MDE518: Database Management Systems
University of Athens
Deprt. of Informatics & Telecommunications

*Programming Project II*
January 2010

**Purpose of the Project:**
The goal of this project is that you develop a simplified a Query Processor and Oprimizer (QPO) for relational algebra expressions. The algebra will involve selections, projections, (multi-way) joins and possibly aggregate functions and pertinent expressions that will have to be processed and optimized.

Expressions will be provided by users with a convenient input mechanism such as a string in a command line.

QPO is expected to work closely with a detailed catalog for a specific database. The catalog offers information regarding the nature and the way data are organized on the disk, existing indexing structures as well as statistics available for (portions of the) relations involved in the processing.

Your query processor should yield an annotated query execution plan that is considered "best" by your QPO. The plan should take into account all structural information available, query processing techniques deployed by the QPO and constraints pertinent to either logical or physical resources (such as buffer space available).

**The Catalog:**
The catalog shall contain information regarding the tables and their organization as well as system wide information including the number of pages in the buffer and the size of the pages. In particular, for every table the catalog will include the following information:

1. The *relation name*, the *file name* (in which the relation is being stored) and the *file structure* used to organize the file.
2. The *name* and *type* for all attributes that are part of the *relation*.
3. Attributes used as *primary*, *secondary* or *foreign* keys.
4. The name for all indexes that have been build and exist of various individual attributes. For these indexes, the catalog will provide the type of structure used (B+, Static Hashing, Extensible-Hashing) and of course the attribute used as the key.
5. Statistics pertinent to the relation including:
   - The *cardinality* of the *relation*.
   - The *size* of the tuple.
   - For each indexed attribute the number of *distinct key values* participating in the indexing mechanism.
   - Height of every B+–index found in the catalog.
   - For (selected) attribute(s), the number of unique values [1] appearing per domain in a relation as well as corresponding *min* and *max* values (if applicable). Alternatively, histogram information could be provided in here.
6. System wide parameters including:

[1] V(A,r) where A is the attribute name and r the relation name

- Number of *buffers* dedicated for processing.
- *Size* of each buffer/page.
- Average *latency* as well as *transfer time* for bringing (random/sequential) pages from disk to main memory and time penalties for writing pages (of intermediate/final results) to the disk.

7. Any other piece of information that might be useful in your processing and optimization approach.

**Input to** QPO**:**
User-input will consist of only well-formed algebraic expressions consisting of (at least) selections, projections and joins [2].

The structure of these operators is expected to follow format discussed in class.

1. $\sigma[cond](r)$ or $sel\ [cond]\ (r)$
2. $\pi[A_1, A_2, ..., A_n](r)$ or $proj\ [A_1, A_2, ..., A_n](r)$
3. $(r) \bowtie [cond](s)$ or alternatively, $\bowtie [cond](r)(s)$ or $join[cond](r)(s)$

In the above expressions, we consider $cond$ to be boolean expression formed either as simple atoms or more complex conjunction or disjunctions or atoms and $r$ and $s$ relations upon which the operators work.

For example the algebraic-tree of the Figure 1 could be entered in a command lines as:

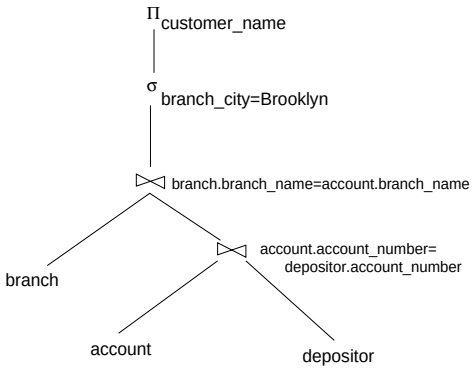


Figure 1: Sample Query Tree

```
>>> proj[customer_name)(sel[branch_city=Brooklyn](
      join[branch.branch_name=account.branch_name)(branch)(
            join[account.account_number=depositor.account_number](account)(depositor)))
```

**Techniques Used:**
You have the freedom to deploy any technique(s) you elect in dealing with the relational operators.

For each such operator you should consider at least two alternatives with the only exception of joins where you will have to consider at least three methods for query materialization. In the latter you could use for instance: nested-loops, sort-merge and a hash-based approach. If convenient, you should consider external-sorting as well.

[2] you could include aggregate and grouping operations if you wish as well

QPO should also examine the option of using pipelining for specific joins that are appropriate to such an operation or intermediate product outcome.

In light of multi-way joins, you may either opt for a deep-left tree or simply resort in a linear programming approach (of your own choice).

**Overall Organization:**
QPO should receive as input a string indicating the query-to-be-processed, create an initial query tree, and subsequently in conjunction with the catalog perform possible syntactic transformations and cost-based optimizations to provide an annotated execution plan. Nominally, the execution plan should be better –in terms of I/O processing– than the originally submitted plan (and your QPO should provide explanations for this fact).

Figure 2 depicts a QPO-produced query evaluation plan in a tree-like layout.

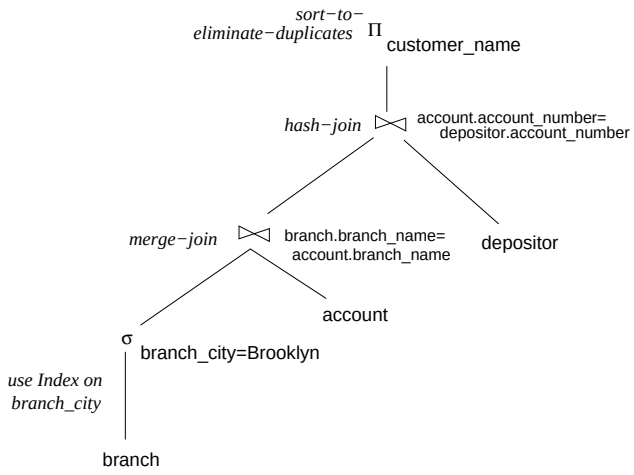


Figure 2: Optimized Query Tree

**Constraints:**

1. You can pick **one** partner to work with and implement the project.
2. You can use any language you want.
3. The function and design of the input module and of the internal structures of QPO are entirely up to you. The same goes for the organization and maintenance of the catalog. To make things more manageable, the structure of the catalog should remain as simple as possible.
4. It is rather important that you can provide -in some concise format- all plans being considered by your QPO and list respective cost estimations.

**What you need to submit:**

- All your work in a tar-ball.
- A design document where you present your ideas/decisions about your program. This should be only a few pages long.
- A methodology (ie, a way) that helps demonstrate the plans produced, the estimations carried out and the final selection of the “optimal” query plan.

**Deadline and Demonstration:**

1. The deadline of the project is the end of the semester (after the final examinations period).
2. You will have to demonstrate your work.