Write Up for acmdb-lab4

在 lab4 中，我们实现了数据库中基于成本的优化器(cost-based optimizer)。

Exercise 1 (IntHistogram)

其实现逻辑大概如下：首先先全表扫描一次，获取每个字段的最大值与最小值（目的是为了获得区间范围）。此时就可以建立直方图的初始坐标轴了，一种简单的方法是使用固定数量的桶 NumB，每个桶表示直方图属性域的固定范围内的记录数。例如，如果字段 f 的范围从 1 到 100，并且有 10 个存储桶，那么存储桶 1 可能包含 1 到 10 之间的记录数计数，存储桶 2 可能包含 11 到 20 之间的记录计数，依此类推。然后再次扫描表，选择所有元组的所有字段，并使用它们填充每个直方图中的桶计数。

Exercise 2 (TableStats)

这个 Exercise 主要是完成 TableStats 类与每页的 IO 花销，构造每个字段的直方图，并利用 Exercise1 的方法，得出每个直方图相关的选择性估算。

Exercise 3 (Join Cost Estimation)

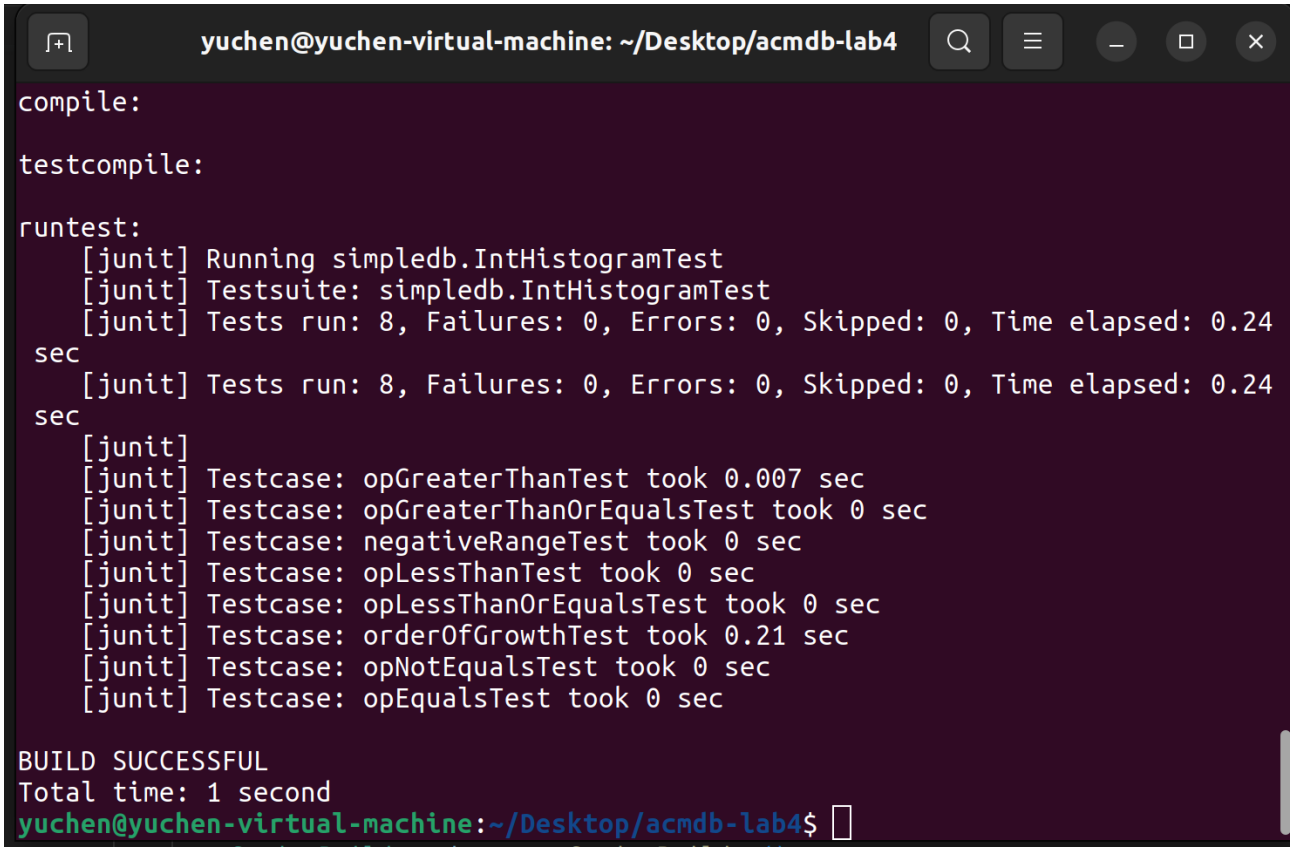
这个 exercise 中完成的是进行对连接成本的估算，根据 lab 中对 join 成本估算的公式完成。

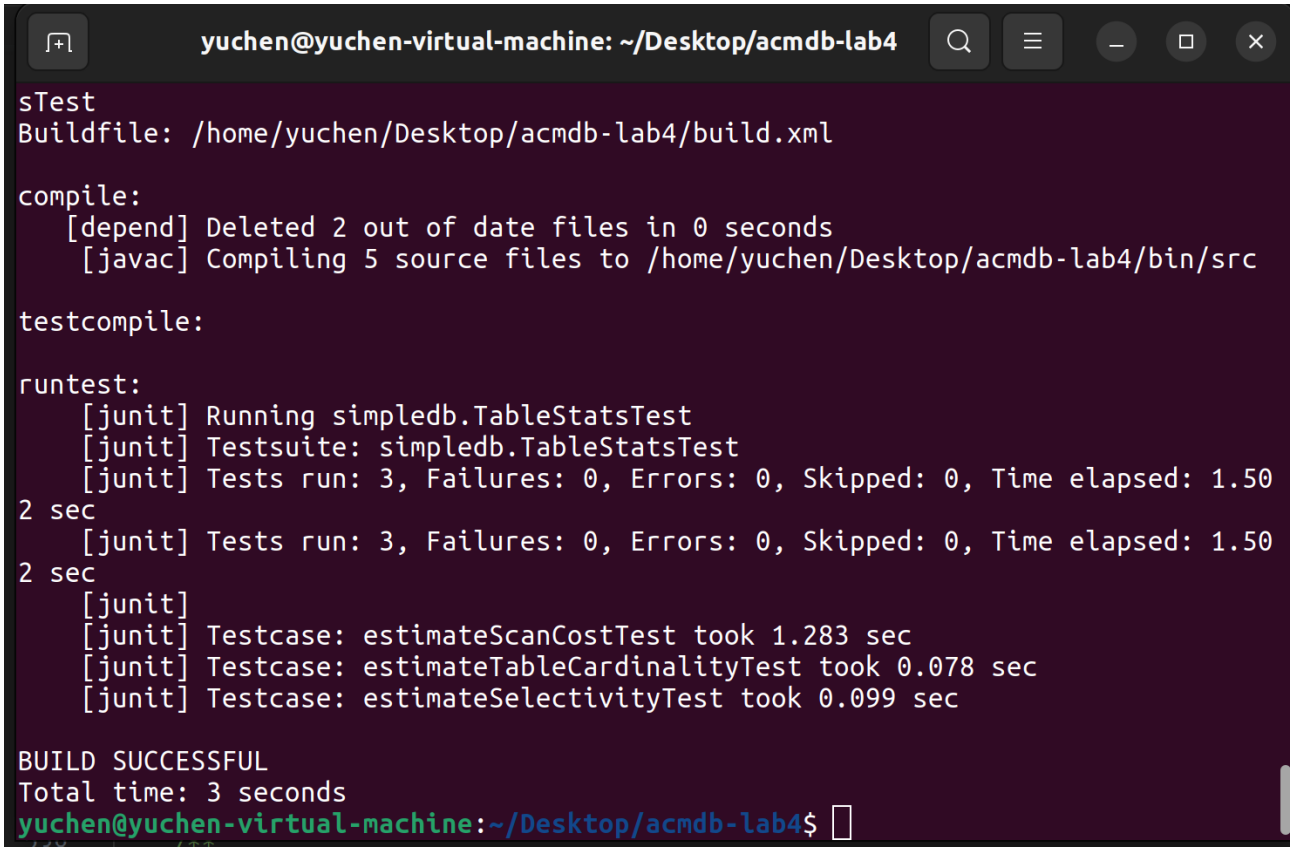
Exercise 4 (Join Ordering)

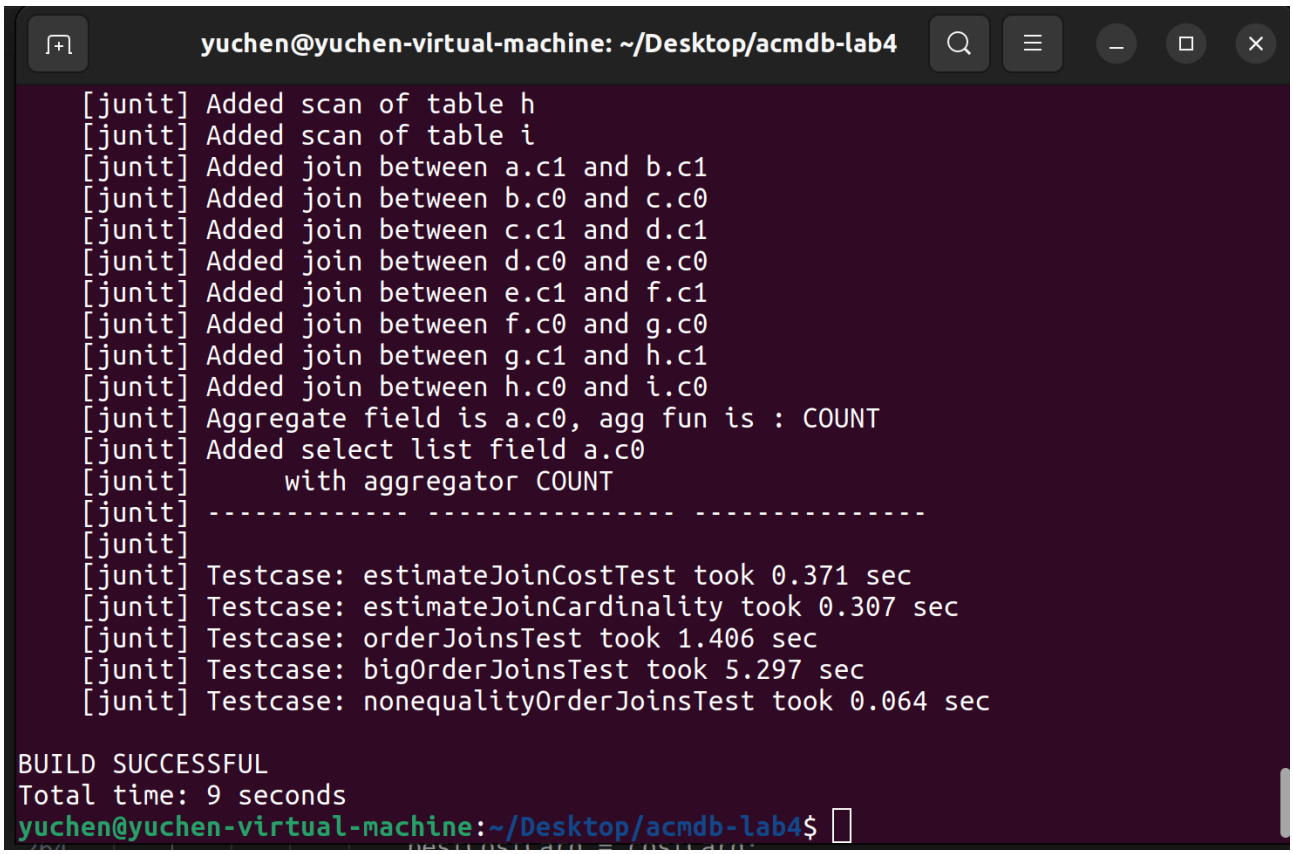
对于 Exercise4 则是总体生成优化过后的连接顺序。根据计算成本的公式，不同顺序的连接成本也不同，这里使用了动态规划的思想完成。

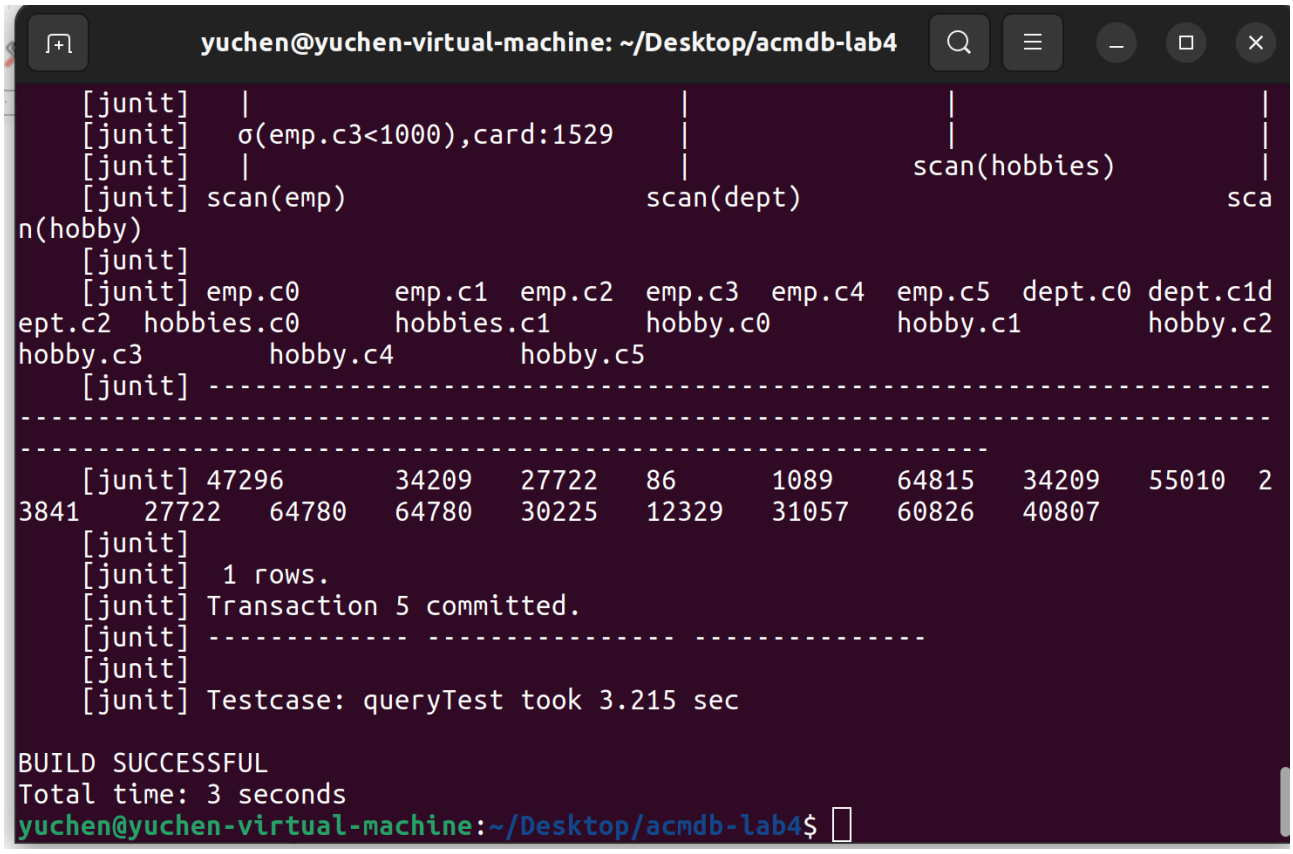
总结

lab4 总体来说理解难度比较高，难度上主要集中在 queryTest 的跑通，有些逻辑要根据测试的逻辑来编写，比较耗费时间。


yuchen@yuchen-virtual-machine: ~/Desktop/acmdb-lab4
compile:
testcompile:
runtest:
[junit] Running simpledb.IntHistogramTest
[junit] Testsuite: simpledb.IntHistogramTest
[junit] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.24
sec
[junit] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.24
sec
[junit]
[junit] Testcase: opGreaterThanTest took 0.007 sec
[junit] Testcase: opGreaterThanOrEqualsTest took 0 sec
[junit] Testcase: negativeRangeTest took 0 sec
[junit] Testcase: opLessThanTest took 0 sec
[junit] Testcase: opLessThanOrEqualsTest took 0 sec
[junit] Testcase: orderOfGrowthTest took 0.21 sec
[junit] Testcase: opNotEqualsTest took 0 sec
[junit] Testcase: opEqualsTest took 0 sec
BUILD SUCCESSFUL
Total time: 1 second
yuchen@yuchen-virtual-machine:~/Desktop/acmdb-lab4$


yuchen@yuchen-virtual-machine: ~/Desktop/acmdb-lab4
sTest
Buildfile: /home/yuchen/Desktop/acmdb-lab4/build.xml
compile:
[depend] Deleted 2 out of date files in 0 seconds
[javac] Compiling 5 source files to /home/yuchen/Desktop/acmdb-lab4/bin/src
testcompile:
runtest:
[junit] Running simpledb.TableStatsTest
[junit] Testsuite: simpledb.TableStatsTest
[junit] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.50
2 sec
[junit] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.50
2 sec
[junit]
[junit] Testcase: estimateScanCostTest took 1.283 sec
[junit] Testcase: estimateTableCardinalityTest took 0.078 sec
[junit] Testcase: estimateSelectivityTest took 0.099 sec
BUILD SUCCESSFUL
Total time: 3 seconds
yuchen@yuchen-virtual-machine:~/Desktop/acmdb-lab4$


yuchen@yuchen-virtual-machine: ~/Desktop/acmdb-lab4
[junit] Added scan of table h
[junit] Added scan of table i
[junit] Added join between a.c1 and b.c1
[junit] Added join between b.c0 and c.c0
[junit] Added join between c.c1 and d.c1
[junit] Added join between d.c0 and e.c0
[junit] Added join between e.c1 and f.c1
[junit] Added join between f.c0 and g.c0
[junit] Added join between g.c1 and h.c1
[junit] Added join between h.c0 and i.c0
[junit] Aggregate field is a.c0, agg fun is : COUNT
[junit] Added select list field a.c0
[junit] with aggregator COUNT
[junit] ------------ --------------- --------------
[junit]
[junit] Testcase: estimateJoinCostTest took 0.371 sec
[junit] Testcase: estimateJoinCardinality took 0.307 sec
[junit] Testcase: orderJoinsTest took 1.406 sec
[junit] Testcase: bigOrderJoinsTest took 5.297 sec
[junit] Testcase: nonequalityOrderJoinsTest took 0.064 sec
BUILD SUCCESSFUL
Total time: 9 seconds
yuchen@yuchen-virtual-machine:~/Desktop/acmdb-lab4$


yuchen@yuchen-virtual-machine: ~/Desktop/acmdb-lab4
[junit] | | | |
[junit] σ(emp.c3<1000),card:1529 | | |
[junit] | | scan(hobbies) |
[junit] scan(emp) scan(dept) sca
n(hobby)
[junit]
[junit] emp.c0 emp.c1 emp.c2 emp.c3 emp.c4 emp.c5 dept.c0 dept.c1d
ept.c2 hobbies.c0 hobbies.c1 hobby.c0 hobby.c1 hobby.c2
hobby.c3 hobby.c4 hobby.c5
[junit] ----------
[junit] 47296 34209 27722 86 1089 64815 34209 55010 2
3841 27722 64780 64780 30225 12329 31057 60826 40807
[junit]
[junit] 1 rows.
[junit] Transaction 5 committed.
[junit] ------------ --------------- --------------
[junit]
[junit] Testcase: queryTest took 3.215 sec
BUILD SUCCESSFUL
Total time: 3 seconds
yuchen@yuchen-virtual-machine:~/Desktop/acmdb-lab4$