

实验报告

一、实验环境

Ubuntu 22.04

Jdk 1.8

Ant 1.10.12

二、实验流程

Exercise 1: IntHistogram.java

IntHistogram 的核心方法是 `estimateSelectivity()`。`selectivity` 的定义：`predicate` 应用在 `table` 后的结果集的 `tuple` 数量占原 `table` 的 `tuple` 数量的比例。

Exercise 2: TableStats.java

TableStats: 通过 `process` 函数，用来计算 `table` 的 `tuple` 数量，计算每一个 `int` 类型的列的最大最小值，计算每一列的 `histogram`。

Exercise 3: Join Cost Estimation

Join Cost 是由 $\text{scan}(t1) + \text{card}(t1) * \text{scan}(t2) + \text{card}(t1) * \text{card}(t2)$ 组成。前两项是 IO 成本，而第三项是循环计算成本。

Exercise 4: Join Ordering

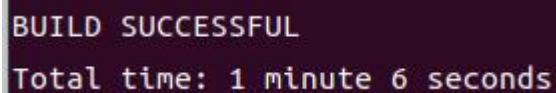
对于 Exercise4 则是总体生成优化过后的连接顺序。根据上述提供的计算成本的公式，不同顺序的连接成本也不同。`orderJoins()` 根据给定各个表的统计数据，与各个表的选择率，返回 `joins` 的最优连接顺序。

Bonus Exercises

主要对 `enumerateSubsets()` 进行优化，提升速度，它依赖于另一个函数 `getSubsetIndex()`。注意到 Join 可以查询具有不同大小的子集，`getSubsetIndex()` 通过位运算预计算大小为 $s < n$ 的子集的所有索引。这里使用两个嵌套循环，外循环的范围从 1 到 $(1 < n) - 1$ ，枚举集合的所有 $2^n - 1$ 非空子集。内循环的范围从 0 到 $n - 1$ ，枚举所选元素的索引。将结果存储在列表中，因此只计算一次，加快了计算速度。

运行结果：

ant test



```
BUILD SUCCESSFUL
Total time: 1 minute 6 seconds
```

ant systemtest

```
BUILD SUCCESSFUL
Total time: 49 seconds
```

三、困难

此次的 lab 相较于上一次，理解难度会高一些，需要理解基数，选择性等概念，而理解后才可以进行编写。所以花了较多时间，总共花了三天时间。