CS 61A Fall 2020

SQL

Discussion 12: November 18, 2020

1 Introduction

SQL is an example of a declarative programming language. Statements do not describe computations directly, but instead describe the desired result of some computation. It is the role of the query interpreter of the database system to plan and perform a computational process to produce such a result.

In SQL, data is organized into *tables*. A table has a fixed number of named **columns**. A **row** of the table represents a single data record and has one **value** for each column. For example, we have a table named `records` that stores information about the employees at a small company[1]. Each of the eight rows represents an employee.

`records`

Name	**Division**	**Title**	**Salary**	**Supervisor**
Ben Bitdiddle	Computer	Wizard	60000	Oliver Warbucks
Alyssa P Hacker	Computer	Programmer	40000	Ben Bitdiddle
Cy D Fect	Computer	Programmer	35000	Ben Bitdiddle
Lem E Tweakit	Computer	Technician	25000	Ben Bitdiddle
Louis Reasoner	Computer	Programmer Trainee	30000	Alyssa P Hacker
Oliver Warbucks	Administration	Big Wheel	150000	Oliver Warbucks
Eben Scrooge	Accounting	Chief Accountant	75000	Oliver Warbucks
Robert Cratchet	Accounting	Scrivener	18000	Eben Scrooge

For this discussion, you can test out your code at sql.cs61a.org. the `records` table should already be loaded in.

2 Creating Tables

We can use a `SELECT` statement to create tables. The following statement creates a table with a single row, with columns named "first" and "last":

```
sqlite> SELECT "Ben" AS first, "Bitdiddle" AS last;
Ben|Bitdiddle
```

Given two tables with the same number of columns, we can combine their rows into a larger table with `UNION`:

```
sqlite> SELECT "Ben" AS first, "Bitdiddle" AS last UNION
   ...> SELECT "Louis",        "Reasoner";
Ben|Bitdiddle
Louis|Reasoner
```

[1]Example adapted from Structure and Interpretation of Computer Programs

To save a table, use `CREATE TABLE` and a name. Here we're going to create the table of employees from the previous section and assign it to the name `records`:

```
sqlite> CREATE TABLE records AS
   ...>   SELECT "Ben Bitdiddle" AS name, "Computer" AS division,
   ...>     "Wizard" AS title, 60000 AS salary,
   ...>     "Oliver Warbucks" AS supervisor UNION
   ...>   SELECT "Alyssa P Hacker", "Computer",
   ...>     "Programmer", 40000, "Ben Bitdiddle" UNION ... ;
```

We can SELECT specific values from an existing table using a `FROM` clause. This query creates a table with two columns, with a row for each row in the `records` table:

```
sqlite> SELECT name, division FROM records;
Alyssa P Hacker|Computer
Ben Bitdiddle|Computer
Cy D Fect|Computer
Eben Scrooge|Accounting
Lem E Tweakit|Computer
Louis Reasoner|Computer
Oliver Warbucks|Administration
Robert Cratchet|Accounting
```

The special syntax `SELECT *` will select all columns from a table. It's an easy way to print the contents of a table.

```
sqlite> SELECT * FROM records;
Alyssa P Hacker|Computer|Programmer|40000|Ben Bitdiddle
Ben Bitdiddle|Computer|Wizard|60000|Oliver Warbucks
Cy D Fect|Computer|Programmer|35000|Ben Bitdiddle
Eben Scrooge|Accounting|Chief Accountant|75000|Oliver Warbucks
Lem E Tweakit|Computer|Technician|25000|Ben Bitdiddle
Louis Reasoner|Computer|Programmer Trainee|30000|Alyssa P Hacker
Oliver Warbucks|Administration|Big Wheel|150000|Oliver Warbucks
Robert Cratchet|Accounting|Scrivener|18000|Eben Scrooge
```

We can choose which columns to show in the first part of the `SELECT`, we can filter out rows using a `WHERE` clause, and sort the resulting rows with an `ORDER BY` clause. In general the syntax is:

```
SELECT [columns] FROM [tables]
  WHERE [condition] ORDER BY [criteria];
```

For instance, the following statement lists all information about employees with the "Programmer" title.

```
sqlite> SELECT * FROM records WHERE title = "Programmer";
Alyssa P Hacker|Computer|Programmer|40000|Ben Bitdiddle
Cy D Fect|Computer|Programmer|35000|Ben Bitdiddle
```

The following statement lists the names and salaries of each employee under the accounting division, sorted in **descending** order by their salaries.

```
sqlite> SELECT name, salary FROM records
   ...>   WHERE division = "Accounting" ORDER BY -salary;
Eben Scrooge|75000
Robert Cratchet|18000
```

Note that all valid SQL statements must be terminated by a semicolon (`;`). Additionally, you can split up your statement over many lines and add as much whitespace as you want, much like Scheme. But keep in mind that having consistent indentation and line breaking does make your code a lot more readable to others (and your future self)!

Questions

Our tables:

`records:`	**Name**	**Division**	**Title**	**Salary**	**Supervisor**

2.1 Write a query that outputs the names of employees that Oliver Warbucks directly supervises.

2.2 Write a query that outputs all information about employees that supervise themselves.

2.3 Write a query that outputs the names of all employees with salary greater than 50,000 in alphabetical order.

3 Joins

Suppose we have another table `meetings` which records the divisional meetings.

`meetings`

Division	**Day**	**Time**
Accounting	Monday	9am
Computer	Wednesday	4pm
Administration	Monday	11am
Administration	Wednesday	4pm

Data are combined by joining multiple tables together into one, a fundamental operation in database systems. There are many methods of joining, all closely related, but we will focus on just one method (the inner join) in this class.

When tables are joined, the resulting table contains a new row for each combination of rows in the input tables. If two tables are joined and the left table has m rows and the right table has n rows, then the joined table will have mn rows. Joins are expressed in SQL by separating table names by commas in the `FROM` clause of a `SELECT` statement.

```
sqlite> SELECT name, day FROM records, meetings;
Ben Bitdiddle | Monday
Ben Bitdiddle | Wednesday
...
Alyssa P Hacker | Monday
...
```

Tables may have overlapping column names, and so we need a method for disambiguating column names by table. A table may also be joined with itself, and so we need a method for disambiguating tables. To do so, SQL allows us to give aliases to tables within a `FROM` clause using the keyword `AS` and to refer to a column within a particular table using a dot expression. In the example below we find the name and title of Louis Reasoner's supervisor.

```
sqlite> SELECT b.name, b.title FROM records AS a, records AS b
   ...>   WHERE a.name = "Louis Reasoner" AND
   ...>         a.supervisor = b.name;
Alyssa P Hacker | Programmer
```

Questions

Our tables:

`records:`	**Name**	**Division**	**Title**	**Salary**	**Supervisor**
`meetings:`	**Division**	**Day**	**Time**		

3.1 Write a query that outputs the meeting days and times of all employees directly supervised by Oliver Warbucks.

3.2 Write a query that outputs the names of all pairs of employees that have a meeting at the same time. Make sure that if `A|B` appears in your output, `B|A` does not appear as well (`A|A` and `B|B` should additionally not appear).

3.3 (Extra question) Will the statement above filter out all redundant output in all cases? Why or why not?

3.4 Write a query that outputs the names of employees whose supervisor is in a different division.

4 Aggregation

So far, we have joined and manipulated individual rows using `SELECT` statements. But we can also perform aggregation operations over multiple rows with the same `SELECT` statements.

We can use the `MAX`, `MIN`, `COUNT`, and `SUM` functions to retrieve more information from our initial tables.

If we wanted to find the name and salary of the employee who makes the most money, we might say

```
sqlite> SELECT name, MAX(salary) FROM records;
Oliver Warbucks|150000
```

Using the special `COUNT(*)` syntax, we can count the number of rows in our table to see the number of employees at the company.

```
sqlite> SELECT COUNT(*) from RECORDS;
9
```

These commands can be performed on specific sets of rows in our table by using the `GROUP BY [column name]` clause. This clause takes all of the rows that have the same value in `column name` and groups them together.

We can find the miniumum salary earned in each division of the company.

```
sqlite> SELECT division, MIN(salary) FROM records GROUP BY division;
Computer|25000
Administration|25000
Accounting|18000
```

These groupings can be additionally filtered by the `HAVING` clause. In contrast to the `WHERE` clause, which filters out rows, the `HAVING` clause filters out entire groups.

To find all titles that are held by more than one person, we say

```
sqlite> SELECT title FROM records GROUP BY title HAVING count(*) > 1;
Programmer
```

Questions

Our tables:

`records`: **Name** **Division** **Title** **Salary** **Supervisor**

`meetings`: **Division** **Day** **Time**

4.1 Write a query that outputs each supervisor and the sum of salaries of all the employees they supervise.

4.2 Write a query that outputs the days of the week for which fewer than 5 employees have a meeting. You may assume no department has more than one meeting on a given day.

4.3 Write a query that outputs all divisions for which there is more than one employee, and all pairs of employees within that division that have a combined salary less than 100,000.

5 Tutorial

5.1 **Tutorial:** This short question is meant to help refresh your memory of topics covered in lecture and lab this week before tackling more challenging problems.

Table A has 5 rows and 4 columns, and Table B has 3 rows and 2 columns. If we join table A and table B, how many rows will the resulting table contain?

Use the following table called `courses` for the questions below:

`courses`

Professor	**Course**	**Semester**
John DeNero	CS 61C	Sp20
John DeNero	CS 61A	Fa19
Dan Garcia	CS 61C	Sp19
John DeNero	CS 61A	Fa18
Dan Garcia	CS 10	Fa18
Josh Hug	CS 61B	Sp18
John DeNero	CS 61A	Sp18
John DeNero	CS 61A	Fa17
Paul Hilfinger	CS 61A	Fa17
Paul Hilfinger	CS 61A	Sp17
John DeNero	Data 8	Sp17
Josh Hug	CS 61B	Sp17
Satish Rao	CS 70	Sp17
Nicholas Weaver	CS 61C	Sp17
Gerald Friedland	CS 61C	Sp17
⋮	⋮	⋮

5.1 Create a table called `num_taught` that contains three columns: `professor`, the `course` they taught, and the number of `times` they taught each course.

Hint: For this problem, it may help to `GROUP BY` multiple columns. Multiple columns and full expressions can appear in the group by clause, and groups will be formed for every unique combination of values that result.

5.2 Write a query that outputs two professors and a course if they have taught that course the same number of times. You may use the `num_taught` table you created in the previous question.

Note: This worksheet is a problem bank—most TAs will not cover all the problems in discussion section.

5.3 Write a query that outputs two professors if they co-taught (taught the same course at the same time) the same course more than once.