Assignment 3 Explanation

Introduction to Database Systems

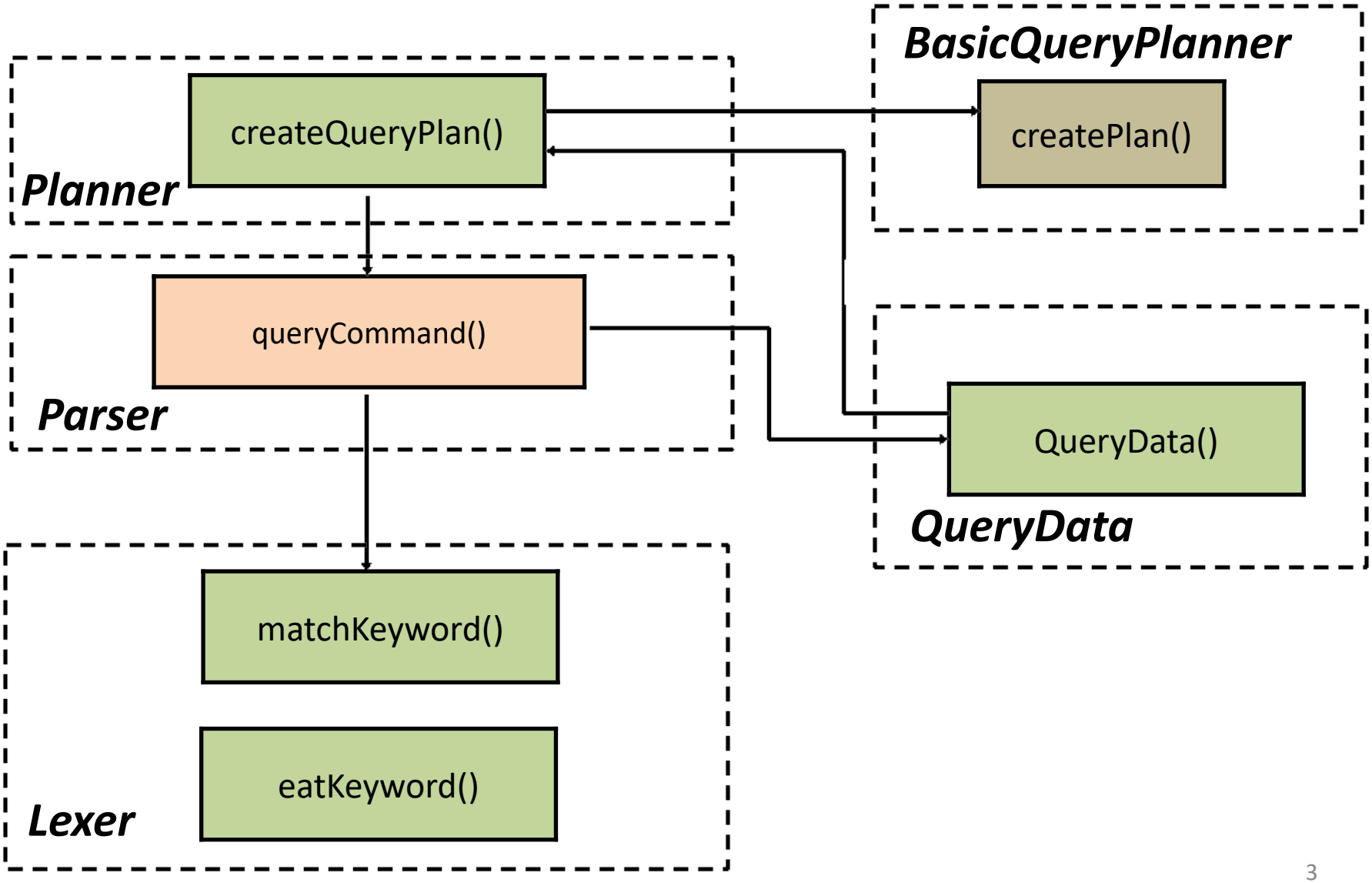
DataLab

CS, NTHU

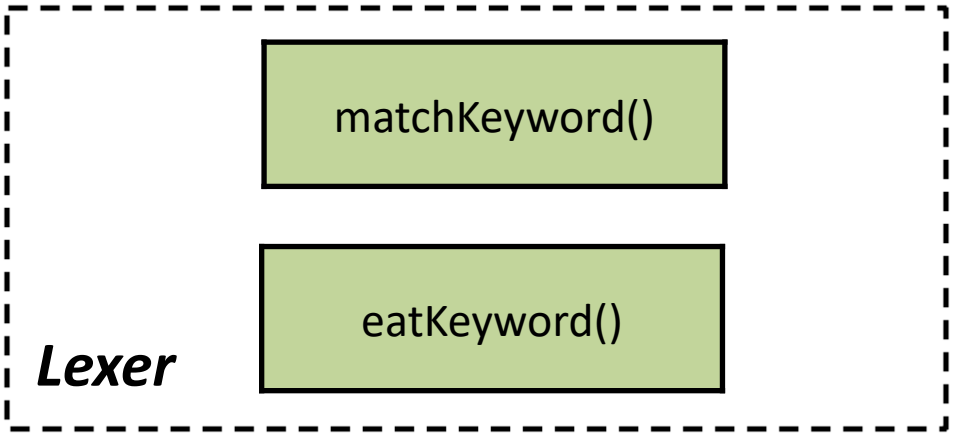
Modified/Added Classes

- Parse
  - *Lexer*
  - *Parser*
  - *QueryData*
- Algebra
  - *ExplainPlan, ExplainScan*
  - *TablePlan, ProductPlan, SelectPlan etc*
- Planner
  - *BasicQueryPlanner*
- An example of Experiment Results

Overview


BasicQueryPlanner
createQueryPlan()
createPlan()
Planner
queryCommand()
Parser
QueryData()
QueryData
matchKeyword()
eatKeyword()
Lexer

Lexer

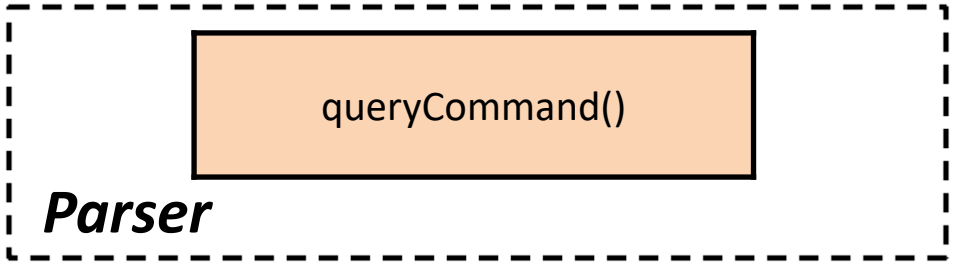


Parse

- Lexer
  - added “explain” in keywords.

```
private void initKeywords() {
    keywords = Arrays.asList("select", "from", "where", "and", "insert",
            "into", "values", "delete", "drop", "update", "set", "create", "table",
            "int", "double", "varchar", "view", "as", "index", "on",
            "long", "order", "by", "asc", "desc", "sum", "count", "avg",
            "min", "max", "distinct", "group", "add", "sub", "mul", "div",
            "explain", "using", "hash", "btree");
}
```

Parser



Parse

- Parser
  - add isExplain

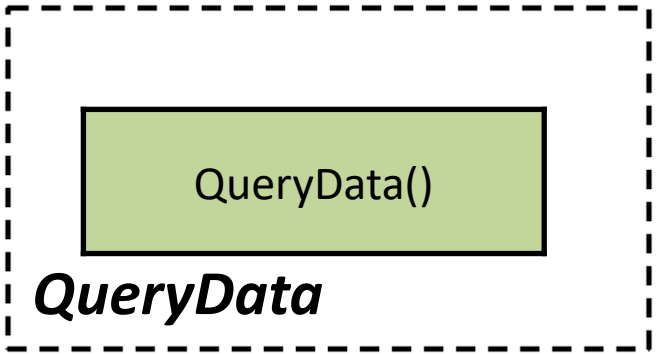
```
public QueryData queryCommand() {
    boolean isExplain = false;
    if (lex.matchKeyword("explain")) {
        isExplain = true;
        lex.eatKeyword("explain");
    }
    lex.eatKeyword("select");
    ProjectList projs = projectList();
```

Parse

- Parser
  - Parser returns SQL data
  - In method “queryCommand()”

```
return new QueryData(isExplain, projs.asStringSet(), tables, pred,
        groupFields, projs.aggregationFns(), sortFields, sortDirs);
```

QueryData



Parse

- QueryData

```
public QueryData(boolean isExplain, Set<String> projFields, Set<String> tables, Predicate pred,
        Set<String> groupFields, Set<AggregationFn> aggFn, List<String> sortFields, List<Integer> sortDirs) {
    this.isExplain = isExplain;
    this.projFields = projFields;
    this.tables = tables;
    this.pred = pred;
    this.groupFields = groupFields;
    this.aggFn = aggFn;
    this.sortFields = sortFields;
    this.sortDirs = sortDirs;
}
```

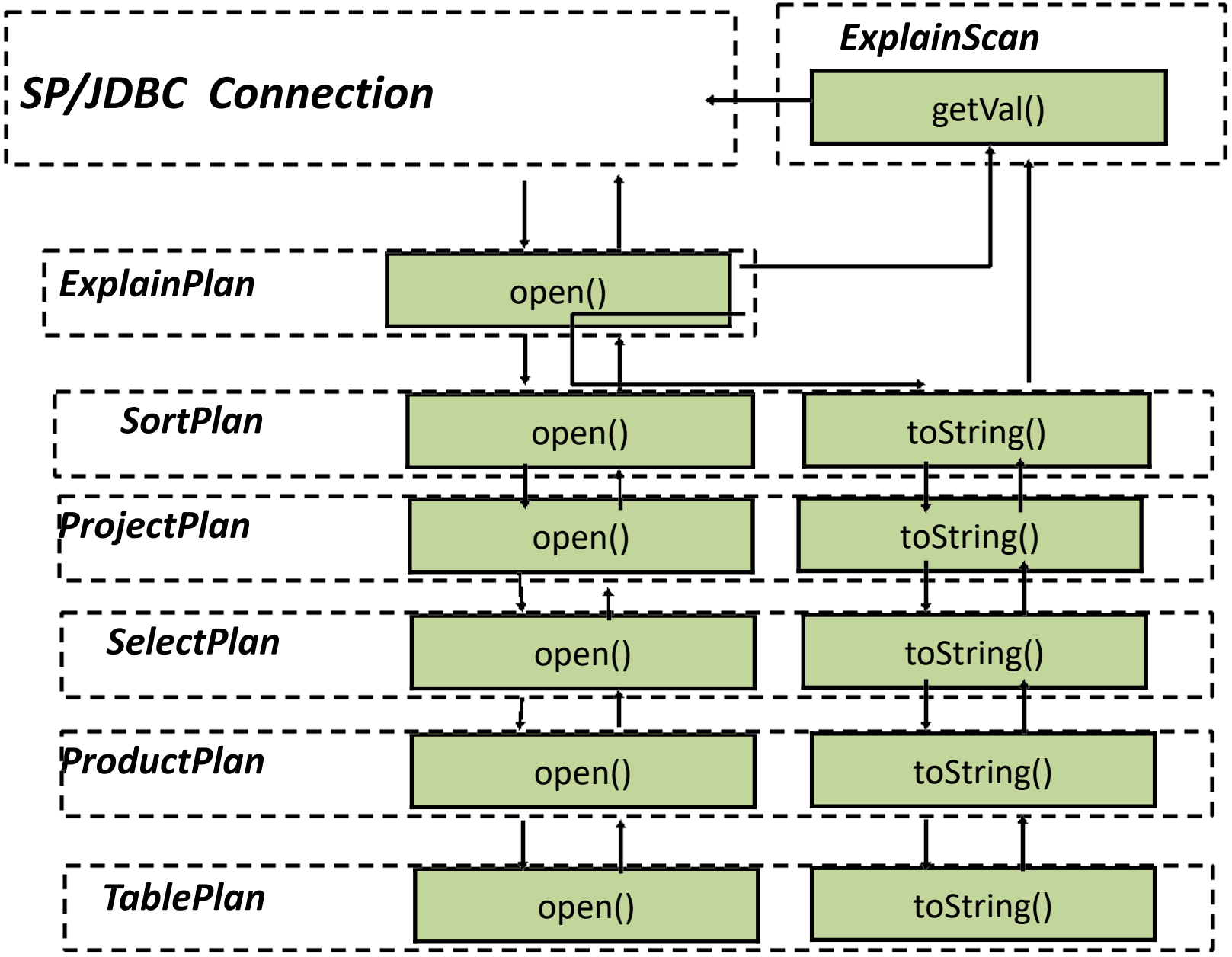
Parse

- QueryData

```
public String toString() {
    StringBuilder result = new StringBuilder();
    if (isExplain)
        result.append("explain ");
    result.append("select ");
```

Modified/Added Classes

- Parse
  - *Lexer*
  - *Parser*
  - *queryData*
- Algebra
  - *ExplainPlan, ExplainScan*
  - *TablePlan, ProductPlan, SelectPlan etc*
- Planner
  - *BasicQueryPlanner*
- Examples of Experiment Results


SP/JDBC Connection
ExplainScan
getVal()
ExplainPlan
open()
SortPlan
open()
toString()
ProjectPlan
open()
toString()
SelectPlan
open()
toString()
ProductPlan
open()
toString()
TablePlan
open()
toString()

ExplainPlan

```java
@Override
public Schema schema() {
    Schema schema = new Schema();
    schema.addField("query-plan", Type.VARCHAR(500));
    return schema;
}
```

```java
@Override
public Scan open() {
    return new ExplainScan(p.open(), schema(), p.toString());
}
```

ExplainScan

- That the result shows once

```
@Override
public void beforeFirst() {
    isBeforeFirst = true;
}

@Override
public boolean next() {
    if (isBeforeFirst) {
        isBeforeFirst = false;
        return true;
    } else
        return false;
}
```

ExplainScan

- Return the result of explain

```
@Override
public Constant getVal(String fldName) {
    if (fldName.equals("query-plan")) {
        return new VarcharConstant(result);
    } else
        throw new RuntimeException("field " + fldName + " not found.");
}
```

ExplainScan

- Return the number of actual records

```
public ExplainScan(Scan s, Schema schema, String explain) {
    this.result = "\n" + explain;
    this.schema = schema;
    s.beforeFirst();
    while (s.next())
        numRecs++;
    s.close();
    this.result = result + "\nActual #recs: " + numRecs;
    isBeforeFirsted = true;
}
```

TablePlan

```
@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("->TablePlan on (").append(ti.tableName())
            .append(") (#blks=");
    sb.append(blocksAccessed()).append(", #recs=").append(recordsOutput())
            .append(")\n");
    return sb.toString();
}
```

->TablePlan on (warehouse) (#blks=2, #recs=1)

ProductPlan

```java
@Override
public String toString() {
    String c2 = p2.toString();
    String[] cs2 = c2.split("\n");
    String c1 = p1.toString();
    String[] cs1 = c1.split("\n");
    StringBuilder sb = new StringBuilder();
    sb.append("->ProductPlan  (#blks=" + blocksAccessed() + ", #recs="
            + recordsOutput() + ")\n");
    // right child
    for (String child : cs2)
        sb.append("\t").append(child).append("\n");
    // left child
    for (String child : cs1)
        sb.append("\t").append(child).append("\n");
    return sb.toString();
}
```

```
->ProductPlan  (#blks=22, #recs=10)
  ->TablePlan on (warehouse) (#blks=2, #recs=1)
  ->TablePlan on (district) (#blks=2, #recs=10)
```

SelectPlan

```java
@Override
public String toString() {
    String c = p.toString();
    String[] cs = c.split("\n");
    StringBuilder sb = new StringBuilder();
    sb.append("->SelectPlan pred:(" + pred.toString() + ") (#blks="
            + blocksAccessed() + ", #recs=" + recordsOutput() + ")\n");
    for (String child : cs)
        sb.append("\t").append(child).append("\n");
    return sb.toString();
}
```

```
->SelectPlan pred:(d_w_id=w_id) (#blks=22, #recs=10)
  ->ProductPlan  (#blks=22, #recs=10)
    ->TablePlan on (warehouse) (#blks=2, #recs=1)
    ->TablePlan on (district) (#blks=2, #recs=10)
```

SortPlan

```java
@Override
public String toString() {
    String c = p.toString();
    String[] cs = c.split("\n");
    StringBuilder sb = new StringBuilder();
    sb.append("->");
    sb.append("SortPlan (#blks=" + blocksAccessed() + ", #recs="
            + recordsOutput() + ")\n");
    for (String child : cs)
        sb.append("\t").append(child).append("\n");
    ;
    return sb.toString();
}
```

```
->SortPlan (#blks=2, #recs=10)
  ->SelectPlan pred:(d_w_id=w_id) (#blks=22, #recs=10)
    ->ProductPlan  (#blks=22, #recs=10)
      ->TablePlan on (warehouse) (#blks=2, #recs=1)
      ->TablePlan on (district) (#blks=2, #recs=10)
```

GroupByPlan

```java
@Override
public String toString() {
    String c = sp.toString();
    String[] cs = c.split("\n");
    StringBuilder sb = new StringBuilder();
    sb.append("->");
    sb.append("GroupByPlan: (#blks=" + blocksAccessed() + ", #recs="
            + recordsOutput() + ")\n");
    for (String child : cs)
        sb.append("\t").append(child).append("\n");
    ;
    return sb.toString();
}
```

```
->GroupByPlan: (#blks=2, #recs=1)
  ->SortPlan (#blks=2, #recs=10)
    ->SelectPlan pred:(d_w_id=w_id) (#blks=22, #recs=10)
      ->ProductPlan  (#blks=22, #recs=10)
        ->TablePlan on (warehouse) (#blks=2, #recs=1)
        ->TablePlan on (district) (#blks=2, #recs=10)
```

ProjectPlan

```java
@Override
public String toString() {
    String c = p.toString();
    String[] cs = c.split("\n");
    StringBuilder sb = new StringBuilder();
    sb.append("->ProjectPlan  (#blks=" + blocksAccessed() + ", #recs="
            + recordsOutput() + ")\n");
    for (String child : cs)
        sb.append("\t").append(child).append("\n");
    return sb.toString();
}
```

```
->ProjectPlan  (#blks=2, #recs=1)
  ->GroupByPlan: (#blks=2, #recs=1)
    ->SortPlan (#blks=2, #recs=10)
      ->SelectPlan pred:(d_w_id=w_id) (#blks=22, #recs=10)
        ->ProductPlan  (#blks=22, #recs=10)
          ->TablePlan on (warehouse) (#blks=2, #recs=1)
          ->TablePlan on (district) (#blks=2, #recs=10)
```

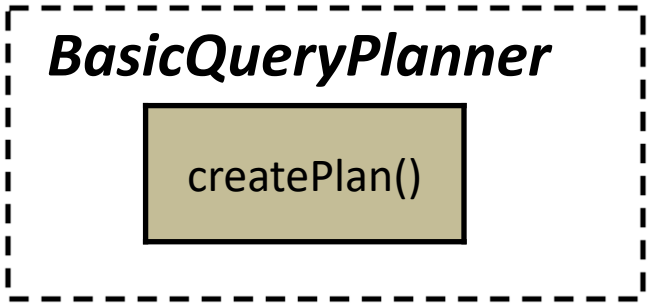
Plan before Scan

- A plan (tree) is a blueprint for evaluating a query
- Estimates cost by accessing statistics metadata only
  - No actual I/Os
  - Memory access only, ***very efficient***
- Once a good plan is decided, we then create a scan following the blueprint

Modified/Added Classes

- Parse
  - *Lexer*
  - *Parser*
  - *queryData*
- Algebra
  - *ExplainPlan*、*ExplainScan*
  - *TablePlan*、*ProductPlan*、*SelectPlan*、 *SortPlan*、*GroupByPlan*、*ProjectPlan*
- Planner
  - *BasicQueryPlanner*
- Examples of Experiment Results

BasicQueryPlanner



BasicQueryPlanner

```
@Override
public Plan createPlan(QueryData data, Transaction tx) {
    // Step 1: Create a plan for each mentioned table or view
    List<Plan> plans = new ArrayList<Plan>();
    for (String tblname : data.tables()) {
        String viewdef = VanillaDb.catalogMgr().getViewDef(tblname, tx);
        if (viewdef != null)
            plans.add(VanillaDb.newPlanner().createQueryPlan(viewdef, tx));
        else
            plans.add(new TablePlan(tblname, tx));
    }
    // Step 2: Create the product of all table plans
    Plan p = plans.remove(0);
    for (Plan nextplan : plans)
        p = new ProductPlan(p, nextplan);
    // Step 3: Add a selection plan for the predicate
    p = new SelectPlan(p, data.pred());
    // Step 4: Add a group-by plan if specified
    if (data.groupFields() != null) {
        p = new GroupByPlan(p, data.groupFields(), data.aggregationFn(), tx);
    }
    // Step 5: Project onto the specified fields
    p = new ProjectPlan(p, data.projectFields());
    // Step 6: Add a sort plan if specified
    if (data.sortFields() != null)
        p = new SortPlan(p, data.sortFields(), data.sortDirections(), tx);
    // Step 7: Add a explain plan if the query is explain statement
    if (data.isExplain())
        p = new ExplainPlan(p);
    return p;
}
```