Database Implementation

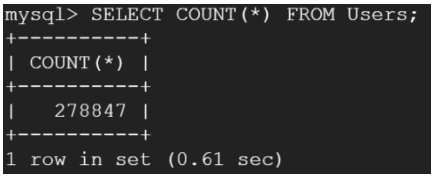
We used the following DDL commands to implement our database:

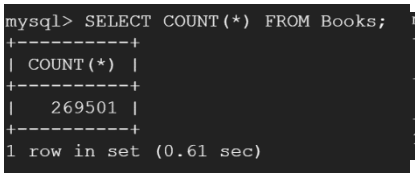
```
-- Load Publishers by creating unique IDs for each publisher
CREATE TABLE Publishers (
  PublisherId INT PRIMARY KEY,
  PublisherName VARCHAR(8192),
);

CREATE TABLE Books (
    -- ISBN10 is always 10 characters long and it's unique
    ISBN            CHAR(10)        PRIMARY KEY,
    Title           VARCHAR(4096),
    Author          VARCHAR(255),
    YearPublished   INT,
    PublisherId     INT,

    FOREIGN KEY (PublisherId) REFERENCES Publisher (PublisherId)

    SELECT * FROM Books_RAW NATURAL JOIN Publishers
);

-- Load the Users dataset
CREATE TABLE Users (
    Username VARCHAR(32) PRIMARY KEY,
    DisplayName VARCHAR(32),
    PasswordHash CHAR(64) -- We use SHA512 so hashes are 512 bits = 64 B
);

-- Load the Ratings dataset
CREATE TABLE Ratings (
    Username    VARCHAR(32) ,
    ISBN        CHAR(10),
    Rating      INT,
    Description VARCHAR(255)

    FOREIGN KEY (Username)  REFERENCES Users (Username),
    FOREIGN KEY (ISBN)      REFERENCES Books (ISBN),
    PRIMARY KEY (Username, ISBN),

    -- The rating must be valid
    CHECK(Rating >= 0 AND Rating <= 10)
);
```

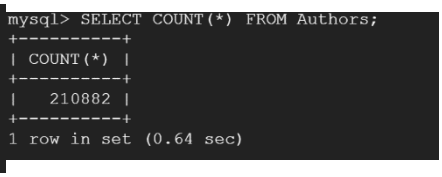
```sql
CREATE TABLE Authors (
    Name VARCHAR(255) PRIMARY KEY,
    Popularity INT
);

CREATE TABLE Friends (
    WantsRecs VARCHAR(32),
    GivesRecs VARCHAR(32),

    FOREIGN KEY (WantsRecs) REFERENCES Users (Username),
    FOREIGN KEY (GivesRecs) REFERENCES Users (Username)
);
```

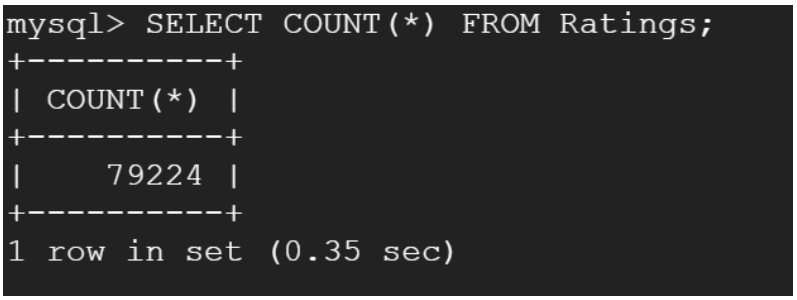
Note that since the PublisherName can be quite large (up to **8192 B = 8 KB**), we made a separate Publishers table to avoid storing it too many times.

Here is a screenshot of the main tables. These tables were primarily implemented on GCP, but some testing was also done locally.









Advanced Queries

All code for the queries can be found at the following link: https://github.com/tommasobassetto/cs411-book-data-loader/releases/tag/stage.3

We implemented the following queries:

- RecommendFromAuthor
- RecommendFromPublisher
- RecommendFromFriends

These queries take as input the minimum rating for a “good” book, and the user to recommend for. The output is a table with new books that meet the following criteria:

1. The author/publisher is the same as that of a “good” book, or one of your friends considered the book a “good” book.
2. You have not already read this book. (Set operations were used to enforce this, as well as subqueries).

Note that we also joined on the Publishers table to get the PublisherName for all recommendation queries. All queries also return a score to order recommendations. For Author/Publisher, this is always 1, but for RecommendFromFriends it’s the sum of all of your friends’ rating of that book. (The Friends query doesn’t add a friend’s rating if they considered it a bad book.)

Screenshot: (Note that since Users only rarely review books, we use user 10030 for all the queries, as they have reviewed a large number of books. This prevents query output from being the empty set.)

```
mysql> CALL RecommendFromAuthor("10030", 7);
Query OK, 48 rows affected (0.52 sec)

mysql> SELECT Title, Author, PublisherName FROM AuthorRatings ORDER BY Score DESC LIMIT 15;
+-----------------------------------------------------------------------------------------------------------------+---------------+----------------------+
| Title                                                                                                           | Author        | PublisherName        |
+-----------------------------------------------------------------------------------------------------------------+---------------+----------------------+
| Chicken Soup for the Soul (Chicken Soup for the Soul Series  (Cloth))                                           | Jack Canfield | Health Communications |
| A 2nd Helping of Chicken Soup for the Soul (Chicken Soup for the Soul Series (Paper))                           | Jack Canfield | Health Communications |
| A 2nd Helping of Chicken Soup for the Soul: 101 More Stories to Open the Heart and Rekindle the Spirit          | Jack Canfield | HCI                   |
| A 3rd Serving of Chicken Soup for the Soul (Chicken Soup for the Soul Series (Paper))                           | Jack Canfield | Health Communications |
| Chicken Soup for the Surviving Soul: 101 Healing Stories to Comfort Cancer Patients and Their Loved Ones        | Jack Canfield | Health Communications |
| Condensed Chicken Soup for the Soul (Chicken Soup for the Soul Series (Paper))                                  | Jack Canfield | Health Communications |
| Chicken Soup for the Woman's Soul (Chicken Soup for the Soul Series (Paper))                                    | Jack Canfield | Health Communications |
| Chicken Soup for the Soul at Work (Chicken Soup for the Soul Series (Paper))                                    | Jack Canfield | Health Communications |
| Chicken Soup for the Woman's Soul (Chicken Soup for the Soul Series (Cloth))                                    | Jack Canfield | Health Communications |
| A 4th Course of Chicken Soup for the Soul: 101 More Stories to Open the Heart and Rekindle the Spirit           | Jack Canfield | Health Communications |
| Chicken Soup for the Mother's Soul (Chicken Soup for the Soul Series (Paper))                                   | Jack Canfield | Health Communications |
| Chicken Soup for the Teenage Soul (Chicken Soup for the Soul)                                                   | Jack Canfield | Health Communications |
| A 4th Course of Chicken Soup for the Soul (Chicken Soup for the Soul)                                           | Jack Canfield | Health Communications |
| Chicken Soup for the Teenage Soul (Chicken Soup for the Soul)                                                   | Jack Canfield | HCI Teens             |
| Chicken Soup for the Christian Soul (Chicken Soup for the Soul Series (Paper))                                  | Jack Canfield | Health Communications |
+-----------------------------------------------------------------------------------------------------------------+---------------+----------------------+
15 rows in set (0.00 sec)
```

We also implemented RecommendFromSimilar, which takes as input the minimum rating for a “good” book, the minimum number of common books for a “similar” user, and the user to recommend for.

This query returns the books that meet the following criteria:

1. At least one “similar” user read the book and rated it “good”.

Screenshot:

```
mysql> CALL RecommendFromSimilar("10030", 7, 1);
Query OK, 13 rows affected (0.31 sec)

mysql> SELECT Title, Author, PublisherName FROM SimilarRatings ORDER BY Score DESC LIMIT 15;
+-----------------------------------------------------------------------------------------------------------------+-------------------------+-----------------------------+
| Title                                                                                                           | Author                  | PublisherName               |
+-----------------------------------------------------------------------------------------------------------------+-------------------------+-----------------------------+
| Politically Correct Bedtime Stories: Modern Tales for Our Life and Times                                        | James Finn Garner       | John Wiley &amp;amp; Sons Inc |
| Of Mice and Men/Cannery Row (2 Books in 1)                                                                      | John Steinbeck          | Penguin USA                 |
| Sybil Leek's Book of the curious and the occult                                                                 | Sybil Leek              | Ballantine Books            |
| Lincoln                                                                                                         | Gore Vidal              | Ballantine Books            |
| Degree of Guilt                                                                                                 | Richard North Patterson | Ballantine Books            |
| The Firm                                                                                                        | John Grisham            | Bantam Dell Publishing Group |
| Died in the Wool                                                                                                | Ngaio Marsh             | Jove Books                  |
| Prime Witness                                                                                                   | Steven Paul Martini     | Jove Books                  |
| Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))                                                | J. K. Rowling           | Arthur A. Levine Books      |
| CYBERPUNK: OUTLAWS AND HACKERS ON THE COMPUTER FRONTIER                                                         | Katie Hafner            | Touchstone                  |
| All I Really Need to Know                                                                                       | ROBERT FULGHUM          | Ivy Books                   |
| Chicken Soup for the Soul (Chicken Soup for the Soul)                                                           | Jack Canfield           | Health Communications       |
| Chicken Soup for the Mother's Soul (Chicken Soup for the Soul (Hardcover Health Communications))                | Jack Canfield           | Health Communications       |
+-----------------------------------------------------------------------------------------------------------------+-------------------------+-----------------------------+
13 rows in set (0.01 sec)
```

The final procedure is RecommendFromAll. This takes the recommendations from all of the previous methods and sets the score to be equal to that author's popularity (or NULL if not found). This operation uses multiple INSERT INTO to combine tables followed by a LEFT OUTER JOIN with the Authors table to get the popularity.

```
mysql> CALL RecommendFromAll("10030", 7, 1);
Query OK, 109 rows affected (1.84 sec)

mysql> SELECT Title, Author, PublisherName, Score FROM CombinedRatings ORDER BY Score DESC LIMIT 15;
+-----------------------------------------------------------+--------------------+-----------------------------+-------+
| Title                                                     | Author             | PublisherName               | Score |
+-----------------------------------------------------------+--------------------+-----------------------------+-------+
| The Tangle Box: A Magic Kingdom of Landover Novel         | Terry Brooks       | Ballantine Books            | 76907 |
| The Firm                                                  | John Grisham       | Bantam Dell Publishing Group | 40334 |
| The Tale of the Body Thief (Vampire Chronicles (Paperback)) | Anne Rice        | Ballantine Books            | 21069 |
| Of Mice and Men/Cannery Row (2 Books in 1)                | John Steinbeck     | Penguin USA                 | 16766 |
| The Great Train Robbery                                   | Michael Crichton   | Ballantine Books            | 13127 |
| The Lost World                                            | Michael Crichton   | Ballantine Books            | 13127 |
| The World According to Garp                               | John Irving        | Ballantine Books            | 11881 |
| The 158-Pound Marriage                                    | John Irving        | Ballantine Books            | 11881 |
| The Hotel New Hampshire                                   | John Irving        | Ballantine Books            | 11881 |
| The Hotel New Hampshire (Ballantine Reader's Circle)      | John Irving        | Ballantine Books            | 11881 |
| Taft                                                      | Ann Patchett       | Ballantine Books            |  9883 |
| Six Wives of Henry VIII                                   | Alison Weir        | Ballantine Books            |  5436 |
| Flesh and Blood                                           | Jonathan Kellerman | Ballantine Books            |  3922 |
| Seasons of Her Life                                       | Fern Michaels      | Ballantine Books            |  3729 |
| Serendipity                                               | Fern Michaels      | Ballantine Books            |  3729 |
+-----------------------------------------------------------+--------------------+-----------------------------+-------+
15 rows in set (0.00 sec)
```

Indexing Analysis

Since the access ISBN, Username, and Rating most often in our procedures (usually in the Ratings table), we tried setting an index for each of these columns inside the Ratings table. Since we were unable to run EXPLAIN ANALYZE on a CALL statement directly, we took the part of the stored procedure where we believed an index would make the most difference and ran EXPLAIN ANALYZE on that single query.

EXPLAIN ANALYZE No. 1 – Index on ISBN in Ratings

Command:

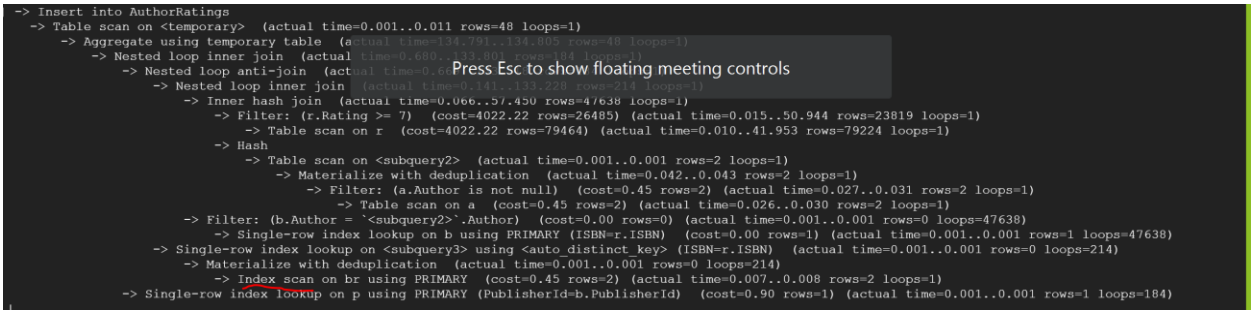
```
mysql> EXPLAIN ANALYZE     INSERT INTO AuthorRatings (
    ->         SELECT ISBN, MIN(b.Title) AS Title, MIN(b.Author) AS Author, MIN(PublisherName) AS PublisherName, 1 as Score
    ->         FROM (Books b NATURAL JOIN Ratings r NATURAL JOIN Publishers p)
    ->         WHERE (r.Rating >= 7) AND EXISTS(
    ->             SELECT *
    ->             FROM Authors_Proc a
    ->             WHERE a.Author = b.Author
    ->         ) AND NOT EXISTS(
    ->             SELECT ISBN
    ->             FROM UserBooksRead br
    ->             WHERE br.ISBN = b.ISBN
    ->         )
    ->         GROUP BY ISBN
    ->         ORDER BY Score DESC
    ->     );
```

Run without index:

```
| -> Insert into AuthorRatings
    -> Table scan on <temporary>  (actual time=0.002..0.011 rows=48 loops=1)
        -> Aggregate using temporary table  (actual time=140.363..140.377 rows=48 loops=1)
            -> Nested loop inner join  (actual time=0.611..139.369 rows=184 loops=1)
                -> Nested loop anti-join  (actual time=0.595..138.820 rows=214 loops=1)
                    -> Nested loop inner join  (actual time=0.093..138.820 rows=214 loops=1)
                        -> Inner hash join  (actual time=0.048..58.520 rows=47638 loops=1)
                            -> Filter: (r.Rating >= 7)  (cost=4022.22 rows=26485) (actual time=0.013..52.161 rows=23819 loops=1)
                                -> Table scan on r  (cost=4022.22 rows=79464) (actual time=0.008..42.794 rows=79224 loops=1)
                            -> Hash
                                -> Table scan on <subquery2>  (actual time=0.001..0.001 rows=2 loops=1)
                                    -> Materialize with deduplication  (actual time=0.028..0.028 rows=2 loops=1)
                                        -> Filter: (a.Author is not null)  (cost=0.45 rows=2) (actual time=0.016..0.019 rows=2 loops=1)
                                            -> Table scan on a  (cost=0.45 rows=2) (actual time=0.014..0.017 rows=2 loops=1)
                        -> Filter: (b.Author = `<subquery2>`.Author)  (cost=0.00 rows=0) (actual time=0.002..0.002 rows=0 loops=47638)
                            -> Single-row index lookup on b using PRIMARY (ISBN=r.ISBN)  (cost=0.00 rows=1) (actual time=0.001..0.001 rows=1 loops=47638)
                    -> Single-row index lookup on <subquery3> using <auto_distinct_key> (ISBN=r.ISBN)  (actual time=0.001..0.001 rows=0 loops=214)
                        -> Materialize with deduplication  (actual time=0.001..0.001 rows=0 loops=214)
                            -> Index scan on br using PRIMARY  (cost=0.45 rows=2) (actual time=0.004..0.006 rows=2 loops=1)
                -> Single-row index lookup on p using PRIMARY (PublisherId=b.PublisherId)  (cost=0.90 rows=1) (actual time=0.001..0.001 rows=1 loops=184)
```

Cost = 0.15s

Run with Index:



Cost = 0.13s.

While this is a small speedup over no index, there are two things to consider. First, creating the index itself takes about half a second, so this would only be worth it over a very large number of queries. Second, a speedup of 0.02s on this query is too small to determine if the index scan saved any time when running the query. Therefore, we decided to not use the index.

EXPLAIN ANALYZE No. 2 – Index on Username in Ratings

Command:

```
mysql> EXPLAIN ANALYZE     INSERT INTO UserBooksRead (
    ->           SELECT ISBN
    ->           FROM Ratings
    ->           WHERE Username = "10030"
    ->      );
```

Without indexing (ISBN is part of the table's primary key):

```
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                                                                                                                                     |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Insert into UserBooksRead
  -> Index lookup on Ratings using PRIMARY (Username='10030')  (cost=1.67 rows=13) (actual time=0.013..0.023 rows=13 loops=1)
 |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

With indexing:

```
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                                                                                                                                     |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| -> Insert into UserBooksRead
  -> Index lookup on Ratings using PRIMARY (Username='10030')  (cost=1.67 rows=13) (actual time=0.011..0.021 rows=13 loops=1)
 |
+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

This request already takes 0.00s and we don’t use Username in any other significant WHERE, GROUP BY, or HAVING clauses, so we decided not to use this index.

EXPLAIN ANALYZE No. 3 – Index on Rating in Ratings

Command:

```
INSERT INTO UserBooksRead (
    SELECT ISBN
    FROM Ratings
    WHERE Username = "10030" AND rating > 5
);
```

Without Index:

```
mysql> EXPLAIN ANALYZE     INSERT INTO UserBooksRead (
    ->         SELECT ISBN
    ->         FROM Ratings
    ->         WHERE Username = "10030" AND rating > 5
    ->     );
+----------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------+
| EXPLAIN
                                                                                                        |
+----------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------+
| -> Insert into UserBooksRead
    -> Filter: (Ratings.Rating > 5)  (cost=1.44 rows=4) (actual time=0.025..0.037 rows=3 loops=1)
        -> Index lookup on Ratings using PRIMARY (Username='10030')  (cost=1.44 rows=13) (actual time=0.022..0.033 rows=13 loops=1)
 |
+----------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

With Index:

```
mysql> EXPLAIN ANALYZE     INSERT INTO UserBooksRead (
    ->         SELECT ISBN
    ->         FROM Ratings
    ->         WHERE Username = "10030" AND rating > 5
    ->     );
+----------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------+
| EXPLAIN
                                                                                                        |
+----------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------+
| -> Insert into UserBooksRead
    -> Filter: (Ratings.Rating > 5)  (cost=0.90 rows=6) (actual time=0.023..0.036 rows=3 loops=1)
        -> Index lookup on Ratings using PRIMARY (Username='10030')  (cost=0.90 rows=13) (actual time=0.021..0.032 rows=13 loops=1)
 |
+----------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

Once again, the portion of the stored procedure where we used the rating was too short (in terms of execution time) for the index to have any meaningful impact on performance.

Summary

Overall, we found that the queries already take so little time to run on our dataset, that any potential improvement from using an index is indistinguishable from a slight hardware speedup. Therefore, we decided not to use any indices for our project.