An Upper Bound on the Number of Accesses for Datalog α Last Queries

John Samuel, Benjamin Momège

An Upper Bound on the Number of Accesses for $Datalog^{\alpha_{Last}}$ Queries

John Samuel [*]
LIMOS, CNRS Université Blaise Pascal
Aubière, France
samuel@isima.fr

Benjamin Momège [†]
LIMOS, CNRS Université Blaise Pascal
Aubière, France
momege@isima.fr

ABSTRACT

In the mediation approach for data integration, domain rules [2, 3] were previously proposed to deal with access limitations (aka access patterns). For data integration systems (e.g., DaWeS [6, 5]) that use domain rules, we study an upper bound on the possible number of accesses implied by the evaluation of an executable query (expressed with relations having access patterns). Indeed it allows to compare various evaluation algorithms, to schedule API operation calls and meet the service level agreements (SLA) of the service providers.

1. BOUNDING THE NUMBER OF ACCESSES FOR $DATALOG^{\alpha_{LAST}}$ QUERIES

Web service providers introduce access patterns in a manner that a complete query response isn't obtained in a single operation call, but rather it mandates multiple operation calls (accesses). For our previous work DaWeS (Data Warehouse fed with Web Services) ([6, 5]), it is important to optimize (i.e. reduce) the number of accesses prior to actually making them. In our work, we introduce $Datalog^{\alpha_{Last}}$ (datalog query with access pattern on the last atom in bodies of rules) similar to the one found in [2, 3], study its operational semantics and compute the upper bound on the number of accesses. Classical inverse query rewriting algorithm doesn't take into consideration data dependencies existing among the web service API operations (relations with access patterns). In DaWeS, we explore such dependencies and study the upper bound defined in terms of source relations to compare various optimization heuristics.

Consider for example, a $Datalog^{\alpha_{Last}}$ query Φ (q is the query predicate):

$\phi_1 : dom_{X_1}(X_1) \leftarrow r_2^o(X_1)$
$\phi_2 : dom_{X_2}(X_2) \leftarrow r_1^o(X_2)$
$\phi_3 : dom_{X_3}(X_3) \leftarrow dom_{X_1}(X_1), dom_{X_2}(X_2), r_3^{iioo}(X_1, X_2, X_3, X_4)$
$\phi_4 : dom_{X_4}(X_4) \leftarrow dom_{X_1}(X_1), dom_{X_2}(X_2), r_3^{iioo}(X_1, X_2, X_3, X_4)$
$\phi_5 : dom_{X_5}(X_5) \leftarrow dom_{X_4}(X_4), r_4^{ioo}(X_4, X_5, X_1)$
$\phi_6 : dom_{X_1}(X_1) \leftarrow dom_{X_4}(X_4), r_4^{ioo}(X_4, X_5, X_1)$
$\phi_7 : rr_1(X_2) \leftarrow r_1^o(X_2)$
$\phi_8 : rr_2(X_1) \leftarrow r_2^o(X_1)$
$\phi_9 : rr_3(X_1, X_2, X_3, X_4) \leftarrow dom_{X_1}(X_1), dom_{X_2}(X_2), r_3^{iioo}(X_1, X_2, X_3, X_4)$
$\phi_{10} : rr_4(X_4, X_5, X_1) \leftarrow dom_{X_4}(X_4), r_4^{ioo}(X_4, X_5, X_1)$
$\phi_{11} : q(X_1, X_3, X_5) \leftarrow rr_1(X_2), rr_2(X_1), rr_3(X_1, X_2, X_3, X_4), rr_4(X_4, X_5, X_1)$

A naive datalog query evaluation of the above program Φ will iterate through every $\phi_j, 1 \leq j \leq 11$ until a fixpoint is obtained. $\mathcal{D}_0$ is the initial database and $\mathcal{D}_i, 1 \leq i \leq n_0$ is the new database obtained after every iteration. For web services API, $|r_3^{iioo}(X_1, X_2, X_3, X_4)|$ or $|r_3^{iioo}(\mathcal{D}_0)|$ can be obtained (e.g., total number of search results) but $|r_3^{iioo}[X_4]|$, (i.e., the number of X_4 attributes in r_3) cannot be obtained. We compute the total number of accesses for all iterations and all queries ϕ_j as:

$$|Accesses(\Phi, \mathcal{D}_0)| = \sum_{i=0}^{n_0} \sum_{j=1}^{11} |Accesses(\phi_j, \mathcal{D}_i)|$$
$$\leq 4n_0 + 3n_0^3 \times (|r_2^o(\mathcal{D}_0)| + |r_4^{ioo}(\mathcal{D}_0)|) \times |r_1^o(\mathcal{D}_0)| + 3 \times n_0^2 \times |r_3^{iioo}(\mathcal{D}_0)|$$

where n_0 is the number of the last iteration (during which the fixpoint is generated). If we consider n_0 as a parameter, we can conclude that:

$|Accesses(\Phi, \mathcal{D}_0)| =$
$\mathcal{O}\left((|r_2^o(\mathcal{D}_0)| + |r_4^{ioo}(\mathcal{D}_0)|) \times |r_1^o(\mathcal{D}_0)| + |r_3^{iioo}(\mathcal{D}_0)|\right)$ ■

Discussion: Web service API operation calls are expensive and must be reduced. An upper bound as shown above is useful to compare various optimization [3, 1] techniques on a $Datalog^{\alpha_{Last}}$ program. For example, with cache rules [3] optimization, same accesses are not repeated again, as seen in the following excerpt (after transforming the above program):

...
$\phi_9 : rr_3(X_1, X_2, X_3, X_4) \leftarrow dom_{X_1}(X_1), dom_{X_2}(X_2), r_3^{iioo}(X_1, X_2, X_3, X_4)$
$\phi_3 : dom_{X_3}(X_3) \leftarrow rr_3(X_1, X_2, X_3, X_4)$

[*]Current Affiliation: Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR-CNRS 5205, Labex IMU, Laboratoire d'Informatique en Image et Systèmes d'Information
Email: john.samuel@liris.cnrs.fr

[†]Current Affiliation: Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France
INRIA, France
Email: benjamin.momege@inria.fr

BDA 14 octobre 2014, Grenoble-Autrans, France.

$\phi_4 : dom_{X_4}(X_4) \leftarrow rr_3(X_1, X_2, X_3, X_4)$
...
The total number of accesses in this case is given by:
$|Accesses(\Phi, \mathcal{D}_0)| \leq 2n_0 + n_0^3 \times (|r_2^o(\mathcal{D}_0)| + |r_4^{ioo}(\mathcal{D}_0)|) \times |r_1^o(\mathcal{D}_0)| + n_0^2 \times |r_3^{iioo}(\mathcal{D}_0)|$.

n_0 as a parameter is a safe assumption for two reasons. Firstly the overall goal is to recognize the products in the bound since they are usually responsible for the majority of accesses. Secondly in our tests, for different relation sizes and randomly generated data, n_0 generally stayed constant (3 for above e.g.).

2. ACKNOWLEDGMENTS

We acknowledge Christophe Rey for his direction. We would like to thank Conseil General of the Region of Auvergne (France), CNRS and FEDER for funding our respective PhD Theses (*Feeding a Data Warehouse with Data coming from Web Services [4]* and *Paths in graphs with forbidden transitions*).

3. REFERENCES

[1] Andrea Calì and Davide Martinenghi. Querying data under access limitations. In *ICDE*, pages 50–59, 2008.
[2] Oliver M. Duschka, Michael R. Genesereth, and Alon Y. Levy. Recursive query plans for data integration. *J. Log. Program.*, 43(1):49–73, 2000.
[3] Chen Li and Edward Y. Chang. Query planning with limited source capabilities. In *ICDE*, pages 401–412, 2000.
[4] John Samuel. *Feeding a data warehouse with data coming from web services. A mediation approach for the DaWeS prototype.* PhD thesis, Université Blaise Pascal, 2014. Thèse de doctorat dirigée par Toumani, Farouk et Rey, Christophe Informatique Clermont-Ferrand 2 2014.
[5] John Samuel. Towards a data warehouse fed with web services. In *ESWC PhD Symposium*, 2014.
[6] John Samuel and Christophe Rey. Dawes: Data warehouse fed with web services. In *INFORSID*, 2014.

APPENDIX

Une Borne Supérieure sur le Nombre d'accès pour les Requêtes $Datalog^{\alpha_{Last}}$

A. RÉSUMÉ

Dans l'approche médiation pour l'intégration de données les règles de domaine ont été proposées [2, 3] pour prendre en compte ces sources. Pour les systèmes d'intégration qui utilisent les règles de domaine (par exemple DaWeS [6, 5]), il est utile de pouvoir calculer une borne supérieure du nombre d'accès impliqués par l'évaluation d'une requête exécutable (exprimée avec des relations limitées en accès). Ainsi cette borne supérieure permet de comparer les différents algorithmes d'évaluation de requêtes exécutables. Ceci permet une meilleure planification des appels de service et donc une plus grande facilité pour gérer les contraintes de qualité de service (service level agreements, SLA) imposées par leurs fournisseurs.