

## 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<sup>1</sup>. Each of the eight rows represents an employee.

<b>Name</b>	<b>Division</b>	<b>Title</b>	<b>Salary</b>	<b>Supervisor</b>
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

## 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
```

---

<sup>1</sup>Example adapted from Structure and Interpretation of Computer Programs

## 2 SQL

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 rows 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 self-supervising employees.
- 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	Thursday	1pm

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.

## 4 SQL

```
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 creates a table with columns: employee, salary, supervisor and supervisor's salary, containing all supervisors who earn more than twice as much as the employee.
  
- 3.2 Write a query that outputs the names of employees whose supervisor is in a different division.
  
- 3.3 Write a query that outputs the meeting days and times of all employees directly supervised by Oliver Warbucks.



## 4 Extra Questions

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

courses		
Professor	Course	Semester
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
John DeNero	CS 61A	Fa16
Paul Hilfinger	CS 61B	Fa16
⋮	⋮	⋮

There's also an additional table which we have constructed from `courses`, named `num_taught`. It contains the number of times each professor has taught a given course:

num_taught		
Professor	Course	Times
Gerald Friedland	CS 61C	1
John DeNero	CS 61A	2
John DeNero	Data 8	1
Josh Hug	CS 61B	1
Nicholas Weaver	CS 61C	1
Paul Hilfinger	CS 61A	2
Paul Hilfinger	CS 61B	1
Satish Rao	CS 70	1
⋮	⋮	⋮

- 4.1 Write a query that selects all courses Paul Hilfinger taught once.
  
- 4.2 Write a query that outputs all the rows from `courses` where the professor teaching that semester has taught the course at least one other time.
  
- 4.3 Write a query that outputs two professors and a course if they have taught that course the same number of times.