Team16 Assignment 2 Phase 2 Report

109062123 曹瀚文 109062323 吳聲宏 109062330 黃鈺臻

1. ExplainPlan的添加處不同

Solution code是在BasicQueryPlanner的createPlan中，最後加上ExplainPlan。我們則是在Planner createPlan中，呼叫完BasicQueryPlanner createPlan後再加上ExplainPlan。

加在BasicQueryPlanner會讓邏輯更清楚，可以看出先建table plan，再來是product, group…最後是explain的整體流程。壞處就是同樣的code在不同的QueryPlanner中要重複出現。但應該只有概念上的差別，不會造成效率或正確性差異。

2. Explain message的函式和結構差異

Solution code中是為了每個plan建立一個toString function，產生各個plan相對應的explain message，並直接以string回傳，每個plan會各自去呼叫其subplan的toString function以獲得完整的explain message。我們的邏輯相同，只是額外建了一個explainTree class來儲存explain message 跟相關function，每個plan也有自己的explainTree function會處理其explain message。最後再由explainScan中的RecursiveGenerate function 利用explainTree來產生完整的explain message。

以sort plan為例，solution code和我們的處理方式分別如下：

```
@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();
    }
```

```
@Override
    public ExplainTree explainTree() {
        ExplainTree ret = new ExplainTree(this.getClass().getSimpleName(),
null, this.blocksAccessed(), this.recordsOutput());
        ret.addChildren(p.explainTree());
        return ret;
    }
```

相比solution code，我們做法的好處在每次建立新explainTree時是直接拿subplan的explainTree當作child，沒有重新再創重複的字串，相比string版本的作法會節省一點時間跟

空間。此做法也更好的abstraction，讓每個plan各自的explainTree function更清楚簡潔、免於重複。

相比solution code的不足處有兩個。一是class name這些應該可以根據各個plan寫死的東西，就直接寫死以省時間就好，不用呼叫function。二是我們多為explainPlan建了一個explainTree function，其實應該不用的，直接呼叫他的subplan就好。