Rylie Byers
U1483890
June 13, 2024

Lab 4: SQL

Part 3 - Simple Retrieval Queries

Connect to the Library database, and create queries to find the specified information below. Some of these are quite simple, and we have already seen them. Note that the Library has some additional information in it to make the queries more interesting. The slides shown in class do not contain the full instances.

1. Get the ISBNs of all books by <Author>

SELECT ISBN FROM Titles WHERE Author = <'Author'>;

2. Get Serial numbers (descending order) of all books by <ISBN>

SELECT Serial FROM Inventory WHERE ISBN = '<ISBN>' ORDER BY Serial DESC;

3. Get the Serial numbers and ISBNs of all books checked out by <Patron's name>

SELECT s.Serial, i.ISBN FROM CheckedOut s Join Patrons p on s.CardNum = p.CardNum JOIN Inventory i ON s.Serial = i.Serial WHERE p.Name = <'Name'>;

4. Get phone number(s) and Name of anyone with <ISBN> checked out

SELECT p.Name, ph.Phone FROM CheckedOut c JOIN Patrons p ON c.CardNum = p.CardNum JOIN Inventory b ON c.Serial = b.Serial JOIN Phones ph ON p.CardNum = ph.CardNum WHERE b.ISBN = <'ISBN'>;

When testing out your queries, you can replace the variables such as <Author> with an author that you know is in the database, such as "Herbert". However, when you hand in your solution, put the placeholder back into the query.

Part 4 - Intermediate Retrieval Queries

1. Find the Authors of the library's oldest <N> books. Assume the lowest serial number is the oldest book.

SELECT t.Author FROM Titles t JOIN Inventory i ON t.ISBN = i.ISBN ORDER BY i.Serial ASC LIMIT <N>;

2. Find the name and phone number of the person who has checked out the most recent book. Assume higher serial numbers are newer. **Note that this query is not concerned with the absolute highest serial number, it is concerned with the highest one that has been checked out.**

SELECT p.Name, ph.Phone FROM CheckedOut c JOIN Patrons p ON c.CardNum = p.CardNum JOIN Phones ph ON p.CardNum = ph.CardNum ORDER BY c.Serial DESC LIMIT 1;

3. Find the phone number(s) and name of anyone who has checked out any book.

SELECT DISTINCT p.Name, ph.Phone FROM CheckedOut c JOIN Patrons p ON c.CardNum = p.CardNum JOIN Phones ph ON p.CardNum = ph.CardNum;

4. Find the Authors and Titles of the books who have NOT been checked out by anyone. The query should not return duplicates.

SELECT DISTINCT t.Author, t.Title FROM Titles t JOIN Inventory i ON t.ISBN = i.ISBN LEFT JOIN CheckedOut c ON i.Serial = c.Serial WHERE c.Serial IS NULL;

Part 5 - Chess Queries

For this part, switch to the Chess database.
Be careful with your queries in this database This one has a lot of data, and a poorly formed query could easily overwhelm the server. If you repeatedly run very expensive queries, your account will be locked.
Note that the schemas in this database have some additional columns that were not shown in Lab 3, and the Result column in the Games table is encoded as 'W', 'B', or 'D', for white winning, black winning, or draw. Also note that names are given as "last, first", so Magnus Carlsen is listed as "Carlsen, Magnus". Feel free to run "describe" commands on the tables in the Chess database.
Provide SQL for the queries below:

1. Find the names and IDs of any player with the 10 highest Elo ratings.

SELECT Name, pID, Elo FROM Players ORDER BY Elo DESC LIMIT 10;

2. Find the names and Elo ratings of any player who has ever played a game as black.

SELECT DISTINCT p.Name, p.Elo FROM Players p JOIN Games g ON p.pID = g.BlackPlayer;

3. Find the names of any player who has ever won a game as white.

SELECT DISTINCT p.Name FROM Players p JOIN Games g ON p.pID = g.WhitePlayer WHERE g.Result = 'W';

4. Find the names of any player who played any games between 2014 and 2018 in Budapest HUN .

SELECT DISTINCT p.Name FROM Players p JOIN Games g ON p.pID IN (g.WhitePlayer, g.BlackPlayer) JOIN Events e ON g.eID = e.eID WHERE e.Site = 'Budapest HUN' AND e.Date BETWEEN '2014-01-01' AND '2018-12-31';

5. Find the Sites and dates of any event in which Garry Kasparov won a game.

SELECT DISTINCT e.Site, e.Date FROM Games g JOIN Events e ON g.eID = e.eID JOIN Players p ON g.WhitePlayer = p.pID WHERE p.Name = 'Kasparov, Garry' AND g.Result = 'W' UNION SELECT DISTINCT e.Site, e.Date FROM Games g JOIN Events e ON g.eID = e.eID JOIN Players p ON g.BlackPlayer = p.pID WHERE p.Name = 'Kasparov, Garry' AND g.Result = 'B';

6. Find the names of all opponents of Magnus Carlsen. An opponent is someone who he has played a game against. **Hint:** Both Magnus and his opponents could play as white or black.

```
SELECT DISTINCT p.Name FROM Players p JOIN Games g ON p.pID IN (g.WhitePlayer,
g.BlackPlayer) JOIN Players mc ON (mc.pID = g.WhitePlayer OR mc.pID = g.BlackPlayer)
WHERE mc.Name = 'Carlsen, Magnus' AND p.Name != 'Carlsen, Magnus';
```

Killing Queries

If properly formed, these queries should all run in a few seconds, including the network transfer time. The computational parts should all run in a fraction of a second, assuming nobody else is overwhelming the server. To kill a query that is taking too long, the simplest approach is to press ctrl-c, which will disconnect you. A better approach is to open another mysql client in another terminal and run "show processlist;". Find the ID of your process, and run "kill query <id>;"

Hints

- Be careful with natural join - both Players and Events have a column called "Name", which have different meanings (but SQL doesn't know this). Trying to natural join any temporary relation with those two columns will not have the desired result, even if the relations do both have "eID". **To be safe, don't use natural join**.
- Be careful when mixing "and" and "or" in boolean conditions. It's always safest to be explicit with parenthesis. For example, (false and false or true) is not the same as (false and (false or true)). Getting this wrong can result in very expensive queries.