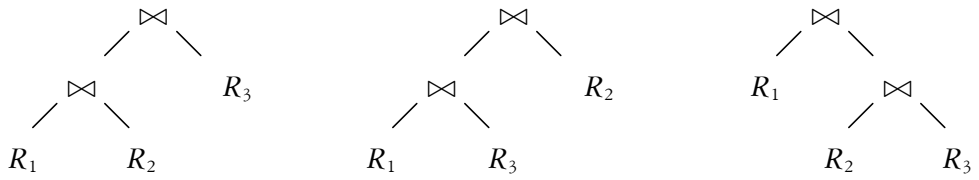
CPS 216 Spring 2003
Homework #4
Assigned: Wednesday, April 9
Due: Wednesday, April 23

Note: If you have already taken CPS 196.3, you should complete Problems 3-5. Otherwise, you may complete Problems 1-3 and 5.

Problem 1.

How many possible plans are there for an n-way join query $R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n$, if we use only one type of asymmetric binary join operator in our plans? Your answer should be a closed-form or recurrence formula in terms of n. Also, compute your answer for $n = 7$.

Remember to consider all bushy plans—not just left-deep ones. For example, three possible plans for $n = 3$ are shown below. There are a total of 12 plans for $n = 3$.



Problem 2.

Consider tables $R(A, B, C)$, $S(C, D)$, and $T(D, E)$. Transform the following query into an equivalent query that:

- Contains no cross products;
- Performs projections and selections as early as possible.

(a) $\pi_{R.B,\, S.D,\, T.E}\, \sigma_{(R.A=10) \text{ and } (R.C = S.C) \text{ and } (S.D = T.D) \text{ and } (R.A > T.E)}\, (R \times S \times T)$

Suppose we have the following statistics:

- $| R | = 1{,}000$; $| \pi_A R | = 1{,}000$; $| \pi_B R | = 100$; $| \pi_C R | = 500$;
- $| S | = 5{,}000$; $| \pi_C S | = 300$; $| \pi_D S | = 10$;
- $| T | = 4{,}000$; $| \pi_D T | = 4{,}000$; $| \pi_E T | = 1{,}500$.

Estimate the number of the rows returned by the following queries:

(b) $\sigma_{A=10}\, R$
(c) $\sigma_{A=10 \text{ and } B = \text{“Bart”}}\, R$
(d) $\sigma_{A=10 \text{ or } B = \text{“Bart”}}\, R$
(e) $R \bowtie S$
(f) $R \bowtie S \bowtie T$

For the following question, further suppose that:

- Each disk/memory block can hold up to 10 rows;
- All tables are stored compactly on disk (10 rows per block) in no particular order;
- No indexes are available;
- 11 memory blocks are available for query processing.

(g) What is the best execution plan (in terms of number of I/O's performed) you can come up with for the query $\sigma_{R.B = \text{"Bart"} \text{ and } S.D = 100} (R \bowtie S)$? Describe your plan and show the calculation of its I/O cost.

Problem 3.

Consider the following relational schema and SQL query. The database stores information about employees, departments, and company finances (organized on a per-department basis).

```
CREATE TABLE Emp(eid INT NOT NULL PRIMARY KEY,
                 did INT, salary INT, hobby CHAR(20));
CREATE TABLE Dept(did INT NOT NULL PRIMARY KEY,
                  name CHAR(20), floor INT, phone CHAR(10));
CREATE TABLE Finance(did INT NOT NULL PRIMARY KEY,
                     budget FLOAT, sales FLOAT, expenses FLOAT);
SELECT d.name, f.budget
FROM Emp e, Dept d, Finance f
WHERE e.did = d.did AND d.did = f.did
AND d.floor = 1
AND e.salary > 59000
AND e.hobby = 'yodeling';
```

(a) Identify a logical plan (using relational algebra operators) that reflects the order of operations a decent query optimizer would choose.

(b) List all join orders (i.e., the orders in which pairs of tables are joined to compute the query result) considered by a query optimizer that follows the heuristic of considering only left-deep plans without cross products.

(c) Suppose that the following information is available:

- There are primary (clustering) indexes on all primary keys.
- There are secondary (non-clustering) indexes on `Emp(did)`, `Emp(salary)`, and `Dept(floor)`.
- There are a total of 50,000 employees and 5,000 departments (each with corresponding financial information) in the database.
- The system's statistics indicate that employee salaries range from 10,001 to 60,000 (inclusive), employees enjoy 200 different hobbies, and the company has two floors in the building.

For each of the query's base tables (`Emp`, `Dept`, and `Finance`), estimate the number of rows that would be initially selected if all selection predicates were pushed down as much as possible.

(d) Suppose that the database system has only one join method available—index nested-loop join. Using the same information in (c), which of the join orders considered by the optimizer in (b) has the lowest estimated cost?

Problem 4.

PostgreSQL has an unconventional way of carrying out query optimization based on genetic programming. Unfortunately, documentation is very scarce on this subject. You should start with `http://developer.postgresql.org/docs/postgres/geqo.html` and get yourself familiarized with the structure and terminology of the GA (Genetic Algorithm). You might need to delve into the source code in `/usr/research/proj/dbgroup/postgresql-7.3.1/src/backend/optimizer/geqo/` on rack40. For questions (a)-(c) below, you can base your answers either on what is currently implemented in PostgreSQL, or on how you think GA should be implemented. Keep your answers at a high level (do not copy and paste source code!), and no more than one paragraph for each question. State clearly whether your answer is based on PostgreSQL or your own design.

(a) What does/should "recombination" do to a set of plans?
(b) What does/should "mutation" do to a set of plans?
(c) What does/should "selection" do to a set of plans?
(d) Name one advantage and one disadvantage of GA compared with Selinger-style query optimization.

Problem 5.

For this problem, you may work either independently or in groups of two or three. If you are not familiar with C++ programming, you may substitute Problems 6 for this problem.

Like in previous two homeworks, you are going to build a small piece of a database management system dubbed "DB216." This time, the focus is on histograms. To get started, copy the source files to you home directory using "`cp –r ~cps216/db216-hw4/ ~/`" on `rack40`. Go into directory `db216-hw4/` and type "`make clean`", and then "`make`". After compilation completes, you will find an executable file `db`. Typing "`./db`" creates a database file `test.db` (if one does not exist already) and runs a number of tests on it.

For this problem, turn in a pointer to the directory containing your documentation, source code, and executable. Make sure the directory is readable to the TA after the homework is due. The directory should contain a `README` file documenting any known bugs/limitations, additional test cases or nifty features you have implemented, and instructions to compile and run your program. In the code, clearly indicate which parts are new or modified.

You need to implement an equi-height histogram in `src/stat/Histogram.{h,cpp}`. The existing code partially implements a trivial histogram with a single bucket. To make your life easier, you only need to deal with integer-valued columns. Modify `src/main.cpp` to run the tests. To exercise your code further, you are encouraged to develop your own test cases in `src/test/hw4.{h,cpp}`.

Problem 6.

If you are not familiar with C++ programming, you may substitute this problem for Problem 5. You must complete this problem independently.

For this problem, you are going to experiment with the query optimizer in IBM DB2. You can see the execution plan chosen by the optimizer using the following command in shell:

```
dynexpln -d cps216 -q "query" -g -t
```

Here, *query* is a string containing the SQL query (with no terminating "`;`") that you wish to optimize. The output should be fairly self-explanatory. For the questions below, write meaningful queries over the system catalog tables including `syscat.tables`, `syscat.columns`, `syscat.indexes`, etc. To see what information is available in these tables, you can use the `describe` command:

```
db2 "describe select * from table"
```

It describes the schema of the specified *table*.

(a) Write a query for which the optimizer chooses a nested-loop join (`NLJOIN`).
(b) Write a query for which the optimizer chooses a sort-merge join (`MSJOIN`).
(c) Write a join query involving three tables. Note the join order chosen by the optimizer.
(d) Write another join query that joins the same three tables as in (c), for which the optimizer chooses a join order that is different from the one in (c). You may use different selection and join conditions.
(e) Try several queries involving equality and range selection conditions on `syscat.tables.tableid`. Based on the estimated cardinality of output (found in `dynexpln` output), can you guess whether DB2 has a histogram on this column? If not, what is DB2's strategy for estimating selectivity factors?

For (a)-(d), turn in a script of running `dynexpln` on these queries. You do not need to run the queries themselves. For (e), turn in a short answer.