論文内容の要旨

論文題目

Accelerating Path-free XML Queries in RDBMS

(XML 中のパスフリー検索を RDBMS 上で高速に行う手法)

氏名　　呉紅艶

Extensible Markup Language (XML) is a simple, flexible text format. XML gains international acceptance for its openness. It can conveniently describe semi-structured data, sparse data, hierarchical data, and metadata, some of which cannot be well managed in RDBMS.

For example, when retrieving the mouse homologs of medaka, we care for the attributes of gene ID, transcript ID, chromosome name and so on. We will get the flat table-formatted result as shown in Fig.1. For the same medaka ensemble gene ENSORLG00000000006, we will have to list it three times, because the gene ID has three different transcript IDs, although the other attributes remain completely unchanged. In flat table format, every attribute can only have one value. For multi-attribute records such as gene ENSORLG00000000006, it has to be separated into different rows. In addition, for the chromosome attributes here only have two different values (only 24 different values even for all data) we are forbidden to aggregate them into two groups. It is the same to strand attribute. Another disadvantage of flat format is that there does not exist any index or query language for it. You have to design the query by yourself.

EnsemblGene ID	EnsemblTranscript ID	Chrname	GeneStart	GeneEnd	Strand	MouseEnsemblGeneID
ENSORLG00000000003	ENSORLT00000000004	1	164201	167143	-1	ENSMUSG00000027327
ENSORLG00000000004	ENSORLT00000000005	1	168507	169421	-1	
ENSORLG00000000006	ENSORLT00000000008	3	368154	395393	-1	ENSMUSG00000033629
ENSORLG00000000006	ENSORLT00000000010	3	368154	395393	-1	ENSMUSG00000033629
ENSORLG00000000006	ENSORLT00000000011	3	368154	395393	-1	ENSMUSG00000033629
ENSORLG00000000007	ENSORLT00000000006	1	179992	184640	-1	ENSMUSG00000028458
ENSORLG00000000009	ENSORLT00000000013	1	219084	231082	-1	ENSMUSG00000027075
ENSORLG00000000009	ENSORLT00000000014	1	219084	231082	-1	ENSMUSG00000027075
ENSORLG00000000010	ENSORLT00000000012	3	396055	405495	1	ENSMUSG00000000154

Fig. 1 **Fragment of flat table data, mouse homologs of medaka's**

XML provides us a more flexible way to manage these problems, as demonstrated in Fig. 2. The Gene ENSORLG00000000006 occurs only once, because every cell can have multiple transcript ID attributes. We can also group the data by strand by adding a *strand* level, and then we can differentiate all genes by their strand direction.

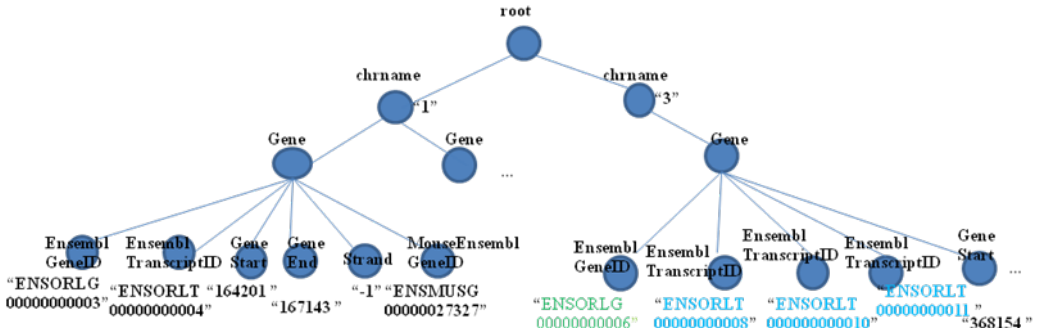

Fig. 2 **XML model for mouse homologs of medaka's**

The seminal event, the emergence of WWW in the 1990, increases the need of data interchange and also the need to deal with more heterogeneous data, which greatly extends the application of XML, since XML data model facilitates heterogeneous and semi-structured data management.

The morphological database, SCMD, was built to identify the morphological changes in individual mutants. Cells can be grouped into their stage in the cell cycle, which reflects the duration of the phase statistically. According to your research interest, besides the size of the bud, the phase of cells in the cell cycle can also be categorized based on the detailed information on nuclear DNA and actin localization. XML provides a flexible management for any preference. The following figures illustrate a bud size tree structure (Fig.3) and an actin localization tree structure (Fig.4), respectively.

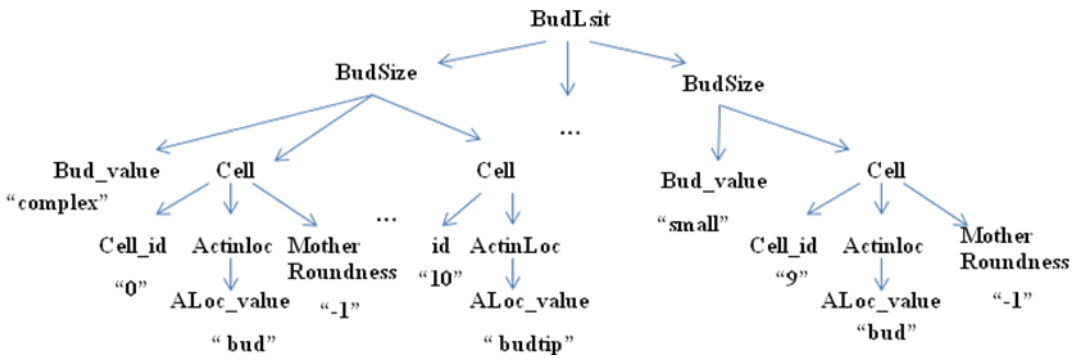

Fig. 3 **XML model for morphological data: group by budsize**

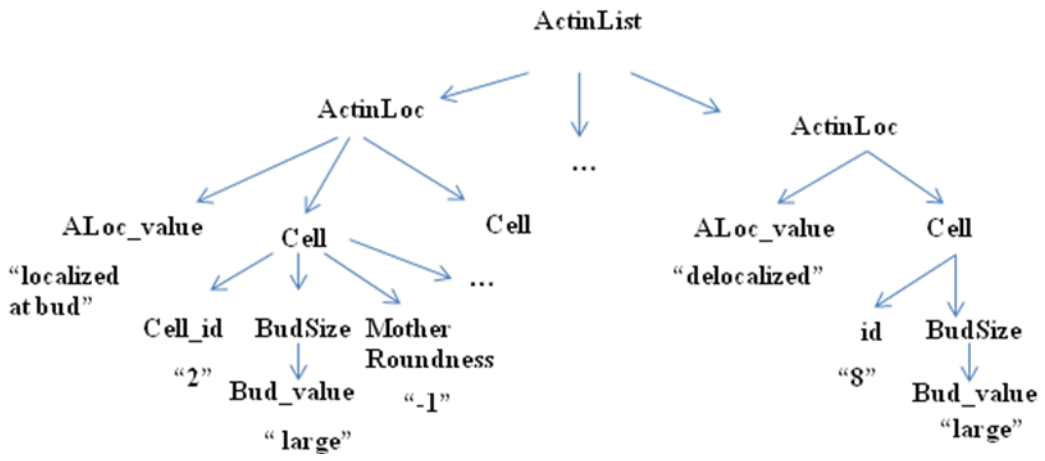

Fig. 4 **An alternative XML model for morphological data: group by actin localization**

However, traditional XML query processing methods, such as XPath and XQuery, must strictly follow the path-expressions in XML, which is not only error-prone, but fragile to changes in the underlying XML structure because path expressions cannot accommodate structural variations that may occur in designing or updating XML data. In the above example, to retrieve the cell's bud value and mother roundness if its actin localization is at bud, two completely different queries have to be written for these two tree structures:

```
Q1:   //BudSize[Cell[ActinLoc[@ALoc_value="bud"]]]/Bud_value |
      //Cell[ActinLoc[@ALoc_value="iso"]]/MotherRoundness
Q2:   //ActinLoc[@ALoc_value="bud"]/Cell//Bud_value |
      //ActinLoc[@ALoc_value="iso"]/Cell/MotherRoundness
```

In this cases a query without the path expressions that looks like *(cell, BudSize, ALoc_value, where @ ALoc_value ="bud")* will be preferable.

Considering the problems of path-based queries mentioned above, schema-free, XRank, relational-style XML query and keyword search have been proposed. All of these previous approaches design their own native XML databases to solve the problem, which not only needs duplicate efforts, lose the portability, but makes the result unreusable for the later queries because of the inherent structural variations of XML. Therefore, we explore the approach of devising path-free XML queries in standard relational database systems with SQL by utilizing their sophisticated data querying and materializing ability.

In this paper, we firstly propose the idea that process path-free XML queries in a pure RDBMS. It is ideal to list all possible structural variations of a given path-free XML query, though it is non-trivial to devise an efficient implementation due to the combinatorial explosion of potential structural variations, n^{n-1} tree patterns for n queried items. In addition the problems that RDBMS cannot offer an ideal query plan and efficient XML structural join algorithms also pose a big challenge.

By adding XML-specific information to rewrite SQL clauses, we make RDBMS aware of tree structures, thereby accelerating structural joins. In addition we analyze the optimization plan of the original RDBMS and propose a more efficient structural combination execution plan. In particular, we incorporate functional dependencies into our SQL queries to eliminate the unfavorable results and reduce the query space. We design an XML interface module for Saccharomyces Cerevisiae Morphological Database (SCMD), a database of budding yeast mutants, so as to improve the feasibility of the original database.

Experiments carried out on SQL Server have proven orders of magnitude improvement over the naïve implementation, demonstrating the feasibility of path-free XML query processing using a pure RDBMS kernel. Retaining the portability, our RDBMS implementation is as fast as the previous native XML implementation.

At last, we introduce potential applications of our proposed approach to data integration. Utilizing the unique characteristics of our method serves as a unified platform for XML data and relational data. Indeed, we can take advantage of the flexibility of XML, and also the powerful ability of relational database. Dealing with XML query in a pure RDBMS efficiently bridges the semantic gap between XML and relational database.