Linked Data Indexing Methods: A Survey[⋆]

Martin Svoboda and Irena Mlýnková

Department of Software Engineering, Charles University in Prague
Malostranske namesti 25, 118 00 Prague 1, Czech Republic
Contact e-mail: {svoboda,mlynkova}@ksi.mff.cuni.cz

Abstract. Documents on the contemporary Web are based especially on HTML formats and, therefore, it is rather difficult to retrieve hidden structured information from them using automated agents. The concept of Linked Data based primarily on RDF data triples seems to successfully solve this drawback. However, we cannot directly adopt the existing solutions from relational databases or XML technologies, because RDF triples are modelled as graph data and not relational or tree data. Despite the research effort in recent years, several questions in the area of Linked Data indexing and querying remain open, not only since the amount of Linked Data globally available significantly increases each year. This paper attempts to introduce advantages and disadvantages of the state-of-the-art solutions and discuss several issues related to our ongoing research effort – the proposal of an efficient querying framework over Linked Data. In particular, our goal is to focus on large amounts of distributed and highly dynamic data.

Keywords: Linked Data, RDF, indexing, querying, SPARQL.

1 Introduction

The majority of documents on the contemporary Web are based primarily on HTML formats. Although these documents often contain hidden structured and interlinked information, it is quite difficult for automated agents to retrieve such information. Therefore, an idea of Linked Data appeared in order to extend the Web of Documents towards the Web of Data [4].

Linked Data do not represent any particular standard; we only talk about a set of recommended principles and techniques, which lead to the publication of data in a way more suitable for their automated processing. First, each real-world entity should be described by a unique URL identifier. These identifiers can be dereferenced by HTTP to obtain information about the given entities. And, finally, these entity representations should be interlinked together to form a global open data cloud – the Web of Data.

Even though there are several particular ways, the most promising is probably the RDF (Resource Description Framework) [15] standard, where data are

[⋆] This work was supported by the Charles University Grant Agency grant 4105/2011, the Czech Science Foundation grant P202/10/0573 and the grant SVV-2011-263312.

modelled as triples conforming to the concept of *subject-predicate-object*. An alternative way to view these triples are graphs, where vertices correspond to *subjects* and *objects*, edges represent the triples themselves and are labelled by *predicates*. At the implementation level we can publish RDF triples in a form of RDF/XML [3] syntax and along the data we can also publish RDFS [6] schemata or OWL [16] ontologies restraining the allowed content of such RDF data.

In recent years, a significant effort was made not only in a theoretical research, but also in the amount of Linked Data globally available. However, since RDF triples are modelled as graphs, we cannot directly adopt existing solutions from relational databases and XML [5] technologies. Thus, the area of Linked Data includes many open problems, starting from application architecture questions and ending, e.g., with user interaction paradigm.

Objectives. The purpose of this paper is to provide a survey of the area of Linked Data indexing techniques. It is clear that indexing is tightly connected with storing and querying, however, we will only focus on state-of-the-art solutions in the area of indexing. We will mutually compare the existing approaches, discuss general issues and important aspects of a proposal of index structures supporting effective and efficient querying, but we will not provide any performance comparisons. Thus, this paper should give the reader the basic insight into the area of Linked Data indexing, its various ideas and principles, but due to the lack of space, we are not able to go to detail.

Outline. In section 2 we discuss issues and dimensions that need to be considered in the comparison of Linked Data indexing methods. Section 3 gives the thorough description of the existing approaches in this area, while in Section 4 we provide an overall summary of these approaches and discuss found observations and open questions. Finally, Section 5 concludes this paper.

2 Analysis

Whereas the logical model of RDF data is based on graphs, standard databases work with relational tables and semi-structured XML documents can be viewed as trees. Despite these differences, we can still find inspiration in these two well established areas. In this section, we outline a number of issues that need to be considered when proposing new indexing methods. We need to consider at least architecture, storage and querying aspects. Next, we introduce a set of dimensions using which we can compare and classify the existing approaches.

2.1 General Issues

Architecture. The fundamental question of each querying system is the mutual relationship between physical storages, index structures and querying capabilities. It is clear that querying without data has no sense, and vice versa. In other words, all presented issues can only hardly be considered separately. First, we need to discuss what querying constructs we need in a particular application, i.e. what expressive power we require or even which querying language we want to choose. Then, the storage itself and index structures should be proposed in the way capable to effectively and efficiently provide results of queries.

Since the concept of Linked Data emerged to support the idea of the Web of Data, we cannot ignore its main feature based on data distribution. Having no central points in the infrastructure would be worth, but it seems that without at least some local knowledge, the querying over the Web would not be sufficiently fast. If we want to evaluate queries only via online accessing remote data sources, we are assured that we obtain accurate and up-to-date results, but we cannot neglect the amount of data needed to be transferred and the required time.

On the other hand, local approaches may enable more efficient query processing, but we need to deal with large amounts of data and their ageing. It seems that a sort of federated and integrated querying framework [21] could be a good way to overcome the mentioned issues. Nevertheless, we need to consider several reasons that may prevent maintaining local data copies. Besides technical ones, there can also be legal reasons such as copyrights.

Storage. In standard relational databases, data are stored in tables and index structures have only an auxiliary role to support efficient query evaluation. The important thing is that no data are stored in indices themselves, since they only contain duplicated data fragments or derived statistics. In the area of RDF storing, we can even find indexing approaches [2] which are completely isolated from the physical database layer, i.e. a query can be evaluated only using the index structure itself.

Although native approaches for querying RDF data could generally represent more efficient solutions, the mentioned relational databases [1] benefit from decades of experience and research results. We can come across three basic ways how to store RDF data in relational tables. The first possible solution is based on one big table with three columns for subjects, predicates and objects. The second approach is based on a set of tables, where values in the first column always identify subjects and values in remaining columns correspond to objects of predicates determined by these columns (we group several predicate and object pairs for a given subject). Finally, the third approach has a separate table for each predicate, always with 2 columns for subject and object pairs.

Querying Language. According to [7] we can divide RDF querying languages into three basic layers: syntactic, structural and semantic. Probably the most widespread language SPARQL [20] represents the structural querying. Having a query formulated as a graph pattern with fixed values and/or variables, the query processor attempts to find all matches of this graph pattern against the entire data graph stored in the database.

The important position has also the concept of fulltext querying based on keywords. Although we may require such functionality even in local querying systems, these techniques play the main role in Web search engines. The important feature of this model is that we are usually not interested in which particular component (subject, predicate, object) the specified keyword should exactly be located, whereas in graph querying models the value equality testing actually needs to consciously distinguish between particular triple particles.

After a submitted query is parsed and transformed into an internal representation, we need to find the most suitable query evaluation plan. In this sense a special position in query optimization techniques has the join ordering. It is quite interesting that we can use similar ideas to the nested loop algorithm from relational databases. However, there are other aspects that need to be considered. The problem is not only that we usually have to rely on heuristics, since we are not able to consider all possible plans, but we also use imprecise statistics.

2.2 Classification of Approaches

Existing approaches can be compared in many different ways. First of all, we can talk about the architecture, which is primarily derived from the scope of processed data. We distinguish between *local*, *distributed* and *global* approaches.

If the index structures themselves allow efficient insertions, updates and deletions, we talk about *dynamic* structures, otherwise about *static* ones. These structures may serve for indexing *data*, or only *statistics* about them. Anyway, each system focuses on different data units: *triples* conform to standard RDF triples, *quads* are extended with the context, whereas in case of *sources* we work with semantic documents or other files and services.

Approaches may also be compared by querying languages, their expressive power or allowed constructs. Although there are *syntactic* and *semantic* layers, we only focus on the *structural* one. In this case, queries can be based on *fulltext* or *graph* patterns. Anyway, the querying model is closely connected with the model of index structures. Thus, we can index *keywords*, *triples*, *quads*, *trees*, *paths* or other *areas*. Indexing approaches also differ in access patterns: *universal* treat subject, predicate and object components equally, whereas *dedicated* not.

3 Approaches

Not all possible combinations derived from introduced dimensions make sense. Nevertheless, we are able to derive three main categories of existing approaches: *local* querying systems, *distributed* source selection techniques and *global* searching engines. The description of existing approaches is the subject of this section. For simplification, we will use abbreviations S, P, O and C for *subject*, *predicate*, *object* and *context* quad particles respectively.

3.1 Local Approaches

Quad Index. Index structures proposed by Harth and Decker [11] enable querying of local data quads with context. These structures involve Lexicon (an inverted list for keywords and two-way translation maps for term identifiers based on B$^+$-trees) and Quad indices (B$^+$-trees for $SPOC$, POC, OCS, CSP, CP, OS orderings) allowing to query in all 16 access patterns. Despite data quads themselves, these indices also contain statistics about data, e.g. quad $(s, p, 0, 0)$ in $SPOC$ index represents the number of all quads with given s and p values.

Vertical Partitioning. Abadi et al. [1] proposed a model of storing RDF triples in relational databases in a way of a separate table for each predicate (with one column for subjects, second for objects). This implies that we are able to store pre-computed paths. Similarly to the previous approach, the authors use the translation of strings into identifiers.

RDF-3X Engine. The core of the stream processor RDF-X proposed by Neumann and Weikum [17] is based on six B^+-tree indices for all SPO, SOP, OSP, OPS, PSO and POS access patterns. Additionally, they also use indices with statistics (S, P, O, SP, PS, PO, OP, SO and OS projections) and selectivity histograms and statistics for pre-computed path or star patterns.

Sextuple Index. The idea of HexaStore approach by Weiss et al. [27] is based on similar SPO, SOP, OSP, OPS, PSO and POS index structures, however, these are implemented as ordered nested lists. All these lists contain only identifiers instead of strings, again.

Matrix Index. BitMat is an approach proposed by Atre et al. [2]. The index model is based on a matrix with three dimensions for S, P and O values (terms are translated to identifiers, which are used as matrix indices). Each cell contains a bit value equal to 1 if and only if the given triple is stored in the database, otherwise value 0. The index is organized as an ordinary file with all SO, OS, PO and PS slices stored using a bit run compression over individual slice rows.

Path Index. An index structure for path queries proposed by Liu and Hu [14] is inspired by traditional information retrieval methods for processing texts. Its main idea is based on suffix arrays, using which we are able to efficiently index paths as sequences of fixed S, P and O values.

GRIN Index. Udrea et al. [26] introduced a model based on splitting data graphs into subgraph areas that are described by conditions limiting their content. The idea is derived from a metric defined on URIs and literals (e.g. a minimal number of edges in a data graph between a given pair of values). The index structure itself is a balanced binary tree, where internal nodes represent mentioned areas and leaf nodes store data triples conforming to these areas.

Structure Index. The last presented local approach is a parameterised index introduced by Tran and Ladwig [25]. Their model is based on bisimilarity relations, putting in a relation such two vertices of the data graph that share the same outgoing and ingoing edges (reflecting only predicates). Vertices from the same equivalence class have the same characteristics and, therefore, prompted queries can first be evaluated over these classes to prune required data.

3.2 Distributed Approaches

The purpose of the following approaches is to work with distributed sources and provide transparent querying over their data.

Repository Index. An index by Stuckenschmidt et al. [22] is inspired by object databases. It captures statistics about paths and enables querying over them using tree pattern queries with fixed values of predicates. The index structure itself is hierarchical (for each path it also contains all its subpaths).

Federated Querying. Quilitz and Leser [21] proposed a system for transparent integrated querying over distributed and autonomous sources. The core of this approach is a language for description of distributed sources, in particular, data triples they contain, together with other source characteristics.

Data Summaries. The purpose of a data summary index by Harth et al. [12] is to enable the source selection over distributed data sources. Data triples are modelled as points in a 3-dimensional space (S, P, and O coordinates are derived by hash functions). The index structure is a QTree based on standard R-Trees. Internal nodes act as minimal bounding boxes for nested nodes, leaf nodes contain statistics about data sources, not data triples themselves.

3.3 Global Approaches

Finally, we briefly outline three global searching approaches. All of them are primarily inspired by traditional information retrieval methods.

Swoogle. The first system was proposed by Ding et al. [9]. Its objective is to serve as a searching engine over semantic documents, both data and ontologies.

SWSE. The purpose of SWSE by Harth et al. [10] is to provide a system for global searching over quads (RDF triples with their context). The querying focuses not only on keyword matching, but it also supports concept filtering.

Sindice. Oren et al. [18] introduced a global engine for searching semantic documents on the Web, allowing to query via keywords, inverse functional properties and resource URIs.

4 Summary

Summarising approach descriptions in the previous section, we can outline several observations and aspects that we consider as important to be discussed during the proposal of new techniques for indexing Linked Data. We have also identified several areas we consider as not yet sufficiently investigated.

4.1 Approaches Comparison

It seems that the existing approaches represent efficient proposals for querying RDF data. However, the majority of existing approaches have several limitations or assumptions which cause their difficult usage in the general case. Moreover, their efficiency often highly depends on allowed querying constructs, but it is not clear, what characteristics real-world data and queries really have.

In Table 1 we provide a brief and simplified overview of all existing approaches we have listed in this paper. Besides different intended purposes of these proposals, we can especially put the attention to the comparison of data, query and index models behind them.

Despite the ideas of particular solutions are sometimes very different, we can find at least the following common aspects.

Approach name	Scope type	Data items	Query model	Index model	Basic approach description
Local Querying Methods					
Harth 2005 Quad index [11]	Local	Quads	Graphs YARS QL	Quads Text	Lexicon using inverted lists Quad indices using B^+-trees
Abadi 2007 Partitioning [1]	Local	Triples	Graphs SQL	Paths	Table for each predicate Standard RDBMS indices
Neumann 2008 RDF-3X [17]	Local	Triples	Graphs SPARQL	Triples	Triple, double, single B^+-trees Histograms and path statistics
Weiss 2008 Hexastore [27]	Local	Triples	Graphs SPARQL	Triples	Nested ordered lists Dictionary encoding of URIs
Atre 2010 BitMat index [2]	Local	Triples	Graphs SPARQL	Triples	Slices of 3D matrix Bit run compression of rows
Liu 2005 Path index [14]	Local	Triples	Paths SPARQL	Paths	Paths as sequences of terms Stored using suffix arrays
Udrea 2007 GRIN index [26]	Local	Triples	Graphs SPARQL	Circles	Unlimited paths querying Binary tree over graph circles
Tran 2010 Struct. index [25]	Local	Triples	Graphs SPARQL	Graphs	Contraction to extensions Sets of in/outgoing predicates
Distributed Querying Methods					
Stucken. 2004 Repositories [22]	Dist.	Sources	Trees SeRQL	Paths	Paths as predicate sequences Hierarchical paths index
Quilitz 2008 DARQ [21]	Dist.	Sources	Graphs DARQ	Services	Service descriptions index Capabilities and selectivity
Harth 2010 Summaries [12]	Dist.	Sources	Graphs SPARQL	Boxes	Triples hashing to 3D points Q-tree index with buckets
Global Searching Methods					
Ding 2004 Swoogle [9]	Global	Files	Fulltext	Text	Semantic web documents Keywords and n-grams
Harth 2007 SWSE [10]	Global	Quads	Fulltext	Text	Global quads querying Keywords, IFP and concepts
Oren 2008 Sindice [18]	Global	Files	Fulltext	Text	Invertes lists to sources Keywords, URI and IFP

Table 1. Comparison of existing approaches

Compression. The shared idea of the majority of indexing methods is the way of storing string values of URIs and literals, since there is a high probability that strings may have multiple occurrences in the database. Therefore, it seems very effective to store these strings only once in a special storage, assign them unique integer identifiers, and use them in RDF triples instead of the original terms. As a consequence, frequently executed value equality tests during the query evaluation may then be executed much faster.

Data Pruning. Efficient systems also support the query evaluation via a set of optimizations. We can propagate data filtering selections as close as possible to their fetching, or we can perform data pruning before the phase of joining. In case of distributed approaches, we focus on the problem of source selection, i.e. to access data only of those remote sources that are relevant to the query. Generally, we want to avoid processing of irrelevant data whenever possible.

4.2 Open Problems

According to the discussed issues and the comparison of the existing approaches, we can formulate the following open problems in the area of indexing.

Architecture. Finding an appropriate compromise between processing local or distributed data forms one of the most important questions. Maintaining local copies of data may benefit from convenient conditions for more efficient query evaluation; however, we are not always able or allowed to gather the data under our control. The second approach is based on accessing distributed data on-the-fly using link traversal. This method suffers from transfer requirements, although it ensures work with up-to-date data.

Scalability. Even though existing approaches work with large sets of data, experiments performed using various sets of data, queries and prototype implementations of discussed solutions imply that we are still not able to sufficiently flatten performance of such approaches and the explosion of the Web of Data size. While we could find about 10 globally important data sources with 920 million triples and 150 thousand links in 2007[1], these numbers increased to about 200 important sources, 25 billion triples with 395 million links in 2010[2].

Dynamicity. Although experiments [8] consider the Web of Documents, it seems that data on the Web of Data tends to aging too. Moreover, we especially need not only to handle simple data modifications, but also deal with broken links and attempt to anticipate or correct them. Achieving consistently connected data [19] is necessary, when we maintain local data copies or summaries. Unfortunately, the problem is that index structures are often static and do not allow any further modifications like inserts, updates or deletes.

Quality. The increasing number of globally available data on the Web also causes issues of data quality and trust. Especially in the context of global search engines we need to propose accurate metrics or other techniques for determining relevance of particular query answers. For this purpose we can utilize data provenance or even knowledge and relationships from social networks [13].

[1] `http://esw.w3.org/SweoIG/TaskForces/CommunityProjects/LinkingOpenData/`
[2] `http://www.ckan.net/`

5 Conclusion

Recent years show that Linked Data became a suitable way of building the Web of Data that can be automatically processed without any special effort, such as without complicated information retrieval from standard HTML pages on the contemporary Web of Documents.

Since Linked Data in a form of the RDF standard represent graph data, we cannot directly adopt existing research results from areas of relational databases and XML technologies. And since the amount of Linked Data globally available still grows, several questions need to be solved to offer efficient systems.

The purpose of this paper was to provide an overview of existing Linked Data indexing approaches. We discussed issues related to the proposal of new indexing methods, introduced a set of dimensions for comparing existing solutions, and also identified aspects that can still be considered as open problems.

This survey is connected with our ongoing research effort [23] that should result in a proposal of an entire framework for efficient and effective querying of Linked Data. In particular, we want to deal especially with the following aspects: data *distribution*, *scalability*, *dynamicity* and *quality*. Although existing approaches show promising ways of solving our problem, the combination of all named assumptions remains unsolved. For this purpose we not only want to propose novel techniques together with the prototype implementation, but we also want to harness characteristics of real-world data detected using Analyzer [24], our framework for robust analyses of documents on the Web.

References

1. Abadi, D.J., Marcus, A., Madden, S.R., Hollenbach, K.: Scalable Semantic Web Data Management Using Vertical Partitioning. In: Proc. of the 33rd Int. Conf. on Very Large Data Bases. pp. 411–422. VLDB '07, VLDB Endowment (2007)
2. Atre, M., Chaoji, V., Zaki, M.J., Hendler, J.A.: Matrix "Bit" loaded: A Scalable Lightweight Join Query Processor for RDF Data. In: Proc. of the 19th Int. Conf. on World Wide Web. pp. 41–50. WWW '10, ACM, New York, NY, USA (2010)
3. Beckett, D.: RDF/XML Syntax Specification (Revised) (2004), `http://www.w3.org/TR/rdf-syntax-grammar/`
4. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story so far. International Journal on Semantic Web and Information Systems 5(3), 1–22 (2009)
5. Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., Yergeau, F., Cowan, J.: Extensible Markup Language (XML) 1.1 (Second Edition) (2006), `http://www.w3.org/TR/xml11/`
6. Brickley, D., Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema (2004), `http://www.w3.org/TR/rdf-schema/`
7. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In: The Semantic Web – ISWC 2002. LNCS, vol. 2342, pp. 54–68. Springer Berlin / Heidelberg (2002)
8. Cho, J., Garcia-Molina, H.: The Evolution of the Web and Implications for an Incremental Crawler. In: Proc. of the 26th Int. Conf. on Very Large Data Bases. pp. 200–209. VLDB '00, Morgan Kaufmann Publishers Inc., USA (2000)

9. Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V., Sachs, J.: Swoogle: A Search and Metadata Engine for the Semantic Web. In: Proceedings of the 13th ACM Int. Conference on Information and Knowledge Management. pp. 652–659. CIKM '04, ACM, New York, NY, USA (2004)
10. Harth, A., Hogan, A., Delbru, R., Umbrich, J., O'Riain, S., Decker, S.: SWSE: Answers Before Links. In: Proc. of the Semantic Web Challenge 2007 co-located with ISWC 2007 + ASWC 2007. vol. 295, pp. 136–144. CEUR-WS.org (2007)
11. Harth, A., Decker, S.: Optimized Index Structures for Querying RDF from the Web. In: Third Latin American Web Congress, 2005. LA-WEB 2005. IEEE (2005)
12. Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K.U., Umbrich, J.: Data Summaries for On-demand Queries over Linked Data. In: Proc. of the 19th Int. Conf. on World Wide Web. pp. 411–420. WWW '10, ACM, NY, USA (2010)
13. Knap, T., Mlynkova, I.: Quality Assessment Social Networks: A Novel Approach for Assessing the Quality of Information on the Web. In: QDB. pp. 1–10 (2010)
14. Liu, B., Hu, B.: Path Queries Based RDF Index. In: Proceedings of the First International Conference on Semantics, Knowledge and Grid. pp. 91–93. IEEE Computer Society, Los Alamitos, CA, USA (2005)
15. Manola, F., Miller, E.: RDF Primer (2004), `http://www.w3.org/TR/rdf-primer/`
16. McGuinness, D.L., Harmelen, F.v.: OWL Web Ontology Language: Overview (2004), `http://www.w3.org/TR/owl-features/`
17. Neumann, T., Weikum, G.: RDF-3X: A RISC-style Engine for RDF. Proc. VLDB Endow. 1, 647–659 (August 2008)
18. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H., Tummarello, G.: Sindice.com: A Document-oriented Lookup Index for Open Linked Data. International Journal of Metadata, Semantics and Ontologies 3(1), 37–52 (2008)
19. Popitsch, N.P., Haslhofer, B.: DSNotify: Handling Broken Links in the Web of Data. In: Proceedings of the 19th International Conference on World Wide Web. pp. 761–770. WWW '10, ACM, New York, NY, USA (2010)
20. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF (2008), `http://www.w3.org/TR/rdf-sparql-query/`
21. Quilitz, B., Leser, U.: Querying Distributed RDF Data Sources with SPARQL. In: The Semantic Web: Research and Applications. LNCS, vol. 5021, pp. 524–538. Springer Berlin / Heidelberg (2008)
22. Stuckenschmidt, H., Vdovjak, R., Houben, G.J., Broekstra, J.: Index Structures and Algorithms for Querying Distributed RDF Repositories. In: Proc. of the 13th Int. Conf. on World Wide Web. pp. 631–639. WWW '04, ACM, NY, USA (2004)
23. Svoboda, M., Mlynkova, I.: Efficient Querying of Distributed Linked Data. In: Proceedings of the 2011 Joint EDBT/ICDT Ph.D. Workshop. pp. 45–50. PhD '11, ACM, New York, NY, USA (2011)
24. Svoboda, M., Starka, J., Sochna, J., Schejbal, J., Mlynkova, I.: Analyzer: A Framework for File Analysis. In: Database Systems for Advanced Applications. LNCS, vol. 6193, pp. 227–238. Springer Berlin / Heidelberg (2010)
25. Tran, T., Ladwig, G.: Structure Index for RDF Data. In: Workshop on Semantic Data Management (SemData@VLDB) 2010 (2010)
26. Udrea, O., Pugliese, A., Subrahmanian, V.S.: GRIN: A Graph Based RDF Index. In: Proceedings of the 22nd National Conference on Artificial Intelligence – Volume 2. pp. 1465–1470. AAAI Press (2007)
27. Weiss, C., Karras, P., Bernstein, A.: Hexastore: Sextuple Indexing for Semantic Web Data Management. Proc. VLDB Endow. 1, 1008–1019 (August 2008)