**COMP90050: Advanced Database Systems**

**Semester 2, 2024**

**Lab 1 Solutions**

1. Run the following query to get all distinct titles in use:

select distinct(title) from employees.titles;

Then run the following for each title identified in the query above:

select count(title) from employees.titles where title="Engineer";

This would return the total number of employees per title filtered using a WHERE clause.

1. Execute a select query like the following without an index on the “*title*” column:

select count(title) from employees.titles where title="Engineer";

This would return an execution time of around 0.105 sec (can be different depending on the underlying device).

Try for other titles in the same manner.

1. You can create an index by typing the query:

create index title\_idx on employees.titles (title);

Then run the same select queries on each employee title as executed in Ans 2. You will observe a decrease in execution time for each query (some in the magnitude of 30-40ms potentially depending on the device).

1. In the example schema given above: “salary”, “first name”, and “last name” might be good candidates for BTree and Hash Indexes. The column dept\_name might have been a good candidate for creating an index if the cardinality for that table was “Many to Many”.
2. One of the primary issues with creating indexes is that it takes more disc space. Similarly, operations like UPDATE, INSERT, and DELETE are slowed down relatively but the UPDATE operation can be sped up using a WHERE clause that uses an indexed field.
3. We can run the EXPLAIN ANALYZE command as so:

EXPLAIN ANALYZE select distinct(dept\_name) from employees.departments;