Presentation Abstract
by :
Jean-Sébastien Légaré

McXML is a native XML database management system. That is, a database specifically made to store semi-structured data documents. The use of the Extensible Markup Language (XML) is continuously growing in fields such as data transmission or web publishing due to its extreme flexibility and simplicity. So far the only native XML database systems that exist focus on data queries uniquely. Unlike others, McXML supports data updates as well as data queries.

The project was put in place 4 years ago by students of Prof. Bettina Kemme. The very first version of McXML performed the updates and queries directly on an ASCII .xml file. Later, a storage manager whose task was to translate the document to a binary representation on disk was implemented to allow the system to minimize the amount of data that had to be rewritten during an update.

Still, the project lacked one of the most critical components present in every database management system: memory management. When a document was requested or an update had to be done, the complete document had to be retrieved, even though a very small portion of the data would need to reside in main memory. In a typical relational database, the Buffer Manager fills this need by limiting the number of pages that can be loaded into memory simultaneously.

Although McXML is not a relational database, the theory behind any buffer manager could be applied. I have been given the chance this summer to integrate a buffer manager into McXML. This is a complex integration that had to be broken down in several steps:

1. The logical data model used by the update engine had to be translated to an intermediary model before being translated to its binary form on disk. The changes from one model to the other had to be reflected directly.

2. Algorithms that reflected the changes on the logical model to the physical binary representation had to be reimplemented.

3. The sections of documents had to be loaded transparently one at a time only when requested and not all at once.

4. A component that would hold the links between the loaded sections had to be integrated so as to be able to "disconnect" them when they were no longer needed.

In my presentation, I will show what steps of the integration have been accomplished and what were the difficulties that I ran into.