Assignment 2 Explanation

Introduction to Database Systems

DataLab

CS, NTHU

Outline

- Client-side classes
  - *As2BenchmarkRte*
  - *As2UpdateItemPriceParamGen*
  - *UpdateItemPriceTxnJdbcJob*
- Server-side classes
  - *UpdateItemPriceProcParamHelper*
  - *UpdateItemPriceTxnProc*
- StatisticMgr

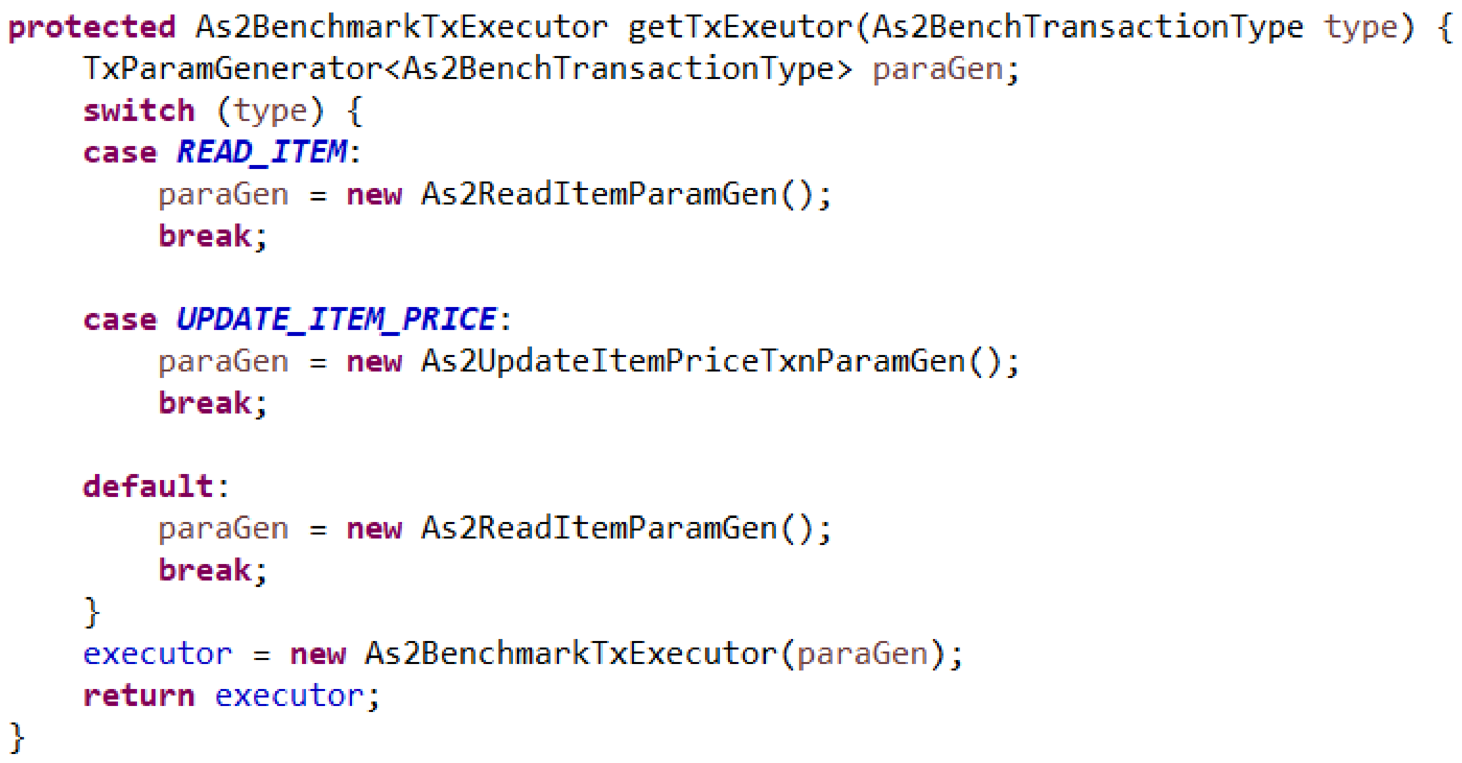
Outline

- Client-side classes
  - *As2BenchmarkRte*
    1. *Executor*
    2. *READ_WRITE_TX_RATE*
  - *As2UpdateItemPriceParamGen*
  - *UpdateItemPriceTxnJdbcJob*
- Server-side classes
  - *UpdateItemPriceProcParamHelper*
  - *UpdateItemPriceTxnProc*
- StatisticMgr

Executor

Q: Which place is better to new an executor?

Solution :

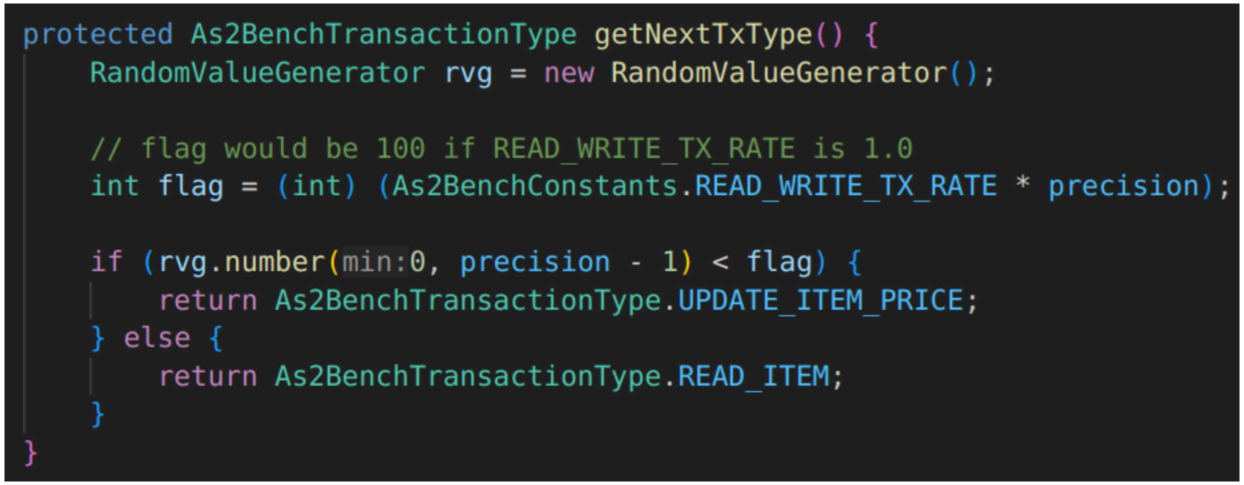


```
protected As2BenchmarkTxExecutor getTxExeutor(As2BenchTransactionType type) {
    TxParamGenerator<As2BenchTransactionType> paraGen;
    switch (type) {
    case READ_ITEM:
        paraGen = new As2ReadItemParamGen();
        break;

    case UPDATE_ITEM_PRICE:
        paraGen = new As2UpdateItemPriceTxnParamGen();
        break;

    default:
        paraGen = new As2ReadItemParamGen();
        break;
    }
    executor = new As2BenchmarkTxExecutor(paraGen);
    return executor;
}
```

READ_WRITE_TX_RATE

Q: Use READ_WRITE_TX_RATE to control the UPDATE rate

Solution:



```
protected As2BenchTransactionType getNextTxType() {
    RandomValueGenerator rvg = new RandomValueGenerator();

    // flag would be 100 if READ_WRITE_TX_RATE is 1.0
    int flag = (int) (As2BenchConstants.READ_WRITE_TX_RATE * precision);

    if (rvg.number(min:0, precision - 1) < flag) {
        return As2BenchTransactionType.UPDATE_ITEM_PRICE;
    } else {
        return As2BenchTransactionType.READ_ITEM;
    }
}
```

FAQ:

Question 1: `READ_WRITE_TX_RATE` 的定義

> L 君: 請問調 ReadItemTxn 跟 UpdateItemPriceTxn 的 Ratio 是指說 1) 多個 RTE 跑不同 Txn , 2) 同一個 RTE 跑多個 Txn?

這個要求的意思是說，在一個 RTE 選擇要跑哪一種 txn (transaction 的縮寫) 時，有多少機率會選到

Outline

- Client-side classes
  - *As2BenchmarkRte*
  - *As2UpdateItemPriceParamGen*
  - *UpdateItemPriceTxnJdbcJob*
- Server-side classes
  - *UpdateItemPriceProcParamHelper*
  - *UpdateItemPriceTxnProc*
- StatisticMgr

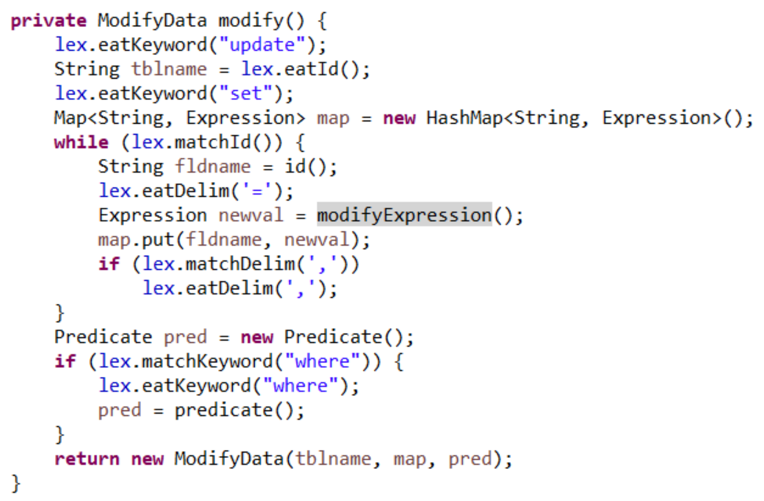
Outline

- Client-side classes
  - *As2BenchmarkRte*
  - *As2UpdateItemPriceParamGen*
  - *UpdateItemPriceTxnJdbcJob*
    1. *Update SQL*
    2. *Update return*
- Server-side classes
  - *UpdateItemPriceProcParamHelper*
  - *UpdateItemPriceTxnProc*
- StatisticMgr

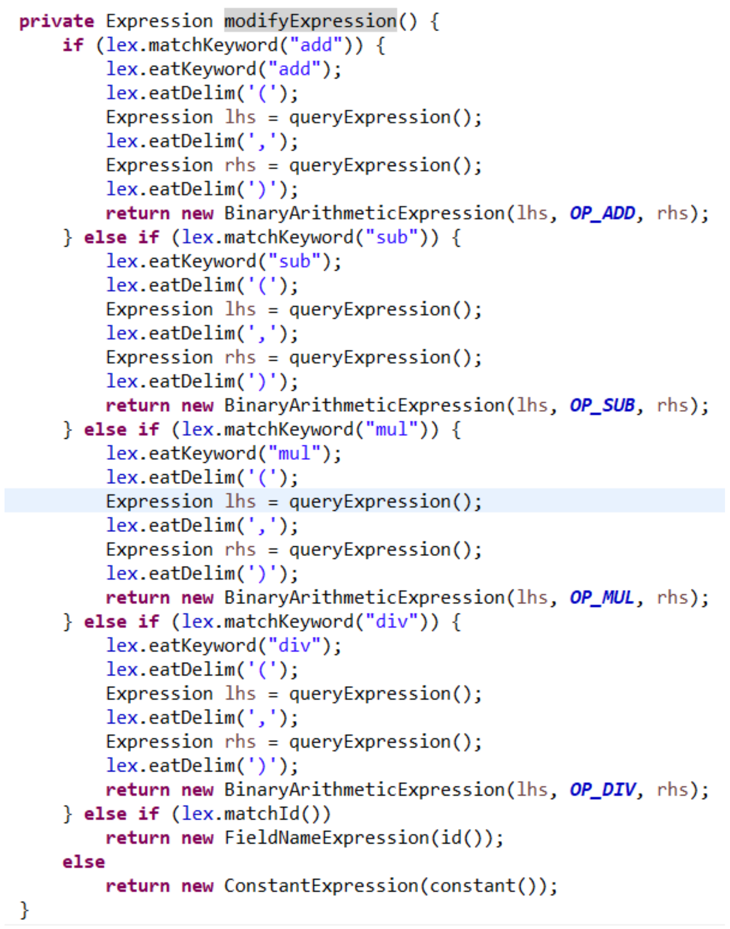
Update SQL

Q: sql = "UPDATE item SET i_price = i_price ~~+ " + pru + "WHERE i_id = " + iid;~~

Parser:



```
private ModifyData modify() {
    lex.eatKeyword("update");
    String tblname = lex.eatId();
    lex.eatKeyword("set");
    Map<String, Expression> map = new HashMap<String, Expression>();
    while (lex.matchId()) {
        String fldname = id();
        lex.eatDelim('=');
        Expression newval = modifyExpression();
        map.put(fldname, newval);
        if (lex.matchDelim(','))
            lex.eatDelim(',');
    }
    Predicate pred = new Predicate();
    if (lex.matchKeyword("where")) {
        lex.eatKeyword("where");
        pred = predicate();
    }
    return new ModifyData(tblname, map, pred);
}
```

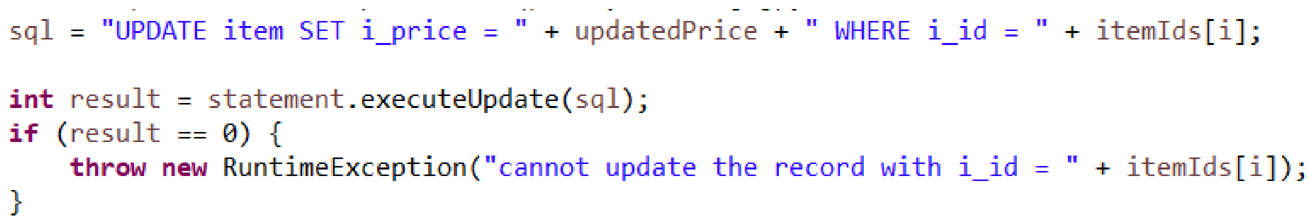


```
private Expression modifyExpression() {
    if (lex.matchKeyword("add")) {
        lex.eatKeyword("add");
        lex.eatDelim('(');
        Expression lhs = queryExpression();
        lex.eatDelim(',');
        Expression rhs = queryExpression();
        lex.eatDelim(')');
        return new BinaryArithmeticExpression(lhs, OP_ADD, rhs);
    } else if (lex.matchKeyword("sub")) {
        lex.eatKeyword("sub");
        lex.eatDelim('(');
        Expression lhs = queryExpression();
        lex.eatDelim(',');
        Expression rhs = queryExpression();
        lex.eatDelim(')');
        return new BinaryArithmeticExpression(lhs, OP_SUB, rhs);
    } else if (lex.matchKeyword("mul")) {
        lex.eatKeyword("mul");
        lex.eatDelim('(');
        Expression lhs = queryExpression();
        lex.eatDelim(',');
        Expression rhs = queryExpression();
        lex.eatDelim(')');
        return new BinaryArithmeticExpression(lhs, OP_MUL, rhs);
    } else if (lex.matchKeyword("div")) {
        lex.eatKeyword("div");
        lex.eatDelim('(');
        Expression lhs = queryExpression();
        lex.eatDelim(',');
        Expression rhs = queryExpression();
        lex.eatDelim(')');
        return new BinaryArithmeticExpression(lhs, OP_DIV, rhs);
    } else if (lex.matchId())
        return new FieldNameExpression(id());
    else
        return new ConstantExpression(constant());
}
```

Update return

Q: Update return should handle return that larger than 1.

Solution:



```
sql = "UPDATE item SET i_price = " + updatedPrice + " WHERE i_id = " + itemIds[i];

int result = statement.executeUpdate(sql);
if (result == 0) {
    throw new RuntimeException("cannot update the record with i_id = " + itemIds[i]);
}
```

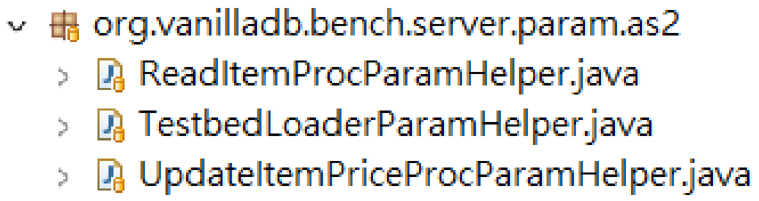
Outline

- Client-side classes
  - *As2BenchmarkRte*
  - *As2UpdateItemPriceParamGen*
  - *UpdateItemPriceTxnJdbcJob*
- Server-side classes
  - *UpdateItemPriceProcParamHelper*
    1. *Reuse Helper*
  - *UpdateItemPriceTxnProc*
- StatisticMgr

Reuse Helper

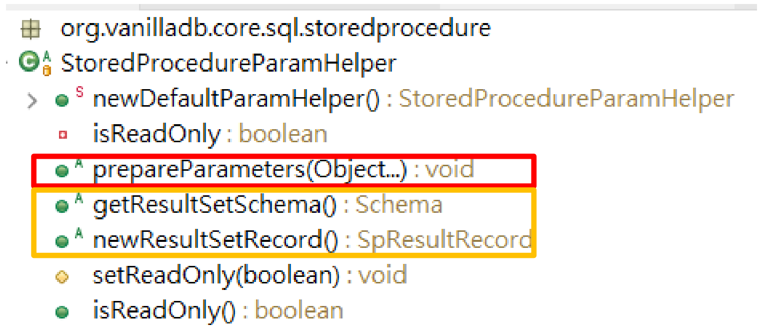
Q: Helper should be called in both JDBC and SP

1. ParamHelper is a server-side class

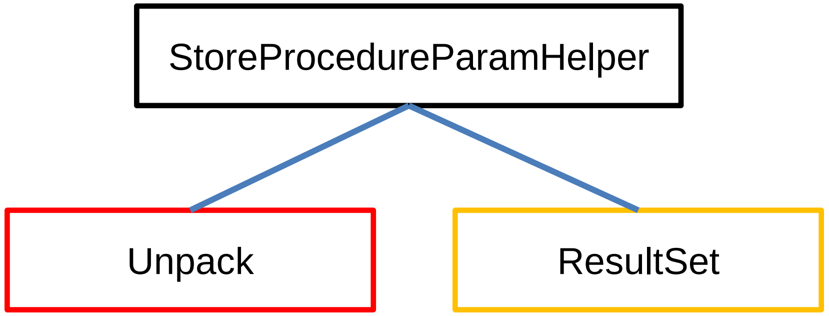


org.vanilladb.bench.server.param.as2
- ReadItemProcParamHelper.java
- TestbedLoaderParamHelper.java
- UpdateItemPriceProcParamHelper.java

2. ParamHelper also prepare resultset.



org.vanilladb.core.sql.storedprocedure
StoredProcedureParamHelper
- newDefaultParamHelper() : StoredProcedureParamHelper
- isReadOnly : boolean
- prepareParameters(Object...) : void
- getResultSetSchema() : Schema
- newResultSetRecord() : SpResultRecord
- setReadOnly(boolean) : void
- isReadOnly() : boolean

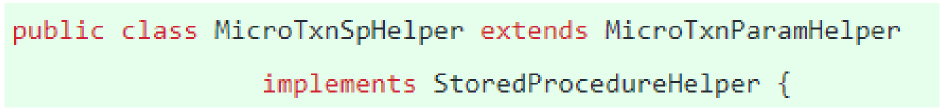


Reuse Helper

Q: Helper should be called in both JDBC and SP

1. All parameter helper put in
org.vanilladb.bench.benchmark.xxx.rte package
2. All sp helper put in
org.vanilladb.bench.server.procedure.xxx package
3. Made StoreProcedureHelper as an interface

Ex:



```
public class MicroTxnSpHelper extends MicroTxnParamHelper
                implements StoredProcedureHelper {
```

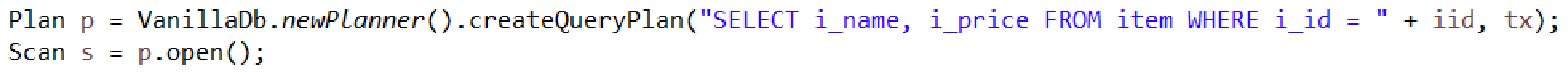
Outline

- Client-side classes
  - *As2BenchmarkRte*
  - *As2UpdateItemPriceParamGen*
  - *UpdateItemPriceTxnJdbcJob*
- Server-side classes
  - *UpdateItemPriceProcParamHelper*
  - *UpdateItemPriceTxnProc*
    1. *Update SQL*
    2. *Update return*
    3. *VanillaDb.newPlanner()*
- StatisticMgr

VanillaDb.newPlanner()

Q: Instead of using StoreProcedureHelper.execute, use the title one

Solution:



```
Plan p = VanillaDb.newPlanner().createQueryPlan("SELECT i_name, i_price FROM item WHERE i_id = " + iid, tx);
Scan s = p.open();
```

Outline

- Client-side classes
  - *As2BenchmarkRte*
  - *As2UpdateItemPriceParamGen*
  - *UpdateItemPriceTxnJdbcJob*
- Server-side classes
  - *UpdateItemPriceProcParamHelper*
  - *UpdateItemPriceTxnProc*
- StatisticMgr
  1. *Accumulate response time*
  2. *High dependency*