Reverse Engineering Zotero Citation Management Database in PostgreSQL

Katherine L. Polley

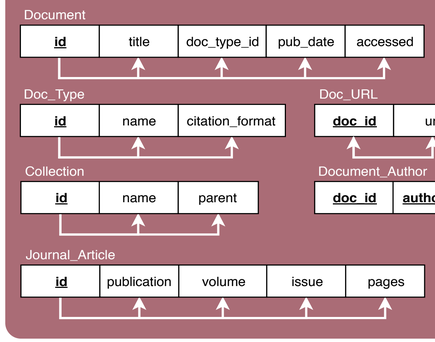
COMP 37500: Database Systems

Relational Algebra

The database was designed such that all of the existing functional dependencies fit the criteria that must be met for a relation to be in third normal form:

- The entire primary key (and only that key) functionally determines all other attributes in the table. In some cases, there are no other attributes beside the key.
- If a non-prime attribute functionally determines any other attribute, that other attribute is the primary key.

Below is a sample of the functional dependencies:

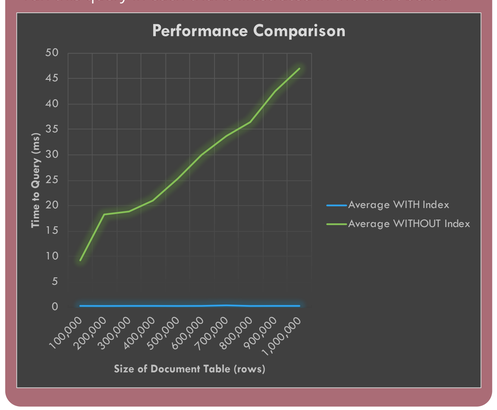


Performance Test: Indexing

Database efficiency was tested using a common Zotero query: finding the URL of a document given its title.

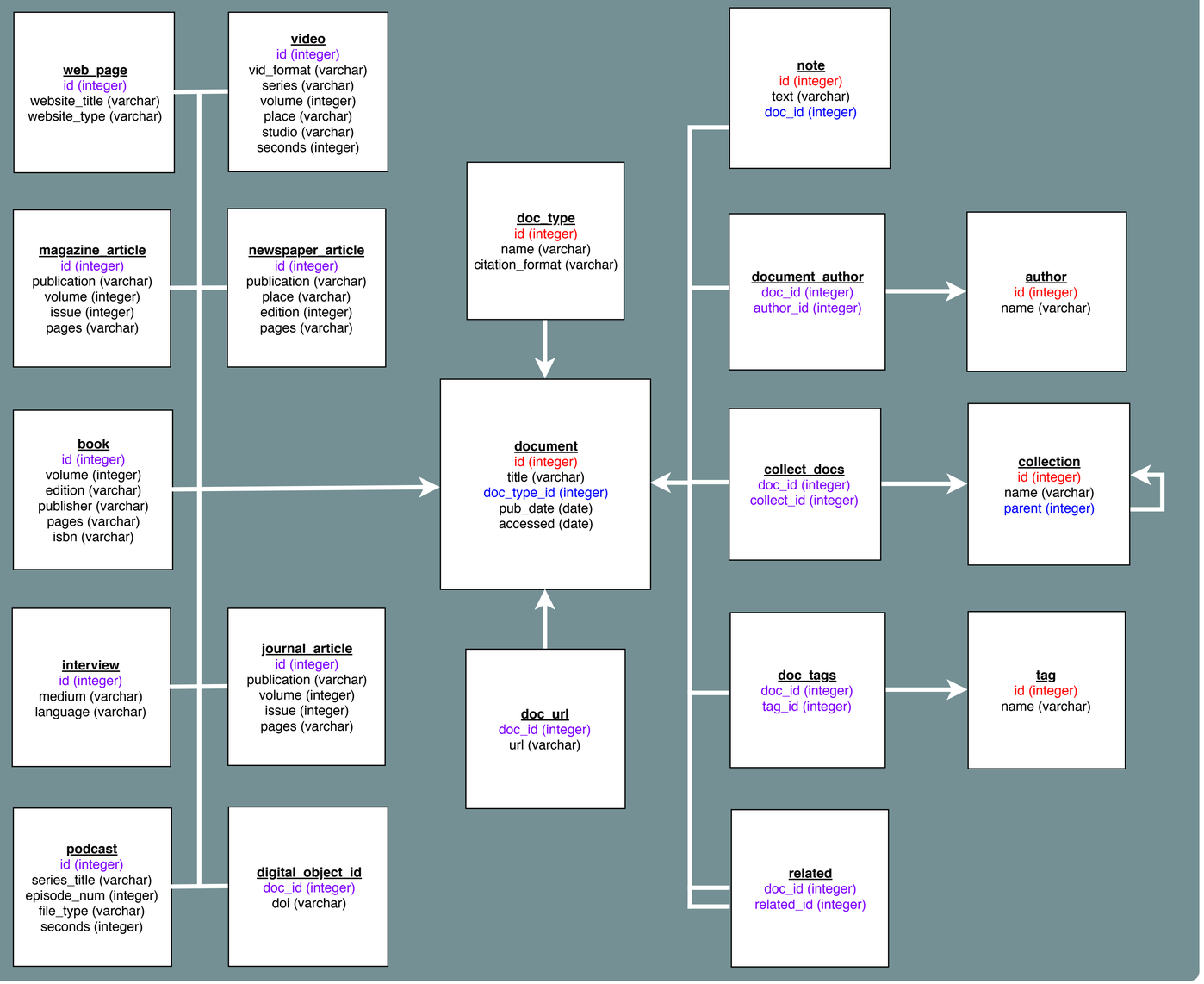
```
SELECT url FROM document
    JOIN doc_url ON document.id = doc_url.doc_id
    WHERE document.title = '<title>';
```

At each size, the query was run 50 times with an index on the title column and 50 times without. The average time to run this query in each trial is illustrated in the chart below.



Schema

This database schema recreates the most important tables that Zotero would require, centering around the document table, which would contain all the sources managed by Zotero, and which is referenced extensively by other tables that contain relevant information about each document. The tables were split up to meet the requirements of third normal form (3NF). Attributes in red are primary keys, blue are foreign keys, and purple function as both a foreign key and a primary key. The arrows indicate which tables are referenced by the foreign keys.



Queries

Below are a selection of queries that could be employed to retrieve useful information from the Zotero database.

- Find all documents related to a document given by title.

```
SELECT new.title FROM document AS og
    JOIN related ON og.id = related.doc_1
    JOIN document AS new ON related.doc_2 = new.id
    WHERE og.title = 'All About Databases'
UNION
SELECT new.title FROM document AS og
    JOIN related ON og.id = related.doc_2
    JOIN document AS new ON related.doc_1 = new.id
    WHERE og.title = 'All About Databases';
```

- Find all documents with a given tag *t*.

```
SELECT document.title FROM document
    JOIN doc_tags ON document.id = doc_tags.doc_id
    JOIN tag ON doc_tags.tag_id = tag.id
    WHERE tag.name = 'PostgreSQL';
```

- Find all sub-collections of a collection given by name.

```
SELECT child.name FROM collection AS child
    JOIN collection AS parent ON
        child.parent_collection = parent.id
    WHERE parent.name = 'COMP 375';
```

- Find all documents that are not filed into any collection.

```
SELECT title FROM document
    LEFT JOIN collect_docs
            ON document.id = collect_docs.doc_id
    WHERE collect_id IS NULL;
```

- Get all information about a document given the title.
  This requires coding outside of SQL; the following the relevant excerpt from a Python script that gets the results.

```
...
cursor.execute("
SELECT document.id, name FROM doc_type JOIN
    document ON document.doc_type_id = doc_type.id
    WHERE document.title = 'All About Databases'
")
results = cursor.fetchone()
id = results[0]
type = results[1]

cursor.execute("
SELECT * FROM document JOIN " + type + " USING(id)
    WHERE id = " + str(id)
)
print(cursor.fetchall())
...
```

Acknowledgements

I would like to thank Professor Ali Erkan for his support and guidance on this final project and throughout the course.