CSE 562: Project #2

In this project you will build a query parser and evaluator that will use your code from project 1 (you may change your code from project #1 as much as you wish).

The following subset of SQL2 should be supported:

1. Relation definitions of the form:

```
CREATE TABLE Table_Name(
A1 VARCHAR2(N1),
…
An VARCHAR2(Nn));
```

Additionally, one attribute is designated as the primary key. For this attribute a primary index should be automatically built (primary key designation should follow the Oracle syntax, i.e., "`A VARCHAR2(N) PRIMARY KEY`").

2. `DROP TABLE Table_Name;`

3. Index definitions of the form:

```
CREATE INDEX Index_Name ON Table_Name(Column_Name);
```

4. `DROP INDEX Index_Name;`

5. SQL2 single-tuple INSERT commands like:

```
INSERT INTO Table_Name (Column_Name_1, …, Column_Name_n)
VALUES (Value_1, …, Value_n);
```

6. The `@File_Name` command to execute an .sql file.

7. Queries of the following form:

```
SELECT Column_Name_1, … , Column_Name_k   (or just *)
FROM Table_Name_1, …, Table_Name_m
WHERE Condition;
```

`Condition` is a conjunction of atomic equality conditions of the form R.A = c or R.A = S.B where A, B are attributes, c a constant, and R and S different table names. Assume there are at most two different table names in a query, i.e., m <= 2.

You are supposed to implement:

1. A parser for the input;

2. A query evaluator for the above subset of SQL2, whose repertoire includes the following operations:
- table scan
- index scan
- filter
- Cartesian product
- nested-loops join
- index join.

3. Display of the selected evaluation plan. (The format will be posted 2 weeks before deadline).

The evaluator should pick an evaluation plan that uses existing indexes as much as possible. The evaluator should use the implementation of the file and index operations developed in Project 1. Operations should be pipelined, no intermediate results should be created.

Extra credit

You may get up to 5% of the final grade for the course for implementing any of the following:

- cost-based optimization in the presence of more than two tables
- aggregation with GROUP BY.