CS298 Report

Schemes to make Aries and XML work in harmony

Thien An Nguyen

Advisor: Dr. Chris Pollett
Committee Members: Dr. Melody Moh
Dr. Tsau Young Lin
Agenda

• Our project goals
• ARIES Overview
• Natix Overview
• Our Project Design and Implementations
• Performance Matrixes
• Conclusion
Project Goals

• Develop an XML toy database.
• Implement ARIES and Natix’s recovery mechanism.
• Support basic operations: insert, load, update, delete, commit, and rollback.
• Allow some basic commands such as printTree, and archive.
• Let user to vary the data page’s size and the buffer pool’s size (for page swapping experiments).
• Set up a test environment for user to experiments
Introduction

• In general, there are three phases:
  – Analysis:
    • read the log records from the last system checkpoint
    • analyze them
    • build a list of actions for the redo and undo phases
  – Redo
    • Use log records to bring the dirty data pages updated by all transactions to a point in time (consistency point – no update operations allow)
  – Undo
    • Use log records to undo updates made by un-committed transactions
ARIES Overview

- Buffer management uses steal/no-force policy:
  - Steal: flush dirty page(s) by a transaction onto disk before the transaction commit.
  - No-force: page(s) modified by a transaction does not have to be flushed onto disk when the transaction commits.

- All update operations must be logged using log records.

- Use write-ahead logging protocol:
  - A log record must be created on storage before the update operation can be completed.

- Undo operations are also recorded using compensation log records (CLR):
  - Log record written for rollback operation to say what has been rolled back. (no need to recover what has been rolled back.)
**Figure 1 - CLR**

![Diagram of CLR](image)

$I'$ is the Compensation Log Record for $I$
$I'$ points to the predecessor, if any, of $I$

**Figure 2 – Subsidiary Logging**

![Diagram of Subsidiary Logging](image)
Natix Overview

• Same as ARIES but modify the way log records written during forward phase.

• Subsidiary Logging:
  – Page interpreter keeps a private log for updates to the same transaction, onto the same page.
  – The log content is locally modified to reduce the number of log records.
  – Subsidiary log’s content is published to log manager right before a page is flushed to disk (according to WAL) or the transaction commit.

• Undo operations also be recorded using compensation log records.

• Selective Redo:
  – Use log records to bring dirty pages (updated by committed transactions) to a point in time from the last image copy.
Annihilator Undo

- Updates to a record that created by the same transaction need not to be undone when the transaction is aborted as the record will be deleted anyway.
- Instead of undo 5, 4, 3, 2, 1, undo 4, 1 would be enough.
Recovery Management

Log

Checkpoint / Failure

ARIES

Analysis
Redo ALL
Undo Losers

NATIX

Analysis
Redo Nonlosers
Undo Losers
Design of the toy Database (p. 16)
XML Documents Example

<XML ID="00000001">
  <G name="CS297">
    <G id="123-45-6789">
      <L>Jane Eyre</L>
    </G>
    <G id="234-56-7890">
      <L>Irene Hugh</L>
      <L>3.0</L>
    </G>
  </G>
</XML>
Logical Tree Structure (p. 19)

Figure 9 – Logical Tree Associate with Its XML Document
Query Engine (p. 17)

- Includes:
  - query compiler, query parser, query preprocessor, and query optimizer.

- We support:
  - query parser
    - Parses tags: <XML>, </XML>, <G>, </G>, <L>, </L>. (=> language is simple but can have an arbitrary tree structure)
    - Parses grammar: <G>, and <L> tags can be nested inside <G> tags. <L> tags can’t be nested.
    - Parses queries: insert, update, delete, rollback,…
    - Parses commands: printTree, archive,…
Some DDL/DML statements

• Data Definition Language
  – Statements that are used to define objects.

  Example of a DDL statement
  
  \( T1 \) CREATE DATABASE DB1DB;
  
  \( T1 \) CREATE XMLDOC XML1DOC IN DB1DB;

• Data Manipulation Language
  – Statements that are used to modify, manipulate objects.

  Example of a DML statement
  
  \( T1 \) INSERT INTO XML1DOC PATH("/G/G[1]/L") VALUES("<L>Jane Eyre</L>",
  
  "<L>3.8</L>");

  \( T1 \) COMMIT;
  
  \( T2 \) UPDATE XML1DOC SET VALUE("<L>392 Lulu Ahh Dr., San Jose CA 95123</L>") WHERE
Execution Engine

- Has input which are either queries, and/or commands.
- Drives the execution for each query/command.
- Returns the result back to user application (printTree).
Buffer Manager (p. 18)

- Allocates, manages, and de-allocates pages in the buffer pool.
The ThPage and ThPageInterpreter Relationship (p. 18)

ThPage implements IThPage

ThPage Interpreter extend ThPage implement IThPage
ThPage and ThPageInterpreter space management (p. 23)
ThPage and ThPageInterpreter space management

• Data Structure:
  – 2 bytes page number
  – Half byte dirty bit
  – 2 bytes number of nodes in a page (maximum nodes in a page is 255)
  – ArrayList of nodes

• Functions support: decode/encode for read/write operations.

• The only different between ThPage and ThPageInterpreter is ThPageInterpreter does subsidiary logging.
Tree Storage Manager

- Construct and manipulate tree structure object at the logical and physical level
- Implemented in class ThXMLDoc
  - Keeps track of a set of pages that contain the tree structure document (called pageSet)
  - Has pointer to the root node
Page Split (p. 22)

Before Split

After Split
Transaction Management

- Contains a Hashtable of transaction
  - A transaction
    - Transaction ID
    - Its current state (committed or not)
    - List of LSNs

Log Management

Reads and writes log records
The `dbsystem.properties` file

- Used to specify system initial values
- For example:

```properties
pageSize = 500  
recoverMethod = ARIES  
# The following section is random test generator parameters  
numberOfNodes = 500  
randomOutFile = c://an//cs297/xmlmdb/src//sm//input//randomTest.txt  
insertWeight = 7  
updateWeight = 3  
deleteCount = 4  
# paragraph = alsk  
djflsaksjffjffMXc.,.,xzMCz.,mczx.,lklksdfgdfs?,.,mczxmcC`12222345z.mcxzMCalksd832~!@#$%^&*_+-=[]{};"|;,:;djflsaksjffjffMXc.,.,xzMCz.,mczx.,lklksdfgdfs?,.,mczxmcC`12222345z.mcxzMCalksd832~!@#$%^&*_+-=[]{};"|;,:;  
paragraph = test
```
Performance Experiments (p. 58)

Large Structure, Small Data vs. Page Size

Time (s)

Page Size (Bytes)

Natix

ARIES
Challenges

• Some of the challenges:
  – Natix and ARIES’s design and implementation
  – Space storage management
  – Buffer manager design and implementation
  – Tree manager design and implementation
Future development

• Some of the area the can be added, improve in the future:
  – Lock management
  – Can be extended to model client-server environment
  – Fast Log applies for recovery management.
  – Defer recovery.
Conclusion (p. 59)

- We developed a toy XML database management system (DBMS) useful for studying recovery and buffer management issues.
- Our system is capable of storing XML document that span over pages
- Our DBMS supports operations and commands such as create, insert, update, delete, rollback, commit, load, and printTree
- It supports both ARIES and Natix recovery methods.
- We also established an experimental environment and experimented with page sizes, and recovery methods.