The query that we used were of the form

```
select u.url from word w, word_to_url t, url u where w.word = <insert word here> and
w.wid = t.wid and t.urlid = u.urlid;.
```

Before creating an index, the query called with 'computer', 'student', 'admissions', and 'carleton' all took about 15000ms. Calling explain on all of these queries revealed that they were all using the same evaluation plan, which was to do a nested hash join. Within this, there were several sequential scans one of which was on the word_to_url relation which contained more than 160 billion tuples.

To remedy this problem, an index on wid attribute was created on word_to_url with the command

```
create index wid_idx_wtu on word_to_url (wid);.
```

This took a whopping 227217ms. After this was created, the same four queries were run. The results are as follows:

w.word = 'computer' : 92ms
w.word = 'student' : 850ms
w.word = 'admissions' : 373ms
w.word= 'carleton' : 1101ms

We can see that there was at least an order of magnitude improvement on the running time of these queries, and analyzing the execution plan reveals why. While a hash join is being done each time, the inner hash join is replaced with a nested loop that utilizes index scanning instead of the massive sequential scanning.