Lecture 3
Fatworm Scan

Hongwei Wang
2014.4.25
Background

• JDBC

Connection conn = DriverManager.getConnection(uri, usr, pw);
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("select id, name from People");
while (rs.next()) {
    int id = rs.getInt(1);
    String name = rs.getString(2);
    // TODO
}

Background

• SQLite (in Android)

ContentResolver resolver = getContext().getContentResolver();

Cursor cursor = resolver.query(Phone.CONTENT_URI, new String[] {Phone.DISPLAY_NAME, Phone.NUMBER}, null, null, null);

if (cursor.moveToFirst()) {
    do {
        String name = cursor.getString(0);
        String phoneNumber = cursor.getString(1);
        // TODO
    }
    while (cursor.moveToNext());
}
Principle of Scan: Pipelining

- Combining several relational operations into a pipeline of operations to eliminate the cost of reading and writing temporary relations
- Example: select a, b from r, s ( Π_{a,b}(r \bowtie s))
- Pipelines can be executed in either of two ways: demand driven and producer driven
- Each operation in a demand-driven pipeline can be implemented as an iterator that provides the following functions: open(), next(), close()
- The iterator maintains the state of its execution in between calls, so that successive next() requests receive successive result tuples
Principle of Scan: Pipelining

- Design of Scan: demand-driven pipelining query evaluation
Example: Plan and Scan

• select distinct A.a, B.b from A, B where A.a > 1 order by B.b
Scan Interface

• Design scan as interface

What is interface in Java?

an abstract type that only contains abstract methods...
specify a serial of methods that must be implemented...

```java
public interface Scan {
    public void beforeFirst();
    public boolean next();
    public Object getObjectByIndex(int index);
    public int getTypeByIndex(int index);
    public int getColumnCount();
    ......
}
```
Scan Interface

```java
public class GroupScan implements Scan {
    Scan scan;

    public GroupScan (…) {...}

    @Override
    public void beforeFirst() {...}
    ......

    /* other private fields and methods */
}
```
Example: Mechanism of Scan

• select a from A where a > 0

• How do we test your code?

stmt.execute(the query above);
rs = stmt.getResultSet();
rs.beforeFirst();
while (rs.next()) {
    for (i=0; i<metadata.getColumnCount(); i++) {
        rs.getObject(i);
    }
}
Example: Mechanism of Scan

• select a from A where a > 0

• rs.beforeFirst()
  -> projectionScan.beforeFirst()
  -> selectionScan.beforeFirst()
  -> tableScan.beforeFirst()
  -> storage initialization

projectionScan
selectionScan
tableScan

<table>
<thead>
<tr>
<th>Table A</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
</tr>
<tr>
<td>-1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

stmt.execute(the query above);
rs = stmt.getResultSet();
rs.beforeFirst();
while (rs.next()) {
  for (i=0; i<metadata.getColumnCount(); i++) {
    rs.getObject(i);
  }
}
Example: Mechanism of Scan

• select a from A where a > 0

• rs.next()
- projectionScan.next()
- selectionScan.next() \( a > 0 \) ?
- tableScan.next()
- blocks and pages

```java
stmt.executeUpdate(the query above);
rs = stmt.getResultSet();
rs.beforeFirst();
while (rs.next()) {
    for (i=0; i<metadata.getColumnCount(); i++) {
        rs.getObject(i);
    }
}
```
Example: Mechanism of Scan

- select a from A where a > 0

- Metadata.getColomnCount()
  -> projectionScan.getColumnCount()
  -> width of the projection

  
  // selectionScan.getColumnCount()
  // tableScan.getColumnCount()

```java
stmt.executeUpdate("select a from A where a > 0");
rs = stmt.getResultSet();
rs.beforeFirst();
while (rs.next()) {
    for (i = 0; i < metadata.getColumnCount(); i++) {
        rs.getObject(i);
    }
}
```
Example: Mechanism of Scan

• select a from A where a > 0

• rs.getObject(i)
  -> projectionScan.getObject(i)
  -> selectionScan.getObject(i’)
  -> tableScan.getObject(i’)
  -> blocks and pages

stmt.execute(the query above);
rs = stmt.getResultset();
rs.beforeFirst();
while (rs.next()) {
  for (i=0; i<metadata.getColumnCount(); i++) {
    rs.getObject(i);
  }
}
Implementation

- Distinct Scan
- Order Scan
- Projection Scan
- Selection Scan
- Group Scan
- Join Scan
- Rename Scan
- Table Scan
Implementation: Order Scan

```java
void beforeFirst() {
    retrieve all tuples in advance;
    sort;
    save;
}

boolean next() {
    traverse in your pre-built table;
}
```
Implementation: Projection Scan

getXxxByIndex(int index) {
    index -> index’;
    scan.getXxxByIndex(index’);
}"
Implementation: Join Scan

boolean next() {
    enumerate every possible combination of its scan list;
}

Distinct Scan
Order Scan
Projection Scan
Selection Scan
Group Scan
Join Scan
Rename Scan
Table Scan
Implementation: Table Scan

• bridge between scans and storage
• For simplicity, you can implement Fatworm in memory first
• Memory version: tuples stored in arrays
• Hard disk version: tuples stored in files, managed by page and block
Challenges

• Group and aggregate functions?
• Rename (table and column)?
• Order by invisible column?
• Other disgusting test cases: select constant, aggregate functions without groupby…
• Storage & Optimization… (what you should concern in following phases)
Tips

• Be familiar with Java;
• Make full use of Eclipse;
• Consult others (TAs, your classmates) if you are confused;
• Current task: make your Fatworm runnable!
• Do not delay!
Thanks