Protein construction-based data partitioning scheme for alignment of protein macromolecular structures through distributed querying in federated databases

Dariusz Mrozek, Senior Member, IEEE EMBS, SMC Jacek Kwiendacz and Bożena Malysiak-Mrozek, Member, IEEE CI, SMC

Abstract—Exploration of various characteristics of 3D protein structures through querying relational databases storing the structures can be challenging due to the necessity to conform to a particular database schema. However, this also brings several advantages, like the ability to perform extensive database searches with declarative SQL language, protect data against hardware damages through regular backup mechanisms, and secure data against unauthorized access. Since relational databases do not provide exploration methods specific for protein data and its biological semantics, like searches on the basis of protein structural patterns, the use of relational databases in this domain is still rare and requires the development of dedicated methods to increase the speed of data exploration techniques.

In this paper, we show a novel data partitioning scheme for distributing data across database clusters that can be used for performing sophisticated explorations of 3D protein structures. The data partitioning scheme relies on protein construction, which requires data preprocessing, but results in shorter exploration times through querying federated databases. We solve the problem of finding proteins in Oracle relational database on the basis of the similarity of 3D protein structures with the use of distributed PAR-P3D-SQL queries. Since 3D protein structure similarity searching is one of the most time-consuming exploration processes that can be performed for protein data, we make use of a distributed environment of Oracle federated databases, distributed query processing, and dedicated load balancing methods to accelerate the exploration. Results of performed tests confirm that we are able to significantly increase the speed of the exploration process, proportionally to the number of database nodes in the federated environment.

Index Terms—Bioinformatics, Proteins, Federated databases, Data partitioning, Distributed querying, 3D protein structures, Similarity Search, Structural alignment

I. INTRODUCTION

3D protein structures reveal every detail of protein construction by showing how atoms of various chemical elements are arranged in the construction and how they are linked by covalent bonds. Studying these atomic arrangements is important for discovery of molecular mechanisms of many cellular processes, including those performed in normal cell life cycles, as well as those responsible for the emergence of serious civilization diseases [1]. Therefore, exploration of 3D protein structures is of great importance while getting knowledge on the life machinery of living organisms, studying reasons for various diseases, and designing drugs to cure them [2]. A significant role in the exploration of protein structures play comparative approaches. Comparative approaches allow drawing useful conclusions, e.g., on the role and the function of a protein molecule in a cellular process, on the basis of the similarity of the molecule to another one, for which the function is already recognized. Comparative approaches can also be used in modeling 3D protein structures on the basis of sequence similarity of the modeled molecule to one of the known structure [3]. They also allow finding substitutes for particular protein molecules or protein drugs that possess improved enzymatic capabilities or better membrane penetration.

3D protein structure similarity searching is one of the comparative approaches that enable protein classification, functional annotation, and plays a supportive role while verifying results of predictions of 3D protein structures [4], [5]. Scientists have access to a variety of tools by which they can carry out the 3D protein structure similarity searching. Most of the tools are dedicated web pages, like DALI [6], [7] and VAST [8], [9], and there are a few desktop applications, e.g., RCSB PDB Protein Comparison Tool [10]. However, performing the process through these tools leaves a very limited control on the data and the process, and makes it difficult to reprocess the data or obtained results. Moreover, as rightly pointed out by Gesing et al. [11], in many cases, possessed peptide chains or protein conformations, either predicted or obtained experimentally [12], may constitute an intellectual property that cannot be simply shared through the Internet. They may be an important element in the drug production process and a decisive factor in the profits of a pharmaceutical company. In such environments, there is a need to store data securely, and process the data effectively and efficiently on hardware kept on premises.

A. Challenges for Storing Protein Data in Relational Databases

For scientists that investigate protein structures and functions, one of the serious alternatives for collecting and processing protein macromolecular data are relational databases [13]. Relational databases collect data in tables describing part of its biological semantics, like searches on the basis of protein structural patterns, the use of relational databases in this domain is still rare and requires the development of dedicated methods to increase the speed of data exploration techniques.

In this paper, we show a novel data partitioning scheme for distributing data across database clusters that can be used for performing sophisticated explorations of 3D protein structures. The data partitioning scheme relies on protein construction, which requires data preprocessing, but results in shorter exploration times through querying federated databases. We solve the problem of finding proteins in Oracle relational database on the basis of the similarity of 3D protein structures with the use of distributed PAR-P3D-SQL queries. Since 3D protein structure similarity searching is one of the most time-consuming exploration processes that can be performed for protein data, we make use of a distributed environment of Oracle federated databases, distributed query processing, and dedicated load balancing methods to accelerate the exploration. Results of performed tests confirm that we are able to significantly increase the speed of the exploration process, proportionally to the number of database nodes in the federated environment.

Index Terms—Bioinformatics, Proteins, Federated databases, Data partitioning, Distributed querying, 3D protein structures, Similarity Search, Structural alignment

I. INTRODUCTION

3D protein structures reveal every detail of protein construction by showing how atoms of various chemical elements are arranged in the construction and how they are linked by covalent bonds. Studying these atomic arrangements is important for discovery of molecular mechanisms of many cellular processes, including those performed in normal cell life cycles, as well as those responsible for the emergence of serious civilization diseases [1]. Therefore, exploration of 3D protein structures is of great importance while getting knowledge on the life machinery of living organisms, studying reasons for various diseases, and designing drugs to cure them [2]. A significant role in the exploration of protein structures play comparative approaches. Comparative approaches allow drawing useful conclusions, e.g., on the role and the function of a protein molecule in a cellular process, on the basis of the similarity of the molecule to another one, for which the function is already recognized. Comparative approaches can also be used in modeling 3D protein structures on the basis of sequence similarity of the modeled molecule to one of the known structure [3]. They also allow finding substitutes for particular protein molecules or protein drugs that possess improved enzymatic capabilities or better membrane penetration.

3D protein structure similarity searching is one of the comparative approaches that enable protein classification, functional annotation, and plays a supportive role while verifying results of predictions of 3D protein structures [4], [5]. Scientists have access to a variety of tools by which they can carry out the 3D protein structure similarity searching. Most of the tools are dedicated web pages, like DALI [6], [7] and VAST [8], [9], and there are a few desktop applications, e.g., RCSB PDB Protein Comparison Tool [10]. However, performing the process through these tools leaves a very limited control on the data and the process, and makes it difficult to reprocess the data or obtained results. Moreover, as rightly pointed out by Gesing et al. [11], in many cases, possessed peptide chains or protein conformations, either predicted or obtained experimentally [12], may constitute an intellectual property that cannot be simply shared through the Internet. They may be an important element in the drug production process and a decisive factor in the profits of a pharmaceutical company. In such environments, there is a need to store data securely, and process the data effectively and efficiently on hardware kept on premises.

A. Challenges for Storing Protein Data in Relational Databases

For scientists that investigate protein structures and functions, one of the serious alternatives for collecting and processing protein macromolecular data are relational databases [13]. Relational databases collect data in tables describing part of

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.
realities, where data are arranged in columns and rows. Modern relational databases also provide a declarative query language – SQL that allows retrieving and processing collected data on the basis of specified filtering criteria. The SQL language gained great power in processing regular, transactional data in real-world production systems since it hides details of the processing under a quite simple SELECT statement. This allows to move the burden of the processing to database management systems (DBMSs) and leaves more freedom in managing the workload.

Unfortunately, little has been done to allow the analysis of protein macromolecular data with the use of declarative SQL language in relational databases. One of the reasons for this can be the way how protein macromolecular data or genetic data are stored natively. These are usually flat files with many records of different types, which definitely have a non-relational form. Macromolecular data of 3D protein structures are natively stored in PDB files [14], mmCIF files [15], and PDBML files [16], which are available from the Protein Data Bank [17]. Although there were some efforts, like the OpenMMS toolkit and the PDBase toolkit [18], to provide access to the macromolecular data through the SQL-based interface, all works have been suspended. Efforts have been redirected to BioJava [19] library for processing flat files locally with the use of Java applications developed by users. This gave a lot of flexibility, but it does not allow for extensive string searches in PDB files, it requires an in-depth knowledge of the BioJava library, and it definitely is not declarative. Moreover, it does not provide any mechanisms to secure the protein data in case of hardware damage, like regular backups in relational databases, and to control access to the data to prevent unauthorized use of the data, like the mechanisms of users, security roles, and privileges.

Another reason why relational databases are not frequently used in storing and processing protein macromolecular data may be the fact that SQL queries work very well while retrieving data on the basis of filtering criteria imposed on columns of simple types, e.g., text or numeric. The LIKE operator that can be used in the WHERE clause of the SELECT statement can even perform a flexible, approximate (not precise) string searching and find similar data, but it does not allow performing approximate searches for protein data, like protein sequence similarity searching or 3D protein structure similarity searching. In other words, standard SQL does not support searches on the basis of biological patterns, like protein amino acid sequence, 3D protein structure or a part of it. These kinds of searches must take into account the specificity of protein data. Consequently, they require completely different algorithmic approaches for flexible searching that are based on sequence alignment [20], [21] or structure alignment [22] (Fig. 1).

B. Related Works

Actually, the problem of storing macromolecular data of proteins and DNA/RNA molecules in relational DBMSs and having appropriate methods, and the query language that would allow processing the data has been noticed in the last decade. Besides the mentioned PDBase toolkit that allowed loading macromolecular data of proteins into the relational databases for various purposes (mainly extended text searches), there are only a few initiatives in the world reporting this kind of solutions. They usually focus on various types of protein data describing molecules on various description levels of their structures (primary-, secondary-, or tertiary structure). For example, ODM BLAST [28] is an implementation of the BLAST family of methods [24] in the commercial Oracle database management system. The ODM BLAST extends the SQL language by providing appropriate functions for local alignment and similarity searching of DNA/RNA and protein amino acid sequences. The ODM BLAST works efficiently, but in terms of protein molecules, its capabilities are limited only to the primary structure (not 3D protein structures). BioSQL [29], which makes use of particular modules of the BioJava library [19], provides extended capabilities by focusing on bio-molecular sequences and various features of bio-molecules, their annotation, a reference taxonomy, and ontologies. However, likewise the ODM BLAST, the BioSQL does not allow exploration of 3D protein structures. Several solutions in terms of declarative querying can be found for secondary structures. For example, in [30], Hammel and Patel describe their search engine and the extension to the SQL language, which allow searching on the secondary structures of protein sequences. Authors have developed a dedicated search engine (Periscope) and an extension to the Oracle commercial database system. Both solutions allow searching proteins based on their secondary structures represented as sequences of characters. In [31], Tata et al. show the Periscope/SQ extension of the Periscope system. Periscope/SQ is a declarative tool for querying primary and secondary structures of protein molecules. In their solution, authors introduced a new language PiQL, new data types, and algebraic operators, according to the defined query algebra PiOA. In the paper [32], authors present their extensions to the object-oriented database (OODB) by adding Protein-QL query language and the Protein-OODB middle layer for requests submitted to the OODB. The ProteinQL allows the formulation of simple queries that operate on the primary, secondary and tertiary level. In [33] and [34], we reported the PSS-SQL (Protein Secondary Structure - Structured Query Language) for searching protein similarities on the basis of their secondary structures in the Microsoft SQL Server relational database. The newest version of the PSS-SQL search engine [35] utilizes a multi-threaded alignment procedure, which allows effective querying on DBMSs hosted on computers with multi-core CPUs. However, capabilities of the PSS-SQL are limited to processing only the secondary structures of proteins. Finally, in [36], we proposed the first SQL-based query language for advanced exploration of 3D protein structures in Oracle relational databases and finding proteins that share common 3D folds. The query language that we proposed, called P3D-SQL, allows processing 3D protein structures and finding proteins on the basis of the similarity of their 3D structures (tertiary structures). Although, P3D-SQL is the only declarative query language that allows for advanced exploration of protein structures, including structural alignment and 3D structure similarity searching, its main
drawback is efficiency. The solution presented in this paper addresses this problem through the distributed execution of explorative queries in a federated database environment.

C. Querying with P3D-SQL

In this section, we show the basic details of the P3D-SQL language. P3D-SQL [36] is the extension to the Oracle PL/SQL language, which allows finding and retrieving proteins having the same or similar 3D structures to the structure of the specified protein. P3D-SQL queries operate on 3D protein structures stored in a relational table in the Oracle database. The similarity of 3D protein structures is assessed on the basis of various measures calculated after the completion of structural alignment and superposition of the given query structure and successive structures from the database (or specified part of them). The structural alignment and superposition of 3D proteins structures are performed by the invocation of particular P3D-SQL functions, namely FATCAT_ALIGN and CE_ALIGN. These P3D-SQL functions implement the FATCAT [37] and the CE [38] methods for structure matching known from the literature and briefly described in Sect. I-D.

A typical P3D-SQL query has specific, complex construction. However, the construction allows specifying all criteria for finding similar proteins flexibly on the basis of the given structural pattern (usually another protein structure or a part of it). Sample query showing the invocation of the FATCAT method is presented in Listing 1.

```
1 SELECT dbPDBID, alignscore, similarity, totalRMSD
2 FROM TABLE(FATCAT_ALIGN(
3 CURSOR(SELECT pdbid, structure FROM Protein
4 WHERE pdbid = '1N6H'),
5 CURSOR(SELECT pdbid, structure FROM Protein
6 WHERE pdbid BETWEEN '1N6A' AND '1N6Z'),
7 PRINT => 1, ALGORITHM_TYPE => 2)
8 WHERE totalRMSD < 4.0
9 ORDER BY alignscore DESC;
```

Listing 1. Sample P3D-SQL query displaying proteins (from the range of pdbid between 1N6A and 1N6Z) that are similar to the given protein (1N6H) with the total root mean square deviation (totalRMSD) lower than 4.0Å. Results are sorted by score (alignscore) measure in descending order.

This query displays proteins from the database (from the range of pdbid between 1N6A and 1N6Z) that are similar to the given protein structure (1N6H). The query selects identifiers of protein structures (dbPDBIDs) and displays three similarity measures showing how the selected proteins are similar to the given (query) protein structure. The structural alignment between the given protein structure 1N6H and the database structures (1N6A to 1N6Z) is performed after calling the FATCAT_ALIGN function that returns a table for the FROM clause of the main SELECT statement. The given (query) protein is specified in the first argument of the function, as a cursor. The nested SELECT statement of the cursor must retrieve the identifier (here pdbid field) and
the 3D structure (structure field) of the given (query) protein, for which we want to find similar structures. The query protein structure can be stored in any table of the database (here in the Proteins table). The search criteria specified in the WHERE clause of the cursor SELECT statement (line 4) enable to point to a proper protein structure in the database. The range of database, candidate protein structures that should be compared to the given query structure is specified by the second cursor and its nested SELECT statement. The search criteria in the WHERE clause of this SELECT statement (line 6) allow to precisely specify proteins that will be matched to the query protein by imposing filtering criteria before the alignment begins. These filtering criteria can involve various characteristics of proteins (e.g., identifiers, length, organism), if they are stored in source tables for the query and the candidate protein structures. Moreover, the query protein structure and explored candidate structures can be stored in different tables. Cursors allow to point to appropriate tables through the nested SELECT statements and return tables of appropriate proteins that will be processed individually by the invoked alignment function (e.g., the FATCAT_ALIGN). The P3D-SQL extension for Oracle database assumes that protein structures are stored in PDB format in table fields of CLOB data type (CLOB, a Character Large Object). The additional parameter of the alignment function, PRINT=>1, enables the generation of the HTML document, like the one presented in Fig. 6, showing a detailed alignment for each pair of the query structure and a candidate database structure. The parameter ALGORITHM_TYPE=>2 determines the use of a flexible version of the FATCAT, which enables twists.

Results of the alignment process are filtered in the WHERE clause of the main SELECT statement (line 8). The presented sample query returns only those candidate database proteins, for which similarity to the given structure, measured by the total RMSD, is lower than 4.0 Å. Results are also sorted according to the alignscore similarity measure in the descending order (line 9).

D. Algorithms for Flexible Matching of Protein Structures

3D protein structure similarity searching refers to the process in which a given protein structure is compared to another protein structure or a set of protein structures collected in a database or any other collection. 3D protein structure similarity searching is usually performed through the alignment of protein structures. The alignment procedure finds fragments of protein structures that match to each other, i.e., indicate high similarity according to the assumed scoring system and given objective function. P3D-SQL implements protein structure similarity searches by using two popular alignment algorithms – CE [38] and FATCAT [37]. Both algorithms are publicly available through the Protein Data Bank (PDB) [17] website for those, who want to search for structural neighbors. FATCAT and CE perform structural alignments by combining so-called Aligned Fragment Pairs (AFPs). Additionally, by entering twists, the FATCAT allows for flexible alignments, eliminating drawbacks of many existing methods that treat proteins as rigid bodies. As a result, for the number of cases, the FATCAT enables capturing actual homology that flows from the sequence similarity, which is impossible for other methods. When developing the P3D-SQL, we integrated implementations of the CE and the FATCAT algorithms included in the BioJava library [19] into the Oracle PL/SQL. The BioJava provides new, enhanced implementations of the CE and the FATCAT algorithms – jCE and jFATCAT. The jFATCAT is delivered in two variants rigid and flexible, for rigid and flexible alignments. The jCE performs a rigid-body alignment of protein structures, similar to jFATCAT-rigid. The jCE also implements the CE method with Circular Permutations (jCECP) variant, which solves the problem of handling circular permutations. This problem is typical for many alignment algorithms, including the CE and the FATCAT, that compute sequence order-dependent alignments.

3D protein structure similarity searching through structural alignment is a time-consuming process. This flows from the computational complexity of the alignment methods, which try to match fragments of protein structures optimally. This significantly affects the execution time of alignment processes performed on local computers of users and also P3D-SQL queries executed in Oracle databases, and constitutes the motivation for works presented in this paper.

E. Scope of this Work

In this paper, we present dedicated data partitioning schemes that we implemented for the purpose of distributed query execution in federated database environments. We show how these partitioning schemes are implemented in the parallel version of the P3D-SQL, called PAR-P3D-SQL, which enables distributed execution of P3D-SQL queries in the cluster of relational databases. We also present hierarchical arrangement of the architecture of the distributed database environment, integration of the PAR-P3D-SQL executables with Oracle instances, and the details of PAR-P3D-SQL query execution (Sect. II). The PAR-P3D-SQL significantly accelerates the execution of P3D-SQL queries, which is confirmed by results of performance tests presented in Sect. III. Finally, we discuss the presented solution and obtained results of performance tests comparing them to serial execution of P3D-SQL queries and other systems that have similar capabilities (Sect. IV).

II. METHODS AND IMPLEMENTATION

PAR-P3D-SQL is an extension to the Oracle PL/SQL language. It allows for distributed execution of P3D-SQL queries in federated environments of Oracle databases. In this section, we show details of the PAR-P3D-SQL extension, including its architecture and components, query execution process, designed and implemented data partitioning schemes, integration with Oracle instance, and invocation in Oracle federated database environment.

A. Distributed P3D-SQL in Oracle Database

In order to accelerate the execution of P3D-SQL queries for protein structure exploration, the PAR-P3D-SQL makes use of a distributed environment of federated Oracle databases [39],...
This environment can be created in a cluster of physical or virtual machines located on premises or on the Cloud. The system has got a master-slave organization with:
- one master database node,
- and many worker database nodes,
as presented in Fig. 2.

The master node is responsible for controlling the whole execution of distributed queries in the federated database system. In order to execute the distributed query the master node:

1) analyzes the query and splits the query job into a number of tasks (smaller portions of the job),
2) initiates the connection to each worker node through a database link,
3) initiates execution of tasks on worker nodes,
4) merges results from worker nodes into one result set.

The worker nodes hold partitioned or replicated macromolecular data of proteins and are responsible for the execution of tasks sent by the master node. Execution of each task contains the following steps:

1) establishing a connection with the master node,
2) retrieving from the master node the identifiers (PDB IDs) of protein structures that should be explored within the task,
3) execution of appropriate P3D-SQL query for finding structural similarities among given molecules,
4) saving task results on the master node.

Execution of particular tasks can take a long time since P3D-SQL queries perform time-consuming structural alignments and superpositions. Therefore, in practice, the master node may simultaneously be a worker node provided that partitioned data are also available on the master node.

3D structures of peptides or protein molecules are stored as whole PDB files in a dedicated relational table (as CLOBs) of the database located on nodes of the federated system. Macromolecular data of protein structures are fully replicated in nodes of the system presented in Fig. 2, since the execution time, not the storage space, is a critical factor for distributing computations. Then, tasks are executed against separate partitions of replicated data. An appropriate portion of data is resolved at the beginning of job execution on the basis of the job specification and the assumed partitioning scheme. This has two advantages. (1) Replicated data are protected against loss in case of damage of any of the cluster nodes. At the same time, storage space is not a problem, since the whole PDB repository located in relational Oracle database takes approximately 150GB. (2) Adding new nodes of the database cluster due to performance reasons is simplified and does not cause the necessity of data re-partitioning, which would affect all existing nodes.

B. Distributed Query Execution

In Oracle federated environment, remote databases are connected together through database links that allow establishing connections between two physical databases and enable a client to access them as one logical database. Query execution in such an environment usually relies on Distributed Partitioned Views and Distributed SQL. However, since passing the whole tables of protein structures through the database links is restricted, we cannot simply build the Distributed Partitioned Views, but we dynamically construct the Distributed SQL queries in each node of the system passing just necessary metadata through temporary structures created on each worker node. In the environment of the federated databases, P3D-SQL queries are executed in parallel as query jobs. The query job comprises a sequence of steps that are performed on the master node and all available worker nodes (Fig. 3). At the beginning of the execution process, for each of the available worker nodes, the job creates temporary data structures that are used for passing data needed for execution of the protein structure alignment process and for assembling results of the alignment. These data structures are created both, on the master and the worker nodes. In the next step, the job creates specifications of data partitions with proteins that should be explored in the same task. These specifications consist of just identifiers of protein structures. Neither real data partitioning is performed, since due to performance reasons macromolecular data are fully replicated across cluster nodes, nor whole protein structures are packed as they will be available on each worker node of the cluster. This step allows specifying how data will be explored on the basis of arguments passed by a user who executed the distributed query. We have implemented three partitioning schemes in the PAR-P3D-SQL:

1) equinumerous (EQUALLY),
2) balanced with respect to the size of the protein structure files (SIZE_B),
3) balanced with respect to the number of amino acids in explored protein structures (AMINOACIDS).

These partitioning schemes will be described in more details in Sect. II-C.

When specifications of data partitions are created, the job creates Oracle database links to the worker nodes and a number of query tasks with appropriate data specifications stored in temporary data structures. Tasks are then executed in parallel on worker nodes. Each query task consists of the dynamically build, parameterized P3D-SQL query, which is similar to the one presented in Listing 1. The query executes the same alignment process, but for a different range of protein structures from the repository.

During task execution, the task establishes the connection to the master node, retrieves appropriate specification of the data partition, specification of other arguments of the distributed query, and saves this information in temporary data structures created on the worker side. Then, it runs a P3D-SQL query parameterized according to the retrieved information. The P3D-SQL query searches for protein structure similarities by performing structural alignments of the specified protein structure to structures defined in the specification of data partition assigned to the query task. The outcome of the P3D-SQL query is stored locally in temporary data structures, and then, transferred and saved on the master node. At the end of task execution, all temporary data structures are removed.

When all worker nodes finish execution of the assigned task, the master node merges results saved by particular tasks
stored in temporary data structures and presents them to the user. Results include a table of candidate protein structures that were compared to the given protein structure and several similarity measures used in protein comparison, e.g., score, similarity, identity, p-value, probability, coverage, and others. Optionally, if priorly specified by the user in query execution arguments, results may also include visualizations of structural alignment, which are stored as HTML files in a specified folder on the master node. Finally, all temporary data structures on the master node are removed to clean the environment, and the execution of the query job is finished.

C. Data Partitioning Schemes

Data partitioning schemes decide how protein structures will be assembled in data partitions and processed together by one task.

PAR-P3D-SQL operates on the replicated repository of 3D protein structures $R$:

$$R = \{ S^D_v | v = 1, 2, ..., |R| \} ,$$  

where $S^D_v$ is the $v$-th 3D protein structure, and $|R|$ is the number of protein structures in the repository.

In practice, imposed filtering conditions in the WHERE clauses of the cursor section of PAR-P3D-SQL queries cause the exploration of just a subset $R^Q$ of the repository $R$:

$$R^Q \subseteq R .$$  

These subsets can vary for different queries submitted for execution:

$$\Omega_R = \{ R_1, R_2, ..., R_{|\Omega_R|} \} ,$$  

where $\Omega$ is the space of all possible subsets of the repository $R$ and $|\Omega_R|$ is the number of all possible subsets of the repository $R$.

Data partition $DP$ is a subset of the explored part of repository $R^Q$ and a collection of 3D protein structures:

$$DP \subseteq R^Q ,$$  

$$DP = \{ S^D_v | v = 1, 2, ..., |DP| \} \leq |R^Q| \leq |R| .$$

where $|DP|$ is the number of proteins in the data partition $DP$, and $S^D_v$ is the $v$-th protein structure assigned to the partition.

The following partitioning schemes have been designed and implemented in the PAR-P3D-SQL:

1) equinumerous (EQUALLY),
2) byte-size-balanced, i.e., balanced with respect to the size of the protein structure files (SIZE_B),
3) protein-size-balanced, i.e., balanced with respect to the number of amino acids in explored protein structures (AMINOACIDS).

The equinumerous partitioning scheme divides the query specific explored part of the repository $R^Q$ into $N$ data partitions of, on average, $|DP|$ protein structures:

$$|DP| = \left\lfloor \frac{|R^Q|}{N} \right\rfloor ,$$  

in such a way that the number of protein structures in each data partition is equal (or almost equal due to indivisibility of integers):

$$\forall DP_m, DP_n \subseteq R^Q \Rightarrow |DP_m| \approx |DP_n| .$$

The number of partitions $N$ in eq. 6 equals the number of worker nodes in the federated database system.

The equinumerous partitioning scheme is the easiest partitioning scheme. Its advantage is that it does not require any pre-processing of the explored part of the repository $R^Q$. However, query tasks may operate on data partitions of unbalanced sizes.
The byte-size-balanced partitioning scheme requires some pre-processing of data and collecting additional statistics of byte size of protein structures, which can be easily retrieved from the table field storing these structures. This data pre-processing should be performed before execution of the first query in the federated database system, and collected statistics should be updated with changes in the protein data.

The byte-size-balanced partitioning scheme assumes that larger proteins require more data for their description, which is reflected in the byte size of the data stored in a relational table. While this is true in many cases, we can also find proteins that have many models that are stored together, thus, increasing the size of the data describing a single database entry (protein structure). Therefore, we also propose the protein-size-balanced partitioning scheme that takes into account the basic construction of proteins built up with amino acids:

\[ S^{3D} = \{ P_w^{3D} | w = 1, 2, ..., |S^{3D}| \} \],

where \( S^{3D} \) is the protein structure built up with \(|S^{3D}|\) amino acids (residues) \( P^{3D} \).

It is worth noting that the number of amino acids in each protein may vary significantly. The protein-size-balanced partitioning scheme takes this fact into account and divides the query specific explored part of the repository \( R^Q \) into \( N \) data partitions in such a way that the number of amino acids in all data partitions is balanced:

\[
\min_{DP_m, DP_n \subseteq R^Q} \left| \sum_{S^{3D} \in DP_m} |S^{3D}| - \sum_{S^{3D} \in DP_n} |S^{3D}| \right|. \tag{11}
\]

Likewise, the byte-size-balanced partitioning scheme, the protein-size-balanced partitioning scheme also requires some pre-processing of data and collecting additional statistics on the number of amino acids in protein structures. This requires additional parsing of the structural data stored in the field of the relational table with protein data and is done with the use of functions of the BioJava library installed in each Oracle node. The data pre-processing should be performed before execution of the first query in the federated database system, and collected statistics should be updated with changes in the protein data.

In all data partitioning schemes the following condition holds:

\[
\forall_{DP_m, DP_n \subseteq R^Q} \, \, \, DP_m \cap DP_n = \emptyset, \tag{12}
\]

which means that any protein structure may be assigned to only one partition.

D. Integration with Oracle Instance

From the technical point of view, the PAR-P3D-SQL extension for Oracle is a set of libraries containing Java and PL/SQL procedures and functions that altogether allow for distributed execution of protein structure exploration queries in federated relational Oracle databases. Before execution of any protein exploration query, the bytecode of PAR-P3D-SQL extension must be loaded into Oracle Database instance and published to PL/SQL. After loading the bytecode, the executable form of PAR-P3D-SQL and dependent libraries are held in the Library

---

Fig. 3. Execution of distributed queries in federated Oracle databases during exploration of 3D protein structures (M – master node, W – worker node).
Cache in the System Global Area (SGA) of the database instance (Fig. 4).

The PAR-P3D-SQL extension for Oracle consists of three modules:
1) PAR-P3D-SQL executables responsible for distributed execution of the exploration process,
2) P3D-SQL executables responsible for the execution of SQL-based queries on worker nodes,
3) BioJava executables responsible for particular operations performed on protein structures.

The PAR-P3D-SQL executables make an Oracle node capable of performing protein structure exploration in parallel. These executables must be installed on each node of the federated database, both master and worker. The PAR-P3D-SQL module consist of Java classes and methods, as well as PL/SQL procedures and functions that allow to link databases in the federated environment, create temporary data structures on master and worker nodes, create specifications for data partitions, divide the exploration job into query tasks, and finally, run the query tasks on worker nodes and collect results.

The P3D-SQL executables allow for the execution of single P3D-SQL queries associated with particular tasks on the Oracle worker node. The module allows navigation to particular relational tables with protein structures and retrieving particular protein structures from the Oracle database by using appropriately formulated SQL queries. It also exposes the SQL interface with call specifications of necessary exploration methods, including the CE_ALIGN and the FATCAT_ALIGN PL/SQL functions for protein structure alignment.

The BioJava executables allow to parse and explore a particular protein structure retrieved by a SQL query or compare a pair of protein structures to each other.

The distributed exploration job is created once a user invokes the procedure called RUN_P3DSQL_PARALLEL as presented in Listing 2.

```
1 call RUN_P3DSQL_PARALLEL(4, 'ORAMASTER', '1521', 'ProteinDB', 'DBUser', 'DBPass',
2   'SELECT pdbid FROM PROTEINS WHERE pdbid = '1n6h' ',
3   'SELECT pdbid FROM PROTEINS WHERE pdbid BETWEEN '1n60' AND '1n6z' ',
4   'FATCAT', 'AMINOACIDS', 'PRINT => 1', '/*myResults*/');
5 SELECT * FROM Results;
```

Listing 2. Sample invocation of RUN_P3DSQL_PARALLEL procedure for distributed exploration of protein structures.

The procedure accepts several important arguments, including: the degree of parallelism (value 4, line 1), SID of the master node (ORAMASTER), the master node port (1521), the name of the database with protein structures (ProteinDB), the name of the database user and its password (DBUser, DBPass). Then, two SELECT statements (lines 2-4) determine which protein structures will be compared (these SELECT statements will be passed for building the cursors that process each protein structure individually, like in Listing 1). In the example, the given protein structure 1N6H (PDB ID code) will be compared to all proteins that fall into the range between 1N60 and 1N62 (the explored part of the repository). The given structure (1N6H) and the range of compared molecules may come from various tables in the database, and they can be retrieved on the basis of more complicated filtering criteria. Hence, the two SELECT operations that provide some flexibility in retrieving appropriate structures from the database. The range of molecules retrieved by the SELECT statement in lines 3-4 (the explored part of the repository) will be partitioned while executing the exploration job. Other arguments (line 5) specify: the name of the alignment algorithm that will be used in protein comparison (FATCAT), data partitioning scheme (AMINOACIDS), and whether the alignment results should be visualized and projected on the protein sequences (PRINT), and the location, where HTML files with visualization reports should be stored.

III. EXPERIMENTAL RESULTS

The PAR-P3D-SQL extension for Oracle PL/SQL was extensively tested in order to verify its effectiveness and performance. The main goal of the experiments was to:

- verify how the performance of PAR-P3D-SQL queries depends on the used data partitioning scheme;
- check the consistency of results returned by the PAR-P3D-SQL queries and P3D-SQL;
- evaluate the performance of PAR-P3D-SQL queries and compare it to the performance of serial P3D-SQL queries;
- verify how the performance depends on the heterogeneity of the database;
- find out how scalable the PAR-P3D-SQL is;
- compare the efficiency of the PAR-P3D-SQL queries to other distributed systems known from the literature.

A. Runtime Environment

The distributed environment of Oracle federated databases was established on nodes of a virtualized cluster controlled by the HyperV hypervisor hosted on Microsoft Windows 2008 R2 Datacenter Edition 64-bit. The cluster contained 1 master node, and 1, 2, 4, or 8 database (worker) nodes, depending on the phase of testing. The master node was also one of the worker nodes. The host server had the following parameters: 2x Intel Xeon CPU E5620 2.40 GHz, RAM 96 GB, 3x HDD 1TB 7200 RPM. Cluster nodes were configured to use 4 virtual CPUs and 4GB RAM per node and worked under the Microsoft Windows 2008 R2 Enterprise Edition 64-bit operating system. The PAR-P3D-SQL extension was installed on each of the worker nodes with Oracle 11gR2 Enterprise Edition database. Such a prepared federated environment was used to run distributed PAR-P3D-SQL queries and carry out assumed performance experiments.

B. Data Sets

Execution time of distributed PAR-P3D-SQL queries is strongly related to the size and construction of explored protein structures. In our performance experiments we used two subsets of the whole repository ($|R| = 93,043 protein structures) to prove this fact and to verify how it influences the execution process:

1) Homogeneous data set $R^H$ containing 3D structures for 1,000 proteins from the repository ($|R^H| = 1,000$) with
aminino acid chains between 80 and 100 residues long (80 ≤ |S^{3D}| ≤ 100) and byte size between 500kB and 600kB each (500kB ≤ sizeof({S^{3D}}) ≤ 600kB).

2) Heterogeneous data set R^Q containing 3D structures for 1,000 proteins from the repository (|R^Q| = 1,000) with amino acid chains between 14 and 100 residues long (14 ≤ |S^{3D}| ≤ 100) and byte size between 128kB and 600kB each (128kB ≤ sizeof({S^{3D}}) ≤ 600kB).

Various characteristics of these two explored parts of the repository allowed us to better assess various data partitioning schemes.

C. A Course of Experiments

For both data sets described in Sect. III-B, we performed a series of consistency and performance tests with the use of (1) serial P3D-SQL queries on a single instance of the Oracle database, and (2) distributed PAR-P3D-SQL queries in the federated environment of Oracle databases. The exploration process involved protein structure comparison and finding structural similarities among many proteins (database search) in the data sets through structural alignment, which is usually a very time-consuming process. The structural alignment was carried out with the use of the FATCAT and the CE methods exposed through P3D-SQL interfaces for relational databases. While testing distributed PAR-P3D-SQL queries we used a single master node and 1, 2, 4, and 8 worker nodes (the master node was used as one of the worker nodes).

In all performance tests, we measured the execution time of performed queries. Each test was carried out at least three times, and the average execution times are presented in the following sections.

D. Consistency of Results

The consistency of results obtained in both, serial (P3D-SQL) and distributed (PAR-P3D-SQL), environments was verified by comparing the rowsets returned by particular queries. The experiments confirmed that results returned by P3D-SQL queries and PAR-P3D-SQL queries were completely consistent in terms of the returned molecules and similarity measures.

Partial results for the sample PAR-P3D-SQL query job from Listing 2 are presented in Fig. 5.

Particular fields of the presented result set have the following meaning:

- **QPDBID** - PDB identifier of the given (query) protein structure,
- **DBPDBID** - PDB identifier of the candidate database protein structure,
- **SCORE** - raw alignment score,
- **SIMILARITY** - % sequence similarity in the alignment,
- **IDENTITY** - % sequence identity in the alignment,
- **PVAL** - p-value of this alignment (FATCAT) or z-score (CE),
- **RMSD** - Root Mean Square Deviation value of the alignment,
- **CA1LEN** - length of the query protein structure (the number of amino acids),
- **CA2LEN** - length of the candidate protein structure (the number of amino acids),
- **CA1COVRGE** - the coverage, or %, of aligned residues in the chain of the query protein structure,
- **CA2COVRGE** - the coverage, or %, of aligned residues in the chain of the candidate protein structure.

Similarity measures, like score, similarity, identity, p-value are typical for the used algorithm. Structural alignment for a pair of compared protein structures can also be observed in a detailed report, like the one presented in Fig. 6. The report shows the structural alignment projected on amino acid sequences of the compared proteins. This projection reveals parts of 3D protein structures that are structurally equivalent with identical residues, structurally equivalent with similar residues (in terms of physio-chemical properties), and
structurally equivalent without any similarity of corresponding residues. This report also allows observing gaps in the alignment (marked as the ‘-’ symbol).

E. Performance of PAR-P3D-SQL

Performance of PAR-P3D-SQL jobs was verified in a series of tests and compared to the performance of serial P3D-SQL queries. Results of performance tests carried out for both, heterogeneous and homogeneous, data sets are presented in Figs. 7 and 8. Left parts of the figures show execution times for both tested alignment algorithms (FATCAT and CE), right parts present achieved speedups when scaling out the environment (adding worker nodes). The upper parts of the figures show results for the FATCAT algorithm, and the bottom parts show results for the CE algorithm. In all tested cases, when scaling out the distributed environment, we achieved a significant speedup of query execution with the use of the PAR-P3D-SQL over the serial P3D-SQL queries. For example, the best 7.62-fold speedup was achieved for the system of federated databases with 8 worker nodes when performing exploration of the homogeneous data set of protein structures with the use of the FATCAT algorithm, when the data set was divided with respect to the length of protein structures (AMINOACIDS partitioning scheme). In this configuration, we were able to reduce the execution time from 7 hours and 12 minutes (serial P3D-SQL) to 56 minutes.

Out of the three implemented data partitioning schemes, the AMINOACIDS partitioning scheme turned out to be the most efficient in terms of the execution time, thus, leading to much better n-fold speedups over the other data partitioning schemes, especially for the heterogeneous data set. The worst results were obtained when dividing protein structures from the heterogeneous data set into equinumerous data partitions – 4.27-fold speedup for the CE and 4.60-fold speedup for the FATCAT when working in the 8-node database environment.

This was somehow expected, since proteins may significantly vary in lengths. For the homogeneous data set of explored proteins the execution times of PAR-P3D-SQL jobs were similar (Fig. 8), especially for AMINOACIDS and SIZE_B data partitioning schemes, since proteins had similar sizes and similar lengths. Speedups are all the more imperfect, the more unbalanced data partitions are, which is clearly visible for the heterogeneous data set divided according to the equinumerous (EQUALLY) data partitioning scheme.

Exploration of 3D protein structures in Oracle database (both stand-alone and federated) with the use of the CE algorithm is much slower than with the FATCAT algorithm. For example, for the PAR-P3D-SQL query job executed on the system of federated databases with 8 worker nodes against the heterogeneous data set with the use of the AMINOACIDS data partitioning scheme, the exploration took 2 hours when using the CE algorithm, and ~ 40 minutes when using the FATCAT algorithm (Fig. 7, left). The same tendency can be observed for the homogeneous data set, other data partitioning schemes, and different sizes of the database environment (Fig. 7 and Fig. 8). This is caused by the fact that the CE algorithm needs more memory, which is limited in the execution environment of the Oracle instance.

The system of federated databases with only one worker node, which actually performs a serial execution of exploration job, is usually a little bit slower than stand-alone Oracle database with standard (also serial) P3D-SQL. This is visible in Figs. 7 and 8 (left), and was expected, since execution of PAR-P3D-SQL exploration jobs on a single worker node requires some additional operations to be performed, like creation of additional temporary structures, transferring data to the structures, retrieving results, and deleting temporary data structures.

F. Comparison to Other Distributed Systems

We also compared the performance of the PAR-P3D-SQL queries to massive similarity searches performed by other distributed systems for 3D protein structure alignment (Fig. 9): Cloud4PSi [43], CloudPSR working in two scheduling schemes S1 and S2 [44], the system developed by Hung and Lin [45], and HDInsight4PSi [46]. Performance is expressed in terms of Structural Alignment Speed [46] which measures a throughput:

$$\text{Speed}_{SA} = \frac{|S^3_D| \times \sum_{S^3_D \in R_Q} |S^3_{Q_i}|}{T},$$

where $|S^3_{Q_i}|$ denotes the number of residues (amino acids) in the query protein structure, the sum in the nominator denotes the total number of residues in the explored part of the repository of protein structures (the database or the data set), and $T$ is the execution time. The speed, expressed in $(\text{residues}^3/s)$, shows how many residue-to-residue comparisons are performed in a time unit and allows measuring the performance of protein alignments regardless of the used collection of protein structures.

As can be observed in Fig. 9 the structural alignment speed achieved by the PAR-P3D-SQL is comparable to those achieved by Hung & Lin’s system and the CloudPSR system.
Fig. 6. Result of structural alignment of two protein structures: 1N6H, chain A [41] and 1N6A [42] projected on their amino acid sequences. Visible structurally equivalent and identical residues (|), structurally equivalent and similar residues (:), structurally equivalent, but not similar residues (.), and gaps (–).

... Structurally equivalent and identical residues
... Structurally equivalent and similar residues
... Structurally equivalent, but not similar residues

working in S1 and S2 scheduling schemes. Structural alignment speeds achieved by the other two systems, namely the Cloud4PSi and the HDInsight4PSi, are much better in all cluster configurations. This confirms that running structural alignments of protein structures within SQL queries comes as a cost of performance. This was expected, since the amount of memory for the PAR-P3D-SQL executables and Java Virtual Machine used by them, all running within an Oracle instance, is limited. However, declarative data processing has got many advantages that may balance out the imperfection. They will be discussed in the next section.

IV. DISCUSSION

By implementing the PAR-P3D-SQL extension for Oracle PL/SQL we proved that sophisticated exploration of protein data is possible not only through various computational frameworks but also in relational databases. 3D protein structure similarity searching can be performed in a declarative SQL query language in Oracle databases to support various analyses, including protein identification, finding common structural motifs or protein regions responsible for important cellular processes, and verification of predicted protein models.

By using distributed PAR-P3D-SQL queries we were able to accelerate the exploration of protein structures proportionally to the number of database (worker) nodes participating in the federated database environment. The achieved speedup was not ideal. It depends on the data set that is explored, its homogeneity, the algorithm used for 3D protein structure alignment, and applied data partitioning scheme. The best 7.62-fold speedup was achieved when performing exploration of the homogeneous data set of protein structures with the use of the FATCAT algorithm when the data set was divided with respect to the length of protein structures (AMINOACIDS partitioning scheme) in the federated database environment with 8 worker nodes. Out of all of the mentioned factors, the number of active database nodes in the federated environment decides the most about the reduction of the execution time. The more worker nodes participate in PAR-P3D-SQL job execution, the higher the degree of parallelism, and the shorter the execution time. The data partitioning scheme decides about the efficiency of the PAR-P3D-SQL query job distribution. This is especially visible for heterogeneous sets of explored protein data, where the sizes of protein structures may vary significantly. However, when we analyze proteins deposited to the Protein Data Bank...
we can see that it is highly heterogeneous. Therefore, the AMINOACIDS partitioning scheme that divides the explored part of the repository with respect to the length of protein structures (the number of amino acids) is very useful. The more heterogeneous the repository is, the more useful the AMINOACIDS data partitioning scheme is.

The overall performance of PAR-P3D-SQL queries executed in the distributed environment of Oracle federated databases is worse than the performance of similarity searching jobs run in systems, like Cloud4PSi [43], [47], CloudPSR [44], HDInsight4PSi [46], and the system proposed by Hung&Lin [45]. This disadvantage results from the limited memory the Oracle instance provides for the internal Java virtual machine. However, mentioned systems are mainly focused on one-to-all comparisons, i.e., massive similarity searches, without any sophisticated filtering that is available in the PAR-P3D-SQL. They also provide the capability of 3D protein similarity searching, but usually, their purpose is different. The PAR-P3D-SQL allows focusing on a highly selected set of protein structures that meets specified biological criteria and enables its exploration. It joins capabilities of standard database searches and searches performed with the use of dedicated algorithms for macromolecular data, taking into account the specificity and complex nature of 3D protein structures. Moreover, the outstanding performance of the HDInsight4PSi system results from using many-task computing in Hadoop-based HDInsight4PSi system [46], was much less efficient than AMINOACIDS partitioning scheme for heterogeneous data sets and almost as much efficient as AMINOACIDS partitioning scheme for homogeneous data sets.

The protein construction-based, AMINOACIDS partitioning scheme presented in the paper is a novel element in the distributed protein exploration, and it is a dedicated partitioning scheme for the specificity of processed data. It allowed us to increase the efficiency of distributed querying, which is reflected in n-fold speedup curves presented in Figs. 7 and 8. Other data partitioning schemes that were also used in systems, such as Cloud4PSi [43], [47], CloudPSR [44], Hung&Lin’s system [45] or HDInsight4PSi [46] for massive 3D protein structure alignment, appeared not to be so efficient in the PAR-P3D-SQL. The EQUALLY data partitioning scheme, the one that divides the explored part of the repository to partitions with the same numbers of protein structures, which is a counterpart of partitioning schemes used in Cloud4PSi [43], [47], CloudPSR [44], and Hadoop-based Hung&Lin’s system [45], was the least efficient. The SIZE_B data partitioning scheme, which produces byte-size-balanced data partitions and is a counterpart of partitioning scheme used in Hadoop-based HDInsight4PSi system [46], was much less efficient than AMINOACIDS partitioning scheme for heterogeneous data sets and almost as much efficient as AMINOACIDS partitioning scheme for homogeneous data sets.
Fig. 8. Execution time of PAR-P3D-SQL queries (left) and n-fold speedup over the serial P3D-SQL query (right) running FATCAT (top) or CE (bottom) algorithm for the various number of worker nodes in the distributed environment of federated databases for the homogeneous data set divided according to three data partitioning schemes (EQUALLY, SIZE_B, and AMINOACID). The dashed line shows the execution time of serial P3D-SQL query on a single database node.

Fig. 9. Structural alignment speed versus the size of the distributed system (#cluster nodes) for the federated environment of Oracle databases with PAR-P3D-SQL and other systems for 3D protein structure similarity searching.

on Azure and linking many protein structures in fixed-size sequential files. When processing individual protein structures, the performance of HDInsight4PSi is much worse [46]. As a cost of performance, the PAR-P3D-SQL provides many advantages. It is based on the declarative SQL language, which simplifies many operations, like filtering protein data, joining various sets of protein data, aggregating data, and sorting results. It allows for extensive text and numerical searches, abstracting from 3D protein structures, and for data indexing to speed up these searches. All these operations can be quickly encoded or embedded in quite simple SQL commands, making them a standardized way of accessing protein data. Storing protein macromolecular data in a relational database has also many advantages in terms of data safety and protection. Access to data is controlled by the database management system, and only authenticated users may submit queries and explore the data. Regular, automated backups or replication tasks allow keeping the copy of the data in case of database damage or disk failure. Finally, exploration of 3D protein structures is especially useful when both, input and output data for database searches, constitute intellectual property and cannot be sent to public servers due to security reasons.

PAR-P3D-SQL complements other declarative languages mentioned in Sect. I-B. The capability of the PAR-P3D-SQL to explore 3D protein structures and to search for proteins on the basis of the structural pattern provided in search criteria is unique. Other, mentioned query languages allow operating
on different, much simpler organizational levels of protein structures. ODM BLAST provided the capability of searching proteins on the basis of their amino acid sequence similarity (primary structure). PSS-SQL was exclusively focused on secondary structures represented as sequences of characters (secondary structure elements). Periscope/SQ joins both capabilities in the Periscope system. And the Protein-QL allows querying primary, secondary, and even tertiary structures of proteins in an object-oriented database, but this querying affects just basic features of protein structures and does not allow for similarity searching and structural alignment. Finally, the PAR-P3D-SQL extension for Oracle relational database presented in the paper mitigates the efficiency problems of P3D-SQL [36] and allows for scalable, distributed querying in a federated database environment.

V. CONCLUDING REMARKS

The PAR-P3D-SQL with protein construction-based data partitioning scheme presented in the paper allows querying databases of 3D protein structures and their extended exploration through similarity searching and structural alignment. The PAR-P3D-SQL is much faster than its predecessor P3D-SQL, due to the replication of protein data in the distributed environment of federated Oracle databases, the use of dedicated data partitioning scheme, and distributed query execution. It also extends capabilities of other query languages dedicated for exploration of protein data by operating on the level of tertiary structures (3D protein structures) and enabling operations that were not possible so far in relational databases, like finding protein data on the basis of 3D protein structure similarity.

Future works will be focused on improving the performance of the PAR-P3D-SQL and providing the possibility to work with protein macromolecular data stored in different formats. We believe that our extension for Oracle relational databases will be a right step toward performing sophisticated exploration tasks as a standard procedure available in declarative query languages, as the SQL.

ACKNOWLEDGMENT

This work was supported by Microsoft Research within Microsoft Azure for Research Award grant, pro-quality grant for highly scored publications or issued patents of the Rector of the Silesian University of Technology, Gliwice, Poland (grant No 02/020/RGJ19/0167), and partially, by Statutory Research funds of Institute of Informatics, Silesian University of Technology, Gliwice, Poland (grant No BK/204/RAU/2/2019).

The PAR-P3D-SQL extension for Oracle PL/SQL is available at the project homepage http://zti.polsl.pl/w3/dmrozek/science/p3dsql.htm. Further development of the system will be carried out by the Cloud4Proteins non-profit, scientific group (http://www.zti.aei.polsl.pl/w3/dmrozek/science/cloud4proteins.htm).

REFERENCES


This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.


