one, RelWalk, is based on the stationary distribution of a random walk on the database graph similarly to PageRank. The second, RelHITS, uses the mutual reinforcement technique of HITS. Therefore, the assignement of rank values is based on the normalized principal eigenvector of a matrix associated to a certain subgraph of the database graph. Both algorithms output rank values of partial tuples which can serve either directly as a ranking of query results, or as input for top-k selection and join algorithms.

5.1 RelWalk

The RelWalk algorithm takes as input the database $\mathcal{D}$, a language $\mathcal{L}$ and preference function f, and computes the rank values for subsets $s \subseteq adom(\mathcal{D})$. The rank value of s corresponds to the value p_s in the stationary distribution $\vec{p}$ of the random walk on the database graph G_M of $M = \langle \mathcal{D}, \mathcal{L}, f \rangle$. The Fundamental Theorem of Markov Chains says that p_s gives the probability that the $(\mathcal{L}, f)$-random querier on $\mathcal{D}$ visits s, given that he was allowed to ask the queries in $\mathcal{L}$ for infinite long time. Intuitively, frequently visited states are regarded as more important. We compute the stationary distribution $\vec{p}$ by solving the eigenvector problem $\vec{p}\mathbf{P} = \vec{p}$ where $\mathbf{P}$ is the transition matrix of the random walk.

The database graph must be strongly connected and non-bipartite in order for the stationary distribution to exist. Of course, not every database graph has this property. However, we can alter the query language such that we always end up with a strongly connected and non-bipartite database graph. Indeed, we simply add queries of the form $q(x_{i_1}, \ldots, x_{i_k}|) \equiv \pi_{i_1,\ldots,i_k} R$ where the projections are chosen such that no new vertices are introduced in the database graph. The database graph is now strongly connected since all vertices are connected to each other. It is also non-bipartite since all vertices have a self-loop. Note that the PageRank algorithm uses the same adaptation by adding the query $q_2(x|) \equiv \pi_1 R(x,y) \cup \pi_2 R(y,x)$.

5.2 RelHITS

In contrast to RelWalk, RelHITS is query dependent. Therefore, RelHITS algorithm takes as input $M = \langle \mathcal{D}, \mathcal{L}, f\rangle$ and an imposed query q. The algorithm considers the database graph G_M and selects the subgraph $G'_M = (V'_M, E'_M, \lambda'_M)$ where

$$E'_M = \{(v,w) \in E \mid w \subseteq adom(Q(q,\mathcal{D}))\},$$

and V'_M consists only of nodes connected by edges in E'_M. The weight function λ'_M is the restriction of λ_M to E'_M. From the graph G'_M we form the $m \times n$ matrix $\mathbf{Q} = (Q_{vw}) = (\lambda'_M(v,w))$, where $m = |H = \{v \in V'_M \mid \exists w \in V'_M (v,w) \in E'_M\}|$ and $n = |A = \{w \in V'_M \mid \exists v \in V'_M (v,w) \in E'_M\}|$. The elements in the sets H and A can be thought of as the hubs and the authorities in the context of the HITS algorithm and therefore RelHITS scores h_j and a_j are computed iteratively as follows.

$$\begin{cases} h_j^t & \leftarrow & \sum_{(j,i)\in E'_M} \lambda'_M(j,i) h_i^{(t-1)} \\ a_j^t & \leftarrow & \sum_{(i,j)\in E'_M} \lambda'_M(i,j) a_i^{(t-1)} \end{cases} \quad (2)$$

The main idea is that important hubs are related to important authorities and vice versa. Moreover, the update schema (2) converges to the principal eigenvector $\vec{h}$ of $\mathbf{Q}\mathbf{Q}^T$ for the hub scores, while the authority scores converge to the principal eigenvector $\vec{a}$ of $\mathbf{Q}^T\mathbf{Q}$ ([22]). The RelHITS normalizes these eigenvectors and outputs them.

6 Experimental evaluation

In this section we describe our implementation for constructing the database graph and obtaining rank values of partial tuples. We give the setup of our experiments and present the corresponding results.

We implemented the database graph construction on top of the Postgres relational database management system. JDBC has been used for connecting to the database system. We ran the RelWalk and the RelHITS algorithm on the bibliography database [2]. This database $\mathcal{D}$ consists of a single relation R with attributes `paper_title`, `author`, `conference`, and `year`. There are 7 677 tuples in the database, 3 062 unique paper titles and 4 203 unique authors.

The main goal of the experiments is to show that for different query languages there are different rankings obtained. These rankings are closely related to the queries used for the construction of the database graph and in all the cases have a meaningful interpretation in terms of these queries. The experiments also show the flexibility of our general framework. Any weighted set of queries can be used to construct a database graph from $\mathcal{D}$. This raises the question which queries should be used and this is a very challenging problem to explore indeed. Computing the rank values of partial tuples for a given query language is only preprocessing step. Once the rank values are computed they can be used for ranking tuples in the answer of queries imposed on the database. We illustrate this for simple query languages and queries in the next sections.

6.1 Experimental setup

For the purpose of the experiments we have constructed the database graphs and obtained partial tuple rankings using both the RelWalk and RelHITS algorithms. For each one of the algorithms we constructed the corresponding database graphs using two different query languages, namely $\mathcal{L}_1$ and $\mathcal{L}_2$ for RelWalk and $\mathcal{L}'_1$ and $\mathcal{L}'_2$ for RelHITS. The languages were selected in such a way that $\mathcal{L}_1$ (and $\mathcal{L}_2$) is expected to show similar rankings to $\mathcal{L}'_1$ (and $\mathcal{L}'_2$).

In the sequel we show the ranked outputs (along with the rank values) we obtained when the following two queries were imposed to the database: $q(x_2|) \equiv \pi_2 R$ and $q'(x_2|) \equiv \pi_2\sigma_{6=\text{'H. Garcia-Molina'}}\sigma_{1=5} R \times R$. Query q is a simple projection on all the authors of the database, while query q' is a projection on all the authors of the database that are co-authors of "H. Garcia-Molina". The selection of q and q' is made mainly for two reasons. First, their output consists of partial tuples already assigned a rank value from our ranking algorithms and thus we do not need to employ any other additional procedure for ranking aggregates. Second, the output of the queries demonstrates the different features of the proposed ranking algorithms.

Query languages for RelWalk

The first language used for obtaining RelWalk rankings was $\mathcal{L}_1 = \{q_1(x_j|x_i), q_2(x|)\}$ where $q_1(x_j|x_i) \equiv \pi_{i,j} R(x_1,x_2,x_3,x_4)$ with $i \neq j$ and preference $f_1 = 0.9$ and $q_2(x|) \equiv \pi_1 R \cup \pi_2 R \cup \pi_3 R \cup \pi_4 R$ with preference $f_2 = 0.1$. The intuition behind $\mathcal{L}_1$ is exactly what one

[2]The data set is available at `http://liinwww.ira.uka.de/bibliography/`

expects the random querier to do when he has the freedom to do a random walk on partial tuples of size one. Query q_2 makes sure that the constructed database graph is strongly connected and non-bipartite. Therefore, there is an underlying stationary distribution. Additionally, notice that for $\mathcal{L}_1$ we can apply the result of Example 6 and we can show that the degree of nodes in the corresponding undirected graph, and hence the stationary distribution, are frequency related. The database graph constructed by RELWALK when language $\mathcal{L}_1$ was considered consists of a total number of 7 294 nodes (partial tuples of size 1) and 50 434 edges (relational links taken into consideration for obtaining the ranking) when query q_1 is considered. The query q_2 makes the graph a complete graph.

The second language used for obtaining RELWALK rankings was $\mathcal{L}_2 = \{q_1(x_j|x_i), q_2(x_2|x_6), q_3(x|)\}$ where $q_1(x_j|x_i) \equiv \pi_{i,j}R(x_1, x_2, x_3, x_4)$ with $i \neq j$ and preference $f_1 = 0.45$, $q_2(x_2|x_6) \equiv \pi_{2,6}\sigma_{1=5,2\neq 6}R \times R$ with preference $f_2 = 0.45$, and finally $q_3(x|) \equiv \pi_1 R \cup \pi_2 R \cup \pi_3 R \cup \pi_4 R$ with preference $f_3 = 0.1$. For this language the results of Example 6 do not apply and thus we expect the obtained rankings to be not related to frequency. This is due to the co-author query q_2 that has been included in the language. The database graph constructed using language $\mathcal{L}_2$ has 7 294 nodes and 68 012 edges when queries q_1 and q_2 are only considered. Query q_3 again in this case makes the graph a complete graph.

Query languages for RELHITS

The two languages used for obtaining the RELHITS rankings are $\mathcal{L}'_1$ and $\mathcal{L}'_2$ and are selected such that they have similar flavor to $\mathcal{L}_1$ and $\mathcal{L}_2$ used for RELWALK. This means that they are expected to give similar rankings. Language $\mathcal{L}'_1 = \{q_1(x_2|x_3), q_2(x_2|x_4)\}$ consists of $q_1(x_2|x_3) \equiv \pi_{2,3}R(x_1, x_2, x_3, x_4)$ with preference $f_1 = 0.5$ and $q_2(x_2|x_4) \equiv \pi_{2,4}R(x_1, x_2, x_3, x_4)$ with preference $f_2 = 0.5$. The intuition behind $\mathcal{L}'_1$ is that, as in $\mathcal{L}_1$, partial tuples of size 1 and direct links between them are again included in the database graph. Language $\mathcal{L}'_2 = \{q_1(x_2|x_3, x_6)\}$ on the other hand consists of a single query $q_1(x_2|x_3, x_6) \equiv \pi_{2,3,6}\sigma_{2\neq 6,1=5}R \times R$ with preference equal to 1. This apparently is the co-author query so that there is a relationship between $\mathcal{L}'_2$ and $\mathcal{L}_2$ and thus the comparison between the obtained rankings makes sense. For the case of RELHITS the size of the constructed database graph depends not only on the query languages ($\mathcal{L}'_1$ and $\mathcal{L}'_2$) used for the graph construction, but also on the imposed queries q and q' the results of which we want to rank. So in the case of $\mathcal{L}'_1$ and query q the constructed database graph consists of 4 232 nodes and 11 375 edges, while for the same language but for query q' the corresponding graph of RELHITS consists of only 86 nodes and 171 edges. For language $\mathcal{L}'_2$ the database graph that corresponds to q consists of 4 203 nodes and 24 620 edges while the one that corresponds to q' has only 144 nodes and 393 edges.

6.2 Experimental results

The ranked output for q when RELWALK ranking algorithm is used is shown in Tables 1 and 2 for the query languages $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively. Rankings obtained by using $\mathcal{L}_1$ are related to frequency and thus are used as a baseline case. On the other hand, the effect on the co-author query in $\mathcal{L}_2$ appears in the obtained rankings. For example, "Nicolas Adiba" is included in the most highly ranked authors, though he appears to have a single paper in the database. However, he participates in a paper with 18 other co-authors among which there are "Michael J. Carey" (ranked first) with 47 entries and "Daniela Florescu" with 16 entries. The same holds for "Steve Kirsch" and "Michael Blow" who are all authors of the same 18-author paper.

The results obtained for q using the rankings of the RELHITS algorithm are shown in Tables 3 and 4 for query languages $\mathcal{L}'_1$ and $\mathcal{L}'_2$ respectively. There, we can make the similar observations as those made for the rankings obtained using the RELWALK algorithm. For example, "Eugene J. Shekita" appears to have much less entries in the database than authors ranked after him. This is due to the fact that "Michael J. Carey" (first in the ranking) appears to be among his co-authors.

A comparison plot for our rankings is shown in Figure 4. Each pair of rankings r, r' is compared by taking the first k authors of each ranking (x-axis) and forming the sets $r(k), r'(k)$. The y axis is the value of $\frac{|r(k)\cap r'(k)|}{k}$ and is used as a similarity between the two rankings. For the results of the query $q(y|) \equiv \pi_2 R(x, y)$ we use the first 200 ranked authors. In the plot the subscripts in the names of the algorithm correspond to the language used for the ranking.

We performed similar experiments for ranking the co-authors of "H. Garcia-Molina". Tables 5 and 6 for RELWALK and Tables 7 and 8 for RELHITS show highest ranked authors that are co-authors of H. "Garcia-Molina" (query $q' \equiv \pi_2\sigma_{6=\text{'H. Garcia-Molina'}}\sigma_{1=5}R \times R$). As before the languages $\mathcal{L}_1$ ($\mathcal{L}'_1$) and $\mathcal{L}_2$ ($\mathcal{L}'_2$) were used for constructing the corresponding graphs. Phenomena analogous to the previous experiment appear here as well. For example in Table 6, "Edward Chang" has only two papers in the database but still he is ranked 5th. The same for "Svetlozar Nestorov" who is highly ranked though he has only 3 entries in the database. This is due to his co-author list which contains very highly ranked authors like S. Abiteboul, J. Widom, R. Motwani and J. Ullman.

In Figure 5 we show an overall comparison plot of the rankings for elements which are answers to the query q' The assumptions and notation are the same as those followed in the previous experiment. For the comparisons presented in Figure 5 we only used the 70 highest ranked authors (out of the 104).

RelWalk Ranking	
Language: $\mathcal{L}_1 = \{q_1(x_j\|x_i), q_2(x\|)\}$	
with: $q_1(x_j\|x_i) \equiv \pi_{i,j}R(x_1,x_2,x_3,x_4)$ with $i \neq j$	$f_1 = 0.9$
$q_2(x\|) \equiv \pi_1 R \cup \pi_2 R \cup \pi_3 R \cup \pi_4 R$	$f_2 = 0.1$
H. Garcia-Molina	1.608E-4
Michael J. Carey	1.608E-4
H. V. Jagadish	1.569E-4
David J. DeWitt	1.552E-4
S. Abiteboul	1.530E-4
Rakesh Agrawal	1.513E-4
C. Faloutsos	1.508E-4
Surajit Chaudhuri	1.502E-4
Michael Stonebraker	1.502E-4
Raghu Ramakrishnan	1.497E-4
Jeffrey F. Naughton	1.497E-4
Jennifer Widom	1.491E-4
Yannis E. Ioannidis	1.486E-4
A. Levy	1.480E-4

Table 1: RelWalk ranking for all the authors in the database using $\mathcal{L}_1$.

6.3 Discussion

The experiments conducted and described in the previous subsection show that the obtained rankings are highly dependent on the query languages that are used for constructing the database graphs. For example when the co-author query was included in the language $\mathcal{L}_2$ the ranking was highly influenced by co-author relationships, meaning that authors with less papers but with more (or highly-ranked) co-authors appear high in the obtained rankings. Additionally, the second experiment gives an idea of how different RelWalk and RelHITS are, since the latter is query dependent and it only considers the part of the database that is related to the imposed query. This difference although not apparent in the first experiment where the imposed query was considering all the authors in the database, it becomes obvious in the second case where only the co-authors of "H. Garcia-Molina" are to be considered. The corresponding comparison Figures 4 and 5 also imply this difference in the behavior of the two ranking algorithms for different queries.

7 Conclusions

We have shown how to associate with each database, query language, and preference function a unique database graph and explored some of its interesting properties. The database graph provides a nice framework for extending existing and creating new link analysis algorithms for the Web and relational databases. The flexibility of the framework is provided by the use of relational algebra queries. The database graph also enables us to define the random querier which performs a random walk on databases by asking queries in each step of the walk. We applied our concepts to obtain two algorithms that provide rank values on partial tuples. These values are interesting on their own, but can also serve as input for top-k selection and join algorithms to obtain ranking of query results.

RelWalk Ranking	
Language: $\mathcal{L}_2 = \{q_1(x_j\|x_i), q_2(x_2\|x_6), q_3(x\|)\}$	
with: $q_1(x_j\|x_i) \equiv \pi_{i,j}R(x_1,x_2,x_3,x_4)$ with $i \neq j$	$f_1 = 0.45$
$q_2(x_2\|x_6) \equiv \pi_{2,6}\sigma_{1=5,2\neq 6}R \times R$	$f_2 = 0.45$
$q_3(y\|) \equiv \pi_1 R \cup \pi_2 R \cup \pi_3 R \cup \pi_4 R$	$f_3 = 0.1$
Michael J. Carey	2.695E-4
S. Abiteboul	2.142E-4
A.Cichocki	1.990E-4
V. Kashyap	1.990E-4
R. Brice	1.990E-4
J. Fowler	1.990E-4
W. Bohrer	1.990E-4
R. J. Bayardo	1.990E-4
David J. DeWitt	1.983E-4
H. Garcia-Molina	1.919E-4
Daniela Florescu	1.915E-4
Steve Kirsch	1.844E-4
Michael Blow	1.844E-4
Nicolas Adiba	1.844E-4

Table 2: RelWalk ranking for all the authors in the database using $\mathcal{L}_2$.

RelHITS Ranking	
Language: $\mathcal{L}'_1 = \{q_1(x_2\|x_3), q_2(x_2\|x_4)\}$	
with: $q_1(x_2\|x_3) \equiv \pi_{2,3}R(x_1,x_2,x_3,x_4)$	$f_1 = 0.5$
$q_2(x_2\|x_4) \equiv \pi_{2,4}R(x_1,x_2,x_3,x_4)$	$f_2 = 0.5$
H. Garcia-Molina	0.007572
Michael J. Carey	0.006437
H. V. Jagadish	0.006081
Surajit Chaudhuri	0.004740
David J. DeWitt	0.004603
Rakesh Agrawal	0.004291
A. Levy	0.004283
Jennifer Widom	0.004207
S. Abiteboul	0.004179
C. Faloutsos	0.004056
Raghu Ramakrishnan	0.003795
Jeffrey F. Naughton	0.003765

Table 3: RelHITS rankings for all the authors in the database using $\mathcal{L}'_1$.

We point out some interesting questions and open problems:

- How can the database graph be used to define measures of similarity between categorical data? Possible measures include the shortest path between tuples and the commute distance between nodes on the database graph.

- Is there a more close connection between the expressive power of $\mathcal{L}$ and the database graph and random querier? What are the properties of the random querier with memory ([14]).

- Finally, is there an objective way of selecting the query language used for defining the database graph.

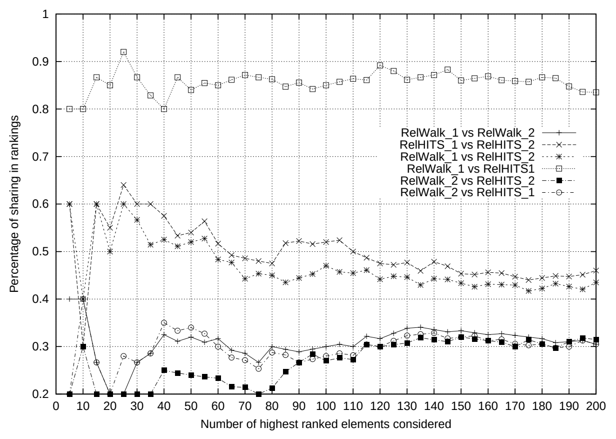

Figure 4: Comparison of rankings for all authors in the database.

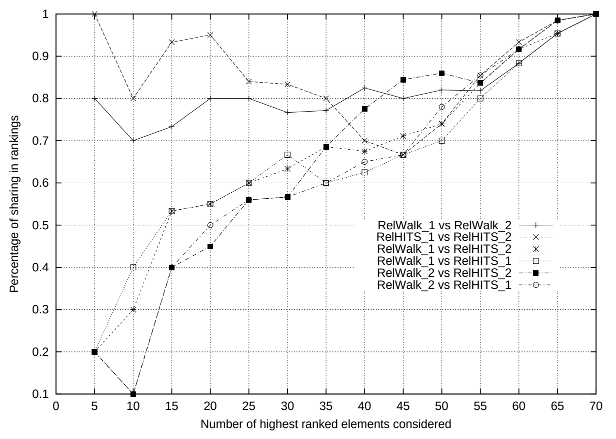

Figure 5: Comparison of rankings for all co-authors of 'H. Garcia-Molina'.

RelHITS Ranking	
Language: $\mathcal{L}'_2 = \{q_1(x_2 \vert x_3, x_6)\}$	
with: $q_1(x_2 \vert x_3, x_6) \equiv \pi_{2,3,6}\sigma_{2\neq 6,1=5} R \times R$	$f_1 = 1.0$
Michael J. Carey	0.55364
David J. DeWitt	0.02967
Jeffrey F. Naughton	0.02593
H. Garcia-Molina	0.02593
Yannis E. Ioannidis	0.02266
Miron Livny	0.01887
Raghu Ramakrishnan	0.01659
H. Pirahesh	0.01442
Michael J. Franklin	0.01086
Eugene J. Shekita	0.01047
Jennifer Widom	0.01042
Praveen Seshadri	0.00925

Table 4: RelHITS rankings for all the authors in the database using $\mathcal{L}'_2$.

Acknowledgment

We would like to thank Aris Gionis for helpful discussions.

RelWalk Ranking	
Language: $\mathcal{L}_1 = \{q_1(x_j \vert x_i), q_2(x\vert)\}$	
with: $q_1(x_j \vert x_i) \equiv \pi_{i,j} R(x_1, x_2, x_3, x_4)$ with $i \neq j$	$f_1 = 0.9$
$q_2(x\vert) \equiv \pi_1 R \cup \pi_2 R \cup \pi_3 R \cup \pi_4 R$	$f_2 = 0.1$
S. Abiteboul	1.530E-4
Raghu Ramakrishnan	1.497E-4
Jennifer Widom	1.491E-4
A. Silberschatz	1.480E-4
J. Ullman	1.425E-4
Rajeev Motwani	1.408E-4
Anand Rajaraman	1.391E-4
Luis Gravano	1.391E-4
Anthony Tomasic	1.375E-4
Ramana Yerneni	1.375E-4
Vasilis Vassalos	1.375E-4
Yannis Papakonstantinou	1.375E-4
Narayanan Shivakumar	1.375E-4
Sergey Brin	1.375E-4
Janet L. Wiener	1.375E-4
Kenneth Salem	1.375E-4

Table 5: RelWalk rankings for all co-authors of 'H. Garcia-Molina' using $\mathcal{L}_1$.

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[2] S. Agrawal, S. Chaudhuri, and G. Das. DBXplorer: A system for keyword-based search over relational databases. In *ICDE*, 2002.

[3] S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis. Automated ranking of database query results. In *CIDR*, 2003.

[4] A. Balmin, V. Hristidis, and Y. Papakonstantinou. ObjectRank: Authority-based keyword search in databases. In *VLDB*, 2004.

[5] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In *ICDE*, 2002.

[6] K. Bharat and M. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *SIGIR*, 1998.

[7] B. Bollobás. *Modern Graph Theory.* Springer-Verlag, 1998.

[8] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem.* Springer-Verlag, 1997.

[9] A. Borodin, J. S. Rosenthal, G. O. Roberts, and P. Tsaparas. Finding authorities and hubs from link structures on the World Wide Web. In *WWW*, 2001.

[10] S. Brin, R. Motwani, L. Page, R. Motwani, and T. Winograd. What can you do with the web in your pocket? *Data Engineering Bulletin*, 1998.

[11] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30:107–117, 1998.

RelWalk Ranking	
Language: $\mathcal{L}_2 = \{q_1(x_j\|x_i), q_2(x_2\|x_6), q_3(x\|)\}$	
with: $q_1(x_j\|x_i) \equiv \pi_{i,j} R(x_1, x_2, x_3, x_4)$ with $i \neq j$	$f_1 = 0.45$
$q_2(x_2\|x_6) \equiv \pi_{2,6}\sigma_{1=5,2\neq 6} R \times R$	$f_2 = 0.45$
$q_3(x\|) \equiv \pi_1 R \cup \pi_2 R \cup \pi_3 R \cup \pi_4 R$	$f_3 = 0.1$
S. Abiteboul	2.142E-4
Jennifer Widom	1.771E-4
A. Silberschatz	1.736E-4
Raghu Ramakrishnan	1.703E-4
Edward Chang	1.619E-4
J. Ullman	1.531E-4
Rajeev Motwani	1.498E-4
Roy Goldman	1.498E-4
Anand Rajaraman	1.490E-4
Svetlozar Nestorov	1.479E-4
Ramana Yerneni	1.443E-4
Yannis Papakonstantinou	1.439E-4
Luis Gravano	1.432E-4
Vasilis Vassalos	1.428E-4
Joachim Hammer	1.423E-4
Ming-Chien Shan	1.419E-4

Table 6: RelWalk rankings for all co-authors of 'H. Garcia-Molina' using $\mathcal{L}_2$.

RelHITS Ranking	
Language: $\mathcal{L}'_1 = \{q_1(x_2\|x_3), q_2(x_2\|x_4)\}$	
with: $q_1(x_2\|x_3) \equiv \pi_{2,3} R(x_1, x_2, x_3, x_4)$	$f_1 = 0.5$
$q_2(x_2\|x_4) \equiv \pi_{2,4} R(x_1, x_2, x_3, x_4)$	$f_2 = 0.5$
Jennifer Widom	0.0552
Ramana Yerneni	0.0487
Narayanan Shivakumar	0.0417
Joachim Hammer	0.0390
Luis Gravano	0.0365
Anthony Tomasic	0.0326
Chen-Chuan K. Chang	0.0298
Yannis Papakonstantinou	0.0294
Junghoo Cho	0.0291
Yue Zhuge	0.0262
Vasilis Vassalos	0.0258
Janet L. Wiener	0.0254
Sudarshan S. Chawathe	0.0245
Jeffrey Ullman	0.0238

Table 7: RelHITS rankings for all co-authors of 'H. Garcia-Molina' using $\mathcal{L}'_1$.

RelHITS Ranking	
Language: $\mathcal{L}'_2 = \{q_1(x_2\|x_3, x_6)\}$	
with: $q_1(x_2\|x_3, x_6) \equiv \pi_{2,3,6}\sigma_{2\neq 6,1=5} R \times R$	$f_1 = 1.0$
Jennifer Widom	0.0514
Narayanan Shivakumar	0.0456
Ramana Yerneni	0.0424
Luis Gravano	0.0352
Joachim Hammer	0.0350
Yannis Papakonstantinou	0.0330
Wilburt J. Labio	0.0285
Chen-Chuan K. Chang	0.0258
Junghoo Cho	0.0254
Vasilis Vassalos	0.0249
Anthony Tomasic	0.0233
Chen Li	0.0223
Jeffrey Ullman	0.0223
Yue Zhuge	0.0203

Table 8: RelHITS rankings for all co-authors of 'H. Garcia-Molina' using $\mathcal{L}'_2$.

[12] C. Ding, X. He, P. Husbands andH. Zha, and H.D. Simon. PageRank, HITS and a unified framework for link analysis. In *SIGIR*, 2002.

[13] H. D. Ebbinghaus and J. Flum. *Finite Model Theory.* Springer-Verlag, 1995.

[14] R. Fagin, A. Karlin, J. Kleinberg, P. Raghavan, S. Rajagopalan, R. Rubinfeld, M. Sudan, and A. Tomkins. Random walks with "back buttons". *Annals of Applied Probability*, 11(3):810–862, 2001.

[15] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS*, 2001.

[16] H. Garcia-Molina, J. Ullman, and J. Widom. *Database Systems, The Complete Book*. Prentice Hall, 2002.

[17] D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based on dynamical systems. In *VLDB*, 1998.

[18] V. Hristidis and Y. Papakonstantinou. DISCOVER: Keyword search in relational databases. In *VLDB*, 2002.

[19] V. Hristidis and Y. Papakonstantinou. Algorithms and applications for answering ranked queries using ranked views. *VLDB Journal*, 2003.

[20] I. F. Ilyas, W. G. Aref, and A. K. Elmagarmid. Supporting top-k join queries in relational databases. In *VLDB*, 2003.

[21] G. Jeh and J. Widom. SimRank: a measure of structural-context similarity. In *KDD*, 2002.

[22] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of ACM*, 46, 1999.

[23] R. Motwani and P. Rhaghavan. *Randomized Algorithms.* MIT Press, 1995.

[24] A. Natsev, Y.-C. Chang, J.R. Smith, C.-S. Li, and J.S. Vitter. Supporting incremental join queries on ranked inputs. In *VLDB*, 2001.

[25] A. Ng, A. Zheng, and M. Jordan. Stable algorithms for link analysis. In *SIGIR*, 2001.

[26] C. R. Palmer and C. Faloutsos. Electricity based external similarity of categorical attributes. In *PAKDD*, 2003.

[27] A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, 1998.