FATWORM IMPLEMENTATION

Chenyang Wu
FATWORM IMPLEMENTATION

Chenyang Wu
OVERVIEW

• Keywords
• A Traditional Architecture
• A Traditional Implementation
• Resources
KEYWORDS
KEYWORDS

- RDBMS, scratch
- Simplified SQL - FwSQL
- Java or JVM-based
- Teamwork, SCM
I'm too difficult to implement and test
TRADITIONAL ARCHITECTURE

File System

Storage Management

Buffer Management

Transaction

Recovery

Logging

Concurrency

Data Management

Metadata

Record

Indexing

Query Processing

Parser

Planner

Executor

Query

User Interaction

JDBC

Console

Transaction Management

Java RMI

I'm boring and not cool

I'm too difficult to implement and test

Query
TRADITIONAL ARCHITECTURE

User Interaction
- JDBC

Query Processing
- Parser
- Planner
- Executor

Data Management
- Metadata
- Record
- Indexing

Storage Management
- Buffer Management

File System
ARCHITECTURE

• Feel free to design, but don’t go too far
ARCHITECTURE

- Feel free to design, but don’t go too far
- Be lightweight and functional
ARCHITECTURE

- Feel free to design, but don’t go too far
- Be lightweight and functional
- The state-of-art designs, columns-based...
ARCHITECTURE

- Feel free to design, but don’t go too far
- Be lightweight and functional
- The state-of-art designs, columns-based ...
- No NOSQL
ARCHITECTURE

• Feel free to design, but don’t go too far
• Be lightweight and functional
• The state-of-art designs, columns-based ...
• No NOSQL
• Design Review
TRADITIONAL IMPLEMENTATION

- Storage
- Parser
- Query Processing
FILE
FILE

- Store data in units (page, block)
FILE

- Store data in units (page, block)
- Table per file / Database per file / One big file
FILE

- Store data in units (page, block)
- Table per file / Database per file / One big file
- Without concurrency, you need to improve performance
FILE

- Store data in units (page, block)
- Table per file / Database per file / One big file
- Without concurrency, you need to improve performance
- InputStream, ByteBuffer, DirectBuffer
BUFFER

- Facade for Storage Module
BUFFER

- Facade for Storage Module
- Cache Revisit
BUFFER

- Facade for Storage Module
- Cache Revisit
- Remember the replacement strategy?
BUFFER

• Facade for Storage Module
• Cache Revisit
• Remember the replacement strategy?
• Memory size will be fixed when testing, so keep everything configurable
RECORD(HEAPFILE)
RECORD(HEAPFILE)

- Table = Array[Column] + Array[Row]
RECORD(HEAPFILE)

- Table = Array[Column] + Array[Row]
- Is record fixed-size?
RECORD(HEAPFILE)

• Table = Array[Column] + Array[Row]

• Is record fixed-size?
  • VARCHAR
RECORD(HEAPFILE)

- Table = Array[Column] + Array[Row]
- Is record fixed-size?
  - VARCHAR
- What happens on record deletion?
RECORD(HEAPFILE)

- Table = Array[Column] + Array[Row]
- Is record fixed-size?
  - VARCHAR
- What happens on record deletion?
  - Space Reuse
RECORD(HEAPFILE)

- Table = Array[Column] + Array[Row]
- Is record fixed-size?
  - VARCHAR
- What happens on record deletion?
  - Space Reuse
- Are the records ordered?
RECORD(HEAPFILE)

- Table = Array[Column] + Array[Row]
- Is record fixed-size?
  - VARCHAR
- What happens on record deletion?
  - Space Reuse
- Are the records ordered?
  - RID
RECORD(HEAPFILE)

- Table = Array[Column] + Array[Row]
- Is record fixed-size?
  - VARCHAR
- What happens on record deletion?
  - Space Reuse
- Are the records ordered?
  - RID
- How to find a row efficiently?
RECORD (HEAPFILE)

- Table = Array[Column] + Array[Row]
- Is record fixed-size?
  - VARCHAR
- What happens on record deletion?
  - Space Reuse
- Are the records ordered?
  - RID
- How to find a row efficiently?
  - Index
MANAGE THE METADATA
MANAGE THE METADATA

- class Column
  - pos: Int
  - type: java.sql.TYPES
  - len: Int
  - hasIndex: Int
MANAGE THE METADATA

- **class Column**
  - `pos`: Int
  - `type`: java.sql.TYPES
  - `len`: Int
  - `hasIndex`: Int

- **class Table**
  - `columns`: Array[Column]
  - `rowCount`: Int
  - `pages`: Array[PageInfo]
MANAGE THE METADATA

- class Column
  - pos: Int
  - type: java.sql.TYPES
  - len: Int
  - hasIndex: Int

- class Table
  - columns: Array[Column]
  - rowCount: Int
  - pages: Array[PageInfo]

- Can Metadata be a table?
MANAGE THE METADATA

- class Column
  - pos: Int
  - type: java.sql.TYPES
  - len: Int
  - hasIndex: Int

- class Table
  - columns: Array[Column]
  - rowCount: Int
  - pages: Array[PageInfo]

* Can Metadata be a table?

* Or an 'i-node' like implementation
INDEXING
INDEXING

• B+ Tree, Hash
INDEXING

- B+ Tree, Hash
- R-Tree? KD-Tree?
INDEXING

• B+ Tree, Hash
• R-Tree? KD-Tree?
• An implicit index on RID, good or bad?
INDEXING

• B+ Tree, Hash
• R-Tree? KD-Tree?
• An implicit index on RID, good or bad?
• VARCHAR Revisited
PARSER

- JFlex + CUP
PARSER

- JFlex + CUP
- Anything else available, like ANTLR, ...
PARSER

- JFlex + CUP
- Anything else available, like ANTLR, ...
- The only place you can use 3rd-party library
QUERY PROCESSING
EXAMPLE 1
QUERY PROCESSING

EXAMPLE 1

• SELECT * FROM A,B WHERE A.a = B.b
QUERY PROCESSING

EXAMPLE 1

- SELECT * FROM A,B WHERE A.a = B.b

- SelectPlan(
  'a=b',
  ProductPlan(
    TablePlan(A),
    TablePlan(B)))
QUERY PLANNING
EXAMPLE 2
QUERY PLANNING
EXAMPLE 2

SELECT * FROM A,B WHERE A.a = 1
QUERY PLANNING
EXAMPLE 2

• SELECT * FROM A,B WHERE A.a = 1

• SelectPlan(
  'a=1',
  ProductPlan(
    TablePlan(A),
    TablePlan(B))))
QUERY PLANNING
EXAMPLE 2

- SELECT * FROM A, B WHERE A.a = 1

- SelectPlan(
  'a=1',
  ProductPlan(
    TablePlan(A),
    TablePlan(B))))

- SELECT * FROM (SELECT * FROM A WHERE A.a=1), B
QUERY PLANNING

EXAMPLE 2

- SELECT * FROM A,B WHERE A.a = 1

- SelectPlan('a=1',
  ProductPlan(
    TablePlan(A),
    TablePlan(B))))

- SELECT * FROM (SELECT * FROM A WHERE A.a=1), B

- ProductPlan(SelectPlan('a'=1, TablePlan(a)),
  TablePlan(b))
QUERY PROCESSING
EXAMPLE 2(CONT)
• What if A has an index on column a?
What if A has an index on column a?

ProductPlan(
  IndexSelectPlan(
    'a'=1, TablePlan(a)),
  TablePlan(b))
• What if A has an index on column a?

• ProductPlan(
  IndexSelectPlan(
    'a' = 1, TablePlan(a)),
  TablePlan(b))

• What if A has an index and the clause is A.a=B.b?
QUERY PROCESSING
ITERATOR

• A very useful pattern
QUERY PROCESSING

ITERATOR

• A very useful pattern

• Iterator<> in Java
QUERY PROCESSING

ITERATOR

• A very useful pattern
• Iterator<> in Java
• Pipeline
QUERY PROCESSING
ITERATOR

• A very useful pattern

• Iterator<> in Java

• Pipeline

• interface Iterator
  - (void) beforeFirst();
  - (bool) next();
  - (Tuple) getCurrent();
QUERY PROCESSING
ITERATOR
public class SelectScan implements Iterator {
    private Scan innerScan;
    public boolean next() {
        while (innerScan.next()) {
            if (condition()) return true;
        }
        return false;
    }
}
QUERY PROCESSING
ITERATOR

- MergeJoinScan(Multiway merge sort)
- SortScan
- GroupByScan
- ...

• SELECT name FROM icecream as ice
  WHERE ice.name = 'S' AND icecream.name = 'B'
• SELECT name FROM icecream as ice
  WHERE ice.name = 'S' AND icecream.name = 'B'

• SELECT name FROM icecream
  WHERE EXIST
    (SELECT name FROM favorite_icecream as fi
     WHERE fi.name = icecream.name)
QUERY PROCESSING
OPTIMIZATION
QUERY PROCESSING OPTIMIZATION

• Select-Projection-Join
QUERY PROCESSING
OPTIMIZATION

• Select-Projection-Join
• Join order with statistics
QUERY PROCESSING OPTIMIZATION

- Select-Projection-Join
- Join order with statistics
- Optimization is endless
SUGGESTIONS

• Well-designed system
SUGGESTIONS

• Well-designed system

• Run a query in your design
SUGGESTIONS

• Well-designed system
• Run a query in your design
• Combine things ASAP
SUGGESTIONS

• Well-designed system

• Run a query in your design

• Combine things ASAP

• A lot of testing
RESOURCES

• Textbooks
  Database systems: The complete book
  Database design and implementation

• Open source DBMS
 MySQL, SQLite, ...

• SimpleDB
  A naive implementation in Java

• Specification and Wiki
  http://fatworm.acm-project.org
Questions?