Semantic optimization of biomolecular queries in object-oriented database systems

Karl Aberer , Klemens Hemm

GMD-IPSI, Dolivostr. 15, D-64293 Darmstadt, GERMANY

email: {aberer,hemm}@darmstadt.gmd.de

Introduction

Research efforts in molecular biology have led to an explosion both in quantity and quality of biomolecular data. In particular databases that integrate data from many different, heterogeneous information resources lead to complex database schemas and large databases. In order to support the user in creatively exploiting these information resources, flexible and efficient retrieval capabilities are needed.

We sketch in this paper our approach to optimization of declarative queries in object-oriented databases, that allows to exploit the specific semantic characteristic of datatypes occurring in scientific applications like molecular biology. These processing capabilities are used within the German national joint project RELIWE (RELIWE is a research consortium in which EMBL Heidelberg , GMD-IPSI and GMD-SCAI participate as research institutions, and BASF, Ludwigshafen and E. Merck, Darmstadt participate as industrial partners; the project is funded by the German "Bundesministerium fuer Bildung und Wissenschaft" BMBW, grant number 01IB302E.). In RELIWE's subproject Docking-D an integrated protein/ligand database covering many datasets, that are relevant for drug design [1], including PDB , Swissprot, PIR , DSSP, HSSP and PMD, has been developed on top of GMD-IPSI's object-oriented database system VODAK [16]. Dedicated indexing mechanisms and algorithms are integrated with the database system in order to provide the user with powerful and efficient access mechanisms. The data has been integrated physically for efficiency reasons, however, conceptually also the virtual integration of databases is feasible and will be used in the future for integrating further information resources, like Medline.

Retrieval approaches

Retrieval capabilities of existing database systems, as they are used in molecular biology, belong in general to one of the following categories.

Browsing and navigation

The user can inspect the database content by browsing through the content of classes, applying simple keyword-driven search and navigating along predefined links. Such functionality is typically offered through hypertext-like interfaces, like WWW. Systems like Entrez [14] or AceDB [6] combine these retrieval facilities typically with comfortable user interfaces. This approach is also used for integrating heterogeneous information resources, like with SRSWWW .

Search algorithms

Many systems offer highly efficient algorithms to search in extreme large and uniform databases or over many different databases for answering fixed or parameterized queries. The search is often supported by application-specific index structures. Examples for this approach are some of the search algorithms incorporated in WHATIF [15] or the BLAST [5] tool. The search interfaces of these tools are either provided within the interfaces of whole systems or as stand-alone scripts.

Declarative Querying

The notion of declarative query language stems from the area of database management systems, in particular relational database systems [12], where SQL is the most prominent example. Query languages are designed to access databases in a completely application-independent way. They allow a declarative specification of the information need, and are backed by powerful query optimization and evaluation components. With declarative query languages the user specifies his information need ("what"), while the system decides on the optimal operational strategy to obtain this information ("how"). Query languages are typically provided through an interpreter or are embedded in programming languages.

Each of the three approaches has its own advantages and disadvantages. Browsing and navigation allows the user to interactively explore a database, but cannot support the evaluation of complex requests. Search algorithms answer special queries with extreme efficiency, but are obviously also extremely inflexible. Declarative query languages offer a very flexible and expressive retrieval capability, but their availability is today mainly restricted to relational database systems, which are not adequate to model the complex biomolecular data in many cases. Our interest is to provide declarative querying capabilities for the complex databases that emerge from the integration of many different information resources in molecular biology.

Object-oriented database systems

Within our project environment we favored object-oriented database systems for several reasons. It is in the meanwhile a well-accepted fact that the object-oriented data model is an adequate approach to model biomolecular data [2]. It allows to represent strongly interconnected data more directly for simpler usage and maintenance. Scientific data types can be efficiently implemented through abstract data type definitions. By means of encapsulation, object-oriented database systems allow to support complex operations within the database schema. Encapsulation is important to maintain complex consistency constraints and to maintain consistency of derived data, but also to integrate complex external operations. As pointed out in [8] object-oriented data models are a suitable basis for the integration of heterogeneous databases.

Retrieval in object-oriented database systems

The primary access to object-oriented databases is through their programming language interface. Within the programming language it is possible to access properties of objects, to follow references to other objects, perform browsing and navigation, to perform simple selections, and to invoke methods on objects. These are for example the capabilities that are typically offered by C++-based object-oriented database systems [11]. Thus, leaving aside the user interface layer, object-oriented database systems

support both browsing and navigation and the usage of search algorithms, which can be encapsulated in methods.

A major drawback of many object-oriented database systems is that they do not support declarative querying capabilities [10]. Although substantial research on declarative object-oriented query languages has been performed, only few systems really offer advanced query language and query processing features. Efficiently supporting declarative querying capabilites in object-oriented database systems is an important step towards integrating all of the three database access mechanisms, namely browsing, algorithms, and declarative querying.

We illustrate the integration of the different retrieval capabilities by an example query:

```
ACCESS      chain2.name, model.name
FROM        chain1, chain2 IN db1::Chain,
            structureModel IN chain2.protein.models
WHERE       chain1.swissprotAccessCode == 'P02924'  AND
            chain1->NWalignment(chain2) >= 0.95     AND
            model.elucidationMethod == 'NMR'
```

This is a declarative query in the VODAK query language VQL against the RELIWE receptor-ligand database. The query retrieves all chains, that are homologous to the chain of a particular protein (i.e. 'P02924'), with the additional constraint, that for the homologous chain there exists a protein structure determined by NMR. Navigation is necessary (e.g. chain2.protein.models), because sequence and structure information is encapsulated in different classes (i.e. Chain, Model) related to the protein class. An externally implemented alignment algorithm (Needleman/Wunsch [13]) has been incorporated as a database method NWalignment. The query is declarative as it leaves for example open, in which order the classes are traversed, the objects within classes are visited, or the conditions are evaluated.

Most research on efficient processing of queries has been directed towards the structural aspects of object-oriented data models. This can be understood for historical reasons, as indexing and query optimization techniques for the structural aspects are close to relational and nested relational techniques. Another reason is that the semantics of methods is far more difficult to access and exploit for query processing. On the other hand considering method semantics for query optimization is an equally important and interesting issue for declarative object-oriented query languages [3]. This fact was already recognized in the area of biomolecular databases [9]. Only when methods' semantic is considered in query processing the full power of methods can be exploited by queries.

Within VODAK we have implemented an extensible query processing component [3]. We are using an algebraic and rule-based approach to query optimization. The query optimizer is generated by using the Volcano optimizer generator [7] and we can generate application-specific query optimization modules. We are currently experimenting with application-specific optimization rules that exploit the particular semantics of database methods of the integrated RELIWE protein-ligand database. Our goal is to arrive at a general methodology for the definition of application-specific optimization rules exploiting method semantics by the schema designer and their integration into query optimization.

Access to data types of scientific applications

For relational systems it is practicable to use the same efficient storage structures in all applications, as the data model supports in principle only one datatype, namely relations. This holds no longer in scientific applications where many different scientific data types with appropriate storage structures must be supported. Object-oriented database systems allow to resolve this by providing efficiently implemented data types. In the context of declarative access however the logical access of a user and the optimal physical access to the scientific data may substantially differ. This is one typical example where semantic query optimization in form of data type specific optimization is needed, as, e.g., also recognized in [17]. Another important aspect is the use of methods within declarative queries, which themselves can be complex calculations, like for the example the Needleman/Wunsch algorithm in the previous example query, but also methods containing accesses to virtually integrated databases, that might be rather expensive. In this case the cost model must reflect this special semantics of methods.

Example

Consider the following query for retrieving close atom pairs, that occurs often as a subquery of more complex requests.

```
ACCESS      atom1, atom2
FROM        atoms IN db1::Atoms,
            atom1 IN atoms->getAllAtoms()
            atom2 IN atoms->getAllAtoms()
WHERE       atoms->distance(atom1,atom2) <= 3.5     AND
            atoms.model.protein.name == '1atr'
```

The ACCESS-clause specifies as the result of the query the indexes of the selected atom pairs. Within the FROM-clause query variables are bound to their domains, either class extensions (db1::Atoms) or results of set-valued method calls (atoms->getAtomIndices()). The condition in the WHERE-clause, expresses neighborhood by using the method call atoms->distance(atom1,atom2) <= 3.5.

Without optimization the cross product of all atoms of a protein is created. As a typical protein may contain up to 5000 atoms, this leads to a large intermediate result (up to 5000^2 = 25.000.000), for which the condition has to be evaluated. In the database an index structure for the atom neighborhood relationship is precomputed, i.e., each atom has direct links to its immediate neighbors up to a certain threshold. A method belowThreshold makes optimal use of this index structure. A semantic optimization rule, that can be included by our techniques into the rule-based algebraic query optimizer, transforms the algebraic representation of this query automatically to one that corresponds to the following query

```
ACCESS    atoms->belowThreshold(atoms->getAllAtoms(),
                                atoms->getAllAtoms(),
                                3.5)
FROM      atoms IN db1::Atoms,
WHERE     atoms.model.protein.name == '1atr'
```

Notice that the transformed query is already much closer to an operational specification of the query evaluation, and - if issued by the user - would require for the formulation considerable knowledge about internal data representation and access mechanisms. The declarative flavor of the original query is lost.

Similar indexing techniques have been also used for supporting efficient substructure search in small molecules.

Further Research

We plan to apply the same techniques that were used to optimize queries that access specialized, physically optimized datastructures for the access to virtually integrated data in the future. Further aspects to be considered in the context of efficient access to biomolecular databases are the optimization of queries within query sessions, by exploiting previous queries as well as by anticipating forthcoming queries. Another interesting research issue is the design of user-assisting facilities in query processing to correct and complement user queries. First considerations on how such facilities can be integrated with our query processing approach were given in [4].

References

[1] Aberer, K.: Demand-Driven Database Integration for Biomolecular Applications, in [8] .

[2] Aberer, K.: The Use of Object-oriented Data Models in Biomolecular Databases, Proc. of the Conf. on Object-Oriented Computing in the Natural Sciences OOCNS, 1994.

[3] Aberer, K., Fischer, G.: Semantic Query Optimization for Methods in Object-Oriented Database Systems. International Conference on Data Engineering, March 1995, Taipei, Taiwan.

[4] Aberer, K., Klas, W., Furtado, A.L.: Designing a User-Oriented Query Modification Facility in Object-Oriented Database Systems, Proceedings of CAiSE*94, Utrecht, June 1994.

[5] Altschul, S.F., Gish, W.Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool, J. Mol. Biol., 215, pp. 403-410, 1990.

[6] Durbin, R., Thierry-Mieg, J.: The ACEDB Genome Database, WWW page.

[7] Graefe, G. : Volcano - An Extensible and Parallel Query Evaluation System, IEEE Transactions on Knowledge and Data Engineering, Vol. 6, No. 1, Feb. 1994, pp. 120-135.

[8] Karp, P. (Ed.) Final report on the 1994 Meeting on Interconnection of Molecular Biology Databases, WWW page, Stanford University, August 9-12, 1994.

[9] Kemp G.J.L., Jiao Z., Gray P.M.D., Fothergill J.E.: "Combining Computation with Database Access in Biomolecular Computing", ADB 94, Vadstena, Sweden, p.317-335.

[10] Kim, W.: Object-Oriented Database Systems: Promises, Reality and Future. In W. Kim (Ed.), Modern Database Systems, ACM Press, 1995.

[11] Lamb, C., Landis, G., et al.: The ObjectStore Database System, Comm. of the ACM, Vol. 34, No. 11, p 32-39, 1991.

[12] Maier, D.: The Theory of Relational Databases, Computer Science Press, 1983.

[13] Needleman, S.B., Wunsch, C. A.: General Method Applicable to the Search for Similarities in the Amino Acid Sequences of Two Proteins, Porc. of the National Academy of Science, Vol. 48, p 444-453, 1970.

[14] NCBI Entrez, WWW page.

[15] Vriend, G., Sander, C., Stouten, P.F.W.: A novel search method for protein sequence-structure relations using property profiles, Protein Engineering, vol. 7, no.1, pp. 23-29, 1994.

[16] VODAK release 4.0, obtainable via anonymous ftp from ftp.darmstadt.gmd.de, directory /pub/dimsys/vodak/vodakV4R0.tar.Z.

[17] R. Wolniewciz, G. Graefe: Algebraic Optimization of Computations over Scientific Databases, Proc. of the 19th VLDB Conf., Dublin, Ireland, 1993, pp. 13-24.