UNIVERSITY OF WISCONSIN–MADISON
COMPUTER SCIENCES DEPARTMENT

CS 564: DATABASE MANAGEMENT SYSTEMS

Assignment 7: Relational Query Optimization
Due: Dec ???, 1996, 5 p.m.

Instructor: Raghu Ramakrishnan

1 Introduction

In this assignment, you will carry out a number of exercises involving the optimization of relational queries using the Minibase optimizer and visualization tool. You will not have to write any code, and your answers should be turned in as a text file. *You will carry out this assignment in teams of two with the same partner as for previous assignments.*

2 Available Documentation

The chapter on optimization and the lecture transparencies should provide all the background information that you need on query optimization. The tool itself has an on-line *help* feature, and additional information is included in this handout as part of exercise descriptions.

3 Setting Up Minibaseview

To use the Minibase query optimization viewer, add following lines to your .cshrc.local file:

```
setenv MINIBASE_ROOT   /p/coral/564/minibase
setenv PLATFORM        `$MINIBASE_ROOT/bin/sysname`
```

and, after your path statements, add:

```
set path = ($path $MINIBASE_ROOT/bin)
```

Then, log out and log back in. This is important for the above changes to take effect. The program to run is called 'minibaseview'.

4 Getting Started

These exercises are based on three catalogs, which are located in files *catalogA*, *catalogB* and *catalogD*. Each exercise says which catalog to use. The first step of each exercise should be to open the appropriate catalog, which you do by choosing *Open* from the *Catalogs* menu of the visualization tool. Then you enter your query by choosing *Enter SQL* from the *Query* menu of the visualization tool.

Catalogs A and B are used in Questions 1 to 11 and are based on the same schema. There are three relations, *Emp, Dpt*, and *Job*, which represent employees, departments, and job titles of some organization. Each of these relations has an integer field which is used as the primary key for the relation—*empid* for the *Emp* relation, *dno* for the *Dpt* relation, and *jno* for the *Job* relation. The *Emp* relation has foreign keys referring to *Dpt* and *Job*, representing each employee's department and job title, and the *Dpt* relation has a foreign key referring to *Emp*, representing the department's manager.

Catalog D is used for all of the multi-column index exercises (Questions 12 to 15). A *multi-column index* on a relation is composed of an *ordered* sequence of a subset of the fields of that relation. Note that *catalogD* is much more complicated than the simple *catalogA* and *catalogB* files. This is because *catalogD* can be used in a realistic database—the special relations *relcat*, *attrcat* and *indcat* allow the catalog to describe itself and the rest of the relations in the database. However, you will not be using any of these special relations in this exercise. The second thing you should note is the way multi-column indexes are specified—via the use of percent signs. The order in which the field names are specified is also important (and this will be demonstrated in the exercises). CatalogD has five incarnations of the well-known and widely-used *emp* relation. Each of the Emp relations has its indexes built on either two or three of its fields (*ename*, *title* and *dname*).

5 The Exercises

1. This question gets you to explore "Level 0" of the optimizer visualization display. This level shows all the access methods for each relation involved in the query. If you double-click the LEFT mouse button on one of these access method boxes, you can also see a list of the relation's attributes with the range of values each attribute takes on.

   Open Catalog A and run the tool on the following query.

   select * from dpt
   where mgrid = 20223

   Answer the following questions, looking at level 0 access methods.

   (a) What are the attributes of the *Dpt* relation?

   (b) How many tuples are in the *Dpt* relation?

   (c) How many different values of the *mgrid* attribute are there in the *Dpt* relation?

   (d) What is the largest manager rid (i.e., mgrid) held by a manager?

2. Level 1 of the optimizer visualization tool shows plans that consist of walking a single relation using one of the access methods defined for that relation. If you double-click the LEFT mouse button

on a level 1 plan (and higher levels too), its box expands to show you more detailed information about the plan and the estimates made about it.

Use the same catalog, Catalog A, and the following query:

select * from emp
where sal = 58000

Answer the following questions, looking at level 1 plans.

(a) What is the estimated total cost of running the (estimated) best plan?

(b) What is the sorting cost associated with this plan? What would the sort cost be for *any* plan for this query?

(c) What is the estimated result cardinality for this plan? The estimated result cardinality is the number of employees that the optimizer estimates have the salary 58000.

(d) Given your knowledge about the number of different *sal* values, is the estimate the optimizer makes a reasonable one? Explain very briefly.

3. If you double-click the RIGHT mouse button on a Level 1 node (similarly, for higher level nodes), the tool opens a separate window and displays the entire plan denoted by that node.

   Use the same catalog, Catalog A, and the same query as before:

   select * from emp
   where ename = "Frank"

   Once again, open Catalog A and run this same query as in questions 1 and 2. Answer the following questions, looking at both Level 1 plans and Level 0 access methods.

   (a) Find the (estimated) best plan. Which access method does this plan use?

   (b) In what order would the tuples be returned by this plan? Why?

   (c) Disable the access method used by the best plan by double-clicking the RIGHT mouse button on its Level 0 box. (The box will then be displayed in black, on a color monitor.) Which access method does the optimizer consider to be the best now?

   (d) Compare the new best plan with the other plans still shown. What is it about the way the best plan accesses tuples that makes it substantially cheaper than the others?

4. Open Catalog A, and run the tool on the following query.

   select * from emp
   where empid < 25000

Answer the following questions.

(a) How many Emp tuples does the optimizer think there are? Where on the tool's display is this information shown?

(b) How many employees whose id number is less than 25000 does the optimizer think there are? Where on the display does this appear?

(c) How does the optimizer arrive at its estimate of the number of employees whose id number is less than 25000? That is, what calculations does it perform, and where does the supporting data come from?

(d) Open Catalog B, then run the same query against this catalog. *(WARNING: opening the new catalog does NOT automatically re-run the current query. You must re-run the query manually.)* What is the new estimate of the number of employees whose id number is less than 25000? Why is it different from before?

5. Open Catalog A, and run the tool on this query.

```
select ename, dname from emp, dpt
where emp.dno = dpt.dno and emp.jno < 75
```

Answer the following questions, looking at the best plan.

(a) What kind of join is the best plan?

(b) How many tuples does the optimizer estimate it will retrieve from the Employee relation? At what level is this information shown?

(c) What calculation does the optimizer do to arrive at the reduction factor estimate for this node? Where does the underlying information come from (the same node as in the previous question)?

(d) How much memory does the optimizer estimate it will take to retrieve the tuples needed from the Employee relation?

(e) How much memory does the optimizer estimate it will it take to run the whole query?

6. Levels 2 and higher of the optimizer tool show the result of joining a lower-level plan with an access method. If you double-click the RIGHT mouse button on a Level 2 plan, the tool opens a window showing the lower-level plans and access methods that this plan is made up of.

Open Catalog A, and run the following query.

```
select * from emp, job
where emp.jno = job.jno
```

Answer the following questions looking at Level 2 plans. (NOTE: be sure not to confuse LEVEL 2 with the line NUMBERED 2. The NUMBER of the line of plans you are to look at is 3, the top line of the display.)

(a) What kind of join is the best plan?

(b) What is the estimated total cost of running this plan?

(c) How many tuples does the optimizer estimate will result from this query?

(d) How big are the tuples that result from this query? (i.e., how many bytes does each tuple occupy?)

(e) How many pages will be filled by the result tuples?

(f) What access methods does the best plan use?

7. Open Catalog B, then answer the following questions referring to these queries:

```
select ename, dname from emp, dpt, job
where emp.dno = dpt.dno and emp.jno = job.jno
and dpt.dname = job.jname
```

(a) Run this query and describe the best plan: list the joins and access methods it uses, and the order in which the relations are joined.

(b) Modify the query by replacing the last '=' (comparing names) with '<', and optimize it again. Describe the best plan.

(c) Explain why there is such a big difference in the plans produced for the query and for its (slightly) modified version.

8. From the *Switches* menu it is possible to disable *index-only* plans. If such plans are disabled, the optimizer will only generate plans that always retrieve tuples from the underlying relation, without checking to see if the index search key (and therefore the data entries in the index) contain all necessary information.

Run the tool on the following query, which lists the number of employees that are at each department if that number is greater than 10. Use Catalog A.

```
select dno, count(*) from emp
group by dno
having count(*) > 10
```

(a) Why is this query a candidate for an index-only plan?

(b) What is the estimated cost of this query when index-only plans are allowed?

(c) You can infer the algorithm that the optimizer assumes by the cost formula it displays. Describe the best plan produced by the optimizer when index-only plans are allowed.

(d) What is the cost of the best plan for this query when index-only plans are disallowed?

(e) Describe the best plan produced by the optimizer when index-only plans are disallowed.

9. Run the following query using Catalog B.

select * from emp, dpt
where emp.dno = dpt.dno and emp.jno < 100

(a) Find the largest number such that replacing the 100 with it in the above query makes the best plan into an Index Nested Loops join with a file scan for the *Emp* table.

(b) Find the largest number such that replacing the 100 with it makes the best plan into an Index Nested Loops join using the index on the *jno* column of the *Emp* table.

(c) Repeat the previous two questions using Catalog A.

10. From the *Switches Toggle Joins* menu of the optimizer visualization tool, you can disable and re-enable the different kinds of joins that the optimizer will consider.

Also, if you double-click the RIGHT mouse button on the box that labels any row of the tool's display, you will turn off (and back on) the pruning of plans that the optimizer estimates are not the 'least expensive'. (Remember that for each subset of relations, for each ordering of generated tuples, only the least cost plan is retained. The toggling referred to here controls this pruning; it should be used with care since the number of retained plans can increase rapidly without pruning!)

Run the following variant of the query from the previous exercise, using Catalog B as before.

select ename, dname from emp, dpt
where emp.dno = dpt.dno and emp.jno < 100

Now suppose you knew that the run-time environment for this query was limited to 10 pages of memory. Find the best (left-deep) plan that fits this requirement and describe its join method, access methods, and total cost.

11. Consider the following query, as run against Catalog B. The query lists the number of employees hired before their manager for each of the first 50 departments.

select M.ename, M.empid, count(*)
from emp E, emp M, dpt D where E.dno = D.dno and D.mgrid = M.empid and E.dno
≤ 50 and E.empid < M.empid group by M.empid, M.ename

(a) Why does the best plan not use the index on *E.dno*?

(b) What additional index do you have to create so that the best plan uses an index on *E.dno* ?

(c) Find one single access method such that, when you disable it, the best plan uses only file scans.

12. This question gets you to explore what query plans are produced when your query does not use all of the columns of a multi-column index, and when your query does use all the fields.

Using Catalog D, type in the following query.

select * from emp1
where ename < "Brian"

Answer the following questions looking at Level 1 plans.

(a) What query plans are produced for this query?

(b) What would be the cost of executing the query?

Using Catalog D again, type in the following query.

select * from emp1
where ename < "Brian" and title = "Bozo"

Answer the following questions looking at Level 1 plans.

(d) What query plans are produced for this query?

(e) What would be the cost of executing the query?

13. In the previous question, the queries we used selected the entire tuple from the relation. However, if our *select* clause projected only those fields which are already a part of the index, then the query does not need to access the datapages of the relation. This question gets you to explore how the query plans and their costs change when you specify index-only queries.

Using Catalog D, type in the following query.

select * from emp1
where ename < "Brian"

Answer the following questions looking at Level 1 plans.

(a) What query plans are produced for this query?

(b) What would be the cost of executing the query?

Using Catalog D, type in the following query.

select ename from emp1
where ename < "Brian"

Answer the following questions looking at Level 1 plans.

(c) What query plans are produced for this query?

(d) What would be the cost of executing the query?

14. This question gets you to explore the difference between queries that require sorting or not.

Using Catalog D, type in the following query.

select title, count(*) from emp1
group by title

Answer the following questions, looking at level 1 plans.

(a) Is the query plan index-only? What indexes are used?

(b) Is sorting required?

Using Catalog D, type in the following query.

select title, count(*) from emp3
group by title

Answer the following questions, looking at level 1 plans.

(c) Is the query plan index-only? What indexes are used?

(d) Is sorting required?

6 Where to Find Makefiles, Code, etc.

Copy all the files from ~cs564-1/fall96/proj6 into your own local directory. The files are:

- *README*: A description to show you how to set up the environment variables for using minibaseview.
- *minibaseview*: A script to start the query optimizer viewer.
- *catalog-format.txt*: A description of the format of the catalogs.
- *catalogA, catalogB, catalogD*: Catalog files.

7 What to Turn In, and When

You should turn in:

1. Brief answers to the questions in a separate file. Use any word processing software that you want, but please make the file easily readable. (Plain text is preferable.)

The assignment is due at 5PM on Dec 10 1996. The solution will be made public then, and solutions turned in after that time will not receive any credit. So be sure to turn in whatever you have, even if it is not working fully, at that time.

I emphasize that late submissions will not receive any credit. Computers—and life!—being what they are, expect everything to take longer than you expect, even taking this expectation into account. So start early, and plan on getting things done well before the due date. Nothing short of a nuclear explosion (in the CS building, not the South Pacific) constitutes a valid reason for an extension.