CSE530S

Studio: Optimize it!

Introduction

In this studio you will:

Learn how to examine execution plans

Create indexes on your database

Get practice using heuristics for optimization

You are encouraged to work in groups of up to 4 people. Please do not just let one person do all of the work while everyone else watches. It is important for everyone to follow these steps and participate in the studio.

Execution Plans

To understand how the server is optimizing queries, we need to look at the result of the query parser and optimizer. The server provides something called an execution plan that allows us to view this information. To see the execution plan for a query, first write the query that you are interested in analyzing. Then put the EXPLAIN keyword in front of that query. An example might look like this:

```
EXPLAIN
SELECT *
FROM Students;
```

The EXPLAIN command outputs the execution plan in table format. For more details on what this report contains, take a look at the documentation.

Using EXPLAIN, along with some sample queries, answer the following questions:

* What can the execution plan tell us about WHERE clauses?

* What can the exeuction plan tell us about joins?

* What can the execution plan tell us about indexes?

Applying indexes

Indexes are a quick and easy way to improve the performance of a database. Indexes are applied to a column (or set of columns) and this is usually done after table creation.

First, examine the design of your course tracking database and decide which columns should be indexed. Common columns for indexing include:

* Foreign key columns

* Columns that you expect to be frequently included in queries

For each index that you create, state what type of index it will be (primary, clustered, secondary).

Before creating an index, write a simple query that includes a join, and run it on your database. Examine the execution plan and the amount of time it takes the query to run.

Now, create some indexes on your database. You can find the documentation for creating indexes here.

After you have created your indexes, run your original query again. Did the execution plan change at all? Why or why not? Was the performance improved? Why or why not?

Time permitting, construct the B+ tree for one of the tables that you created an index for. Feel free to make up additional data points if necessary.

Query Optimization

In class we discussed some heuristics that can be used to improve query performance. Below you will find a list of queries that you were asked to write in a previous studio. For each query, create an unoptimized query tree, then optimize the query tree using the heuristics.

* Write a query that well retrieve a list of all students currently enrolled in the school. This list should only contain their first and last names.

* Write a query that will show which courses are offered by which departments. If you do not have a Department table, you should incorporate it into your design and add it to your database, making sure to include the appropriate foreign keys. The result of this query should have one row for each course that contains the name of the department and the name of the course offered by the department.

* Select one course that exists in your database, and write a query that will show all of the students who have taken or are currently taking that course. This will likely require a join of two or more tables.

* Select one student that exists in your database, and write a query that will show what courses they have taken.

* Write a query that lists the number of students that are enrolled in each course.