```
//
// Created by Nikolay Yakovets on 2018-02-02.
//

#ifndef QS_SIMPLEEVALUATOR_H
#define QS_SIMPLEEVALUATOR_H


#include <memory>
#include <cmath>
#include "SimpleGraph.h"
#include "RPQTree.h"
#include "Evaluator.h"
#include "Graph.h"

struct exhaustiveIndexes {
    // exhaustive indexes: SOP, PSO, POS, OSP
//    std::vector<std::vector<std::pair<uint32_t,uint32_t>>> SOP;
    std::vector<std::vector<std::pair<uint32_t,uint32_t>>> PSO;
    std::vector<std::vector<std::pair<uint32_t,uint32_t>>> POS;
//    std::vector<std::vector<std::pair<uint32_t,uint32_t>>> OSP;
};

class SimpleEvaluator : public Evaluator {

    std::shared_ptr<SimpleGraph> graph;
    std::shared_ptr<SimpleEstimator> est;
    std::map<std::vector<std::string>, cardStat> cache;
    std::map<std::string, cardStat> estcache;
    exhaustiveIndexes exh_indexes;

public:

    std::vector<std::string> treeToString(RPQTree *q);
    void treeToString(RPQTree *q, std::vector<std::string> &vec);

    explicit SimpleEvaluator(std::shared_ptr<SimpleGraph> &g);
    SimpleEvaluator() = default;

    void prepare() override ;
    cardStat evaluate(RPQTree *query) override ;

    std::vector<RPQTree*> getLeaves(RPQTree *query);
    RPQTree* optimizeQuery(RPQTree *query);

    void attachEstimator(std::shared_ptr<SimpleEstimator> &e);

    std::shared_ptr<SimpleGraph> evaluate_aux(RPQTree *q);
    static std::shared_ptr<SimpleGraph> project(uint32_t label, bool inverse, std::shared_ptr<SimpleGraph> &g);
    static std::shared_ptr<SimpleGraph> join(std::shared_ptr<SimpleGraph> &left, std::shared_ptr<SimpleGraph> &right);

    static cardStat computeStats(std::shared_ptr<SimpleGraph> &g);
    void createAggregateIndex();
    void createExhaustiveIndex();
    std::shared_ptr<SimpleGraph> project_agg_index(uint32_t label, bool inverse, std::shared_ptr<SimpleGraph> &g);
    std::shared_ptr<SimpleGraph> project_exh_index(uint32_t label, bool inverse, std::shared_ptr<SimpleGraph> &g);

};


#endif //QS_SIMPLEEVALUATOR_H
```