CS50 Section 7

Slides and Exercises at github.com/cjleggett/2022-section

Please scan your ID!

Think.

Pair.

Share.

Think. Pair. Share.

- Why do we use SQL?
- How do we design a SQL database?
- How do we start to make more complicated queries?

Outline

- Big Ideas (3:05)
- Designing a Database (3:10)
- Setting up a Database (3:25)
- Adding Data to a Database (3:35)
- Querying a Database (3:40)
- Editing a Database (3:50)
- Python and SQL (3:55)
- Exercises (4:00)
- Test Intro (4:10)

Big Ideas (3:05)

What is SQL?

- Structured Query Language
- A programming language used only to interact with databases
- Doesn’t look like any other programming language we’ve seen so far…

Why do we use SQL?

- We need a way for data to persist
- It’s fast!
- It’s standard

What are the Different Types of SQL?

- There are several different SQL **dialects**
- They all share the same logic, but have slightly different syntax and data types
- In this class, we'll generally use **SQLite**
- An advantage of **SQLite** is that we can store our entire database in one file called "something.db"

Designing a Database (3:10)

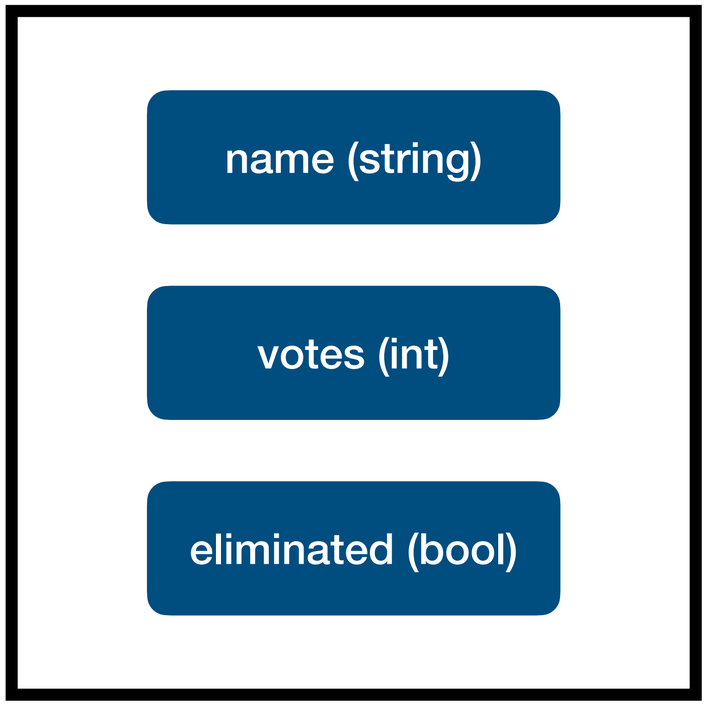
What’s a Database?

- An organized way of storing information
- Made up of **tables** that each store specific information
- **tables** can relate to each other in various ways

What’s a Table?

- Similar to a CSV or a spreadsheet
- Each row in a table is one distinct object
- Each column in a table is an attribute of the object and must have a data type

struct candidate



candidates

name (string)	votes (int)	eliminated (bool)
connor	3	TRUE
sophie	13	FALSE
sam	4	FALSE
isabelle	5	FALSE

candidates

id	name (string)	votes (int)	eliminated (bool)
1	connor	3	TRUE
2	sophie	13	FALSE
3	sam	4	FALSE
4	isabelle	5	FALSE

candidates

id	name (string)	votes (int)	eliminated (bool)
1	connor	3	TRUE
2	sophie	13	FALSE
3	sam	4	FALSE
4	isabelle	5	FALSE

Primary Key

candidates

id (INTEGER)	name (TEXT)	votes (INTEGER)	eliminated (INTEGER)
1	connor	3	TRUE
2	sophie	13	FALSE
3	sam	4	FALSE
4	isabelle	5	FALSE

How do we connect tables?

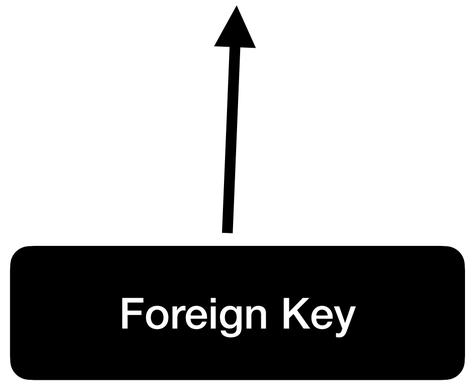
- We use the **primary key** of one table when referencing it in another table
- This is known as a **foreign key** when it is used in another table

voters

id (INTEGER)	name (TEXT)	candidate (INTEGER)
1	kalos	1
2	ben	2
3	amia	2

voters

id (INTEGER)	name (TEXT)	candidate (INTEGER)
1	kalos	1
2	ben	2
3	amia	2



candidates

id (INTEGER)	name (TEXT)	votes (INTEGER)	eliminated (INTEGER)
1	connor	3	TRUE
2	sophie	13	FALSE
3	sam	4	FALSE
4	isabelle	5	FALSE

voters

id (INTEGER)	name (TEXT)	candidate (INTEGER)
1	kalos	1
2	ben	2
3	amia	2

candidates

id (INTEGER)	name (TEXT)	votes (INTEGER)	eliminated (INTEGER)
1	connor	3	TRUE
2	sophie	13	FALSE
3	sam	4	FALSE
4	isabelle	5	FALSE

voters

id (INTEGER)	name (TEXT)	candidate (INTEGER)
1	kalos	1
2	ben	2
3	amia	2

Let’s try to make our own database!

- We’re working for the Harvard Registrar.
- They want a database to store data on who’s taking each class and who’s teaching each class.
- How should we get started?
- What table(s) should we have?
- What datatypes should go in our table(s)?

people

id (INTEGER) (Primary Key)	name (TEXT)	senior (INTEGER)

courses

id (INTEGER) (Primary Key)	title (TEXT)	code (TEXT)

students

student (INTEGER) (Foreign Key)	course (INTEGER) (Foreign Key)	score (REAL)

instructors

teacher (INTEGER) (Foreign Key)	course (INTEGER) (Foreign Key)

people

id (INTEGER)	name (TEXT)	senior (INTEGER)
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

courses

id (INTEGER)	title (TEXT)	code (TEXT)
1	Introduction to Computer Science	CS50
2	Principles of Economics (Microeconomics)	ECON10a
3	Abstraction and Design in Computer Science	CS51

students

student (INTEGER)	course (INTEGER)	score (REAL)
1	3	23.6
3	1	96.7

instructors

teacher (INTEGER)	course (INTEGER)
1	1
4	2

people

id (INTEGER)	name (TEXT)	senior (INTEGER)
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

courses

id (INTEGER)	title (TEXT)	code (TEXT)
1	Introduction to Computer Science	CS50
2	Principles of Economics (Microeconomics)	ECON10a
3	Abstraction and Design in Computer Science	CS51

students

student (INTEGER)	course (INTEGER)	score (REAL)
1	3	23.6
3	1	96.7

instructors

teacher (INTEGER)	course (INTEGER)
1	1
4	2

people

id (INTEGER)	name (TEXT)	senior (INTEGER)
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

courses

id (INTEGER)	title (TEXT)	code (TEXT)
1	Introduction to Computer Science	CS50
2	Principles of Economics (Microeconomics)	ECON10a
3	Abstraction and Design in Computer Science	CS51

students

student (INTEGER)	course (INTEGER)	score (REAL)
1	3	23.6
3	1	96.7

instructors

teacher (INTEGER)	course (INTEGER)
1	1
4	2

Setting Up (3:25)

How do we create a database?

- In SQLite, create a new file with the **.db** extension

How do we run queries?

- Type “sqlite3 **your_file.db**” in the terminal
  - You can quit this with CONTROL-Z
- **OR** Create a file with a **.sql** extension, and then in the terminal type “sqlite3 **your_file.db** < **your_sql_file.sql**”

Create a Table

```
CREATE TABLE table_name (
    column_name type,
    column_name type,
    …
);
```

Create a Table: people

```
CREATE TABLE people (
    id INTEGER,
    name TEXT,
    senior INTEGER
);
```

Create a Table: people

```
CREATE TABLE people (
    id INTEGER,
    name TEXT NOT NULL,
    senior INTEGER
);
```

Create a Table: people

```
CREATE TABLE people (
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    senior INTEGER
);
```

Create a Table: people

```
CREATE TABLE people (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    senior INTEGER
);
```

Create a Table: people

```
CREATE TABLE people (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    senior INTEGER
);
```

Create a Table: courses

```
CREATE TABLE courses (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    title TEXT NOT NULL,
    code TEXT NOT NULL
);
```

Create a Table: students

```
CREATE TABLE students (
    student INTEGER NOT NULL,
    course INTEGER NOT NULL,
    score REAL
);
```

Create a Table: instructors

```
CREATE TABLE instructors (
    teacher INTEGER NOT NULL,
    course INTEGER NOT NULL
);
```

How do I remember all this stuff??

- Don’t!
- Use Google
- Use W3 Schools
- Use StackOverflow

Google

create table in sql

All Images Videos Shopping News More Tools

About 351,000,000 results (0.65 seconds)

https://www.w3schools.com › sql › sql_create_table

SQL CREATE TABLE Statement - W3Schools

The **CREATE TABLE** statement is used to create a new table in a database. ... The following example creates a table called "Persons" that contains five ...

https://www.w3schools.com › sql › sql_ref_create_table

SQL CREATE TABLE - W3Schools

The **CREATE TABLE** command creates a new table in the database. The following **SQL** creates a table called "Persons" that contains five columns: PersonID, LastName, ...

People also ask

How do you create a table with SQL?

What is the syntax of CREATE TABLE?

How do you create a new table?

Why do we CREATE TABLE in SQL?

Feedback

The SQL CREATE TABLE Statement

The `CREATE TABLE` statement is used to create a new table in a database.

Syntax

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

The column parameters specify the names of the columns of the table.

The datatype parameter specifies the type of data the column can hold (e.g. varchar, integer, date, etc.).

Tip: For an overview of the available data types, go to our complete Data Types Reference.

Adding Data (3:35)

Add Data to a Table

```
INSERT INTO <table>
(<col1_name>, <col2_name>, …)
VALUES
(<col1_value>, <col2_value>, …)
```

Add Data to a Table

```
INSERT INTO people
(name, senior)
VALUES
(“Connor”, TRUE)
```

Querying (3:40)

What is Querying?

- Once we have data in a database, we want to be able to access it!
- These queries are built around the **SELECT** keyword
- There are **tons** of methods for combining tables and refining our searches
- We'll go over just a few, but you'll practice this more in lab

Making Queries

```
SELECT <columns>
FROM <table>
WHERE <conditions>;
```

people

id (INTEGER)	name (TEXT)	senior (INTEGER)
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

Making Queries

```
SELECT *
FROM people;
```

people

id (INTEGER)	name (TEXT)	senior (INTEGER)
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

Making Queries

```
SELECT id, senior
FROM people;
```

people

id (INTEGER)	name (TEXT)	senior (INTEGER)
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

Making Queries

```
SELECT id, senior
FROM people
WHERE name = “sam”;
```

people

id (INTEGER)	name (TEXT)	senior (INTEGER)
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

Making Queries

```
SELECT id, senior
FROM people
WHERE name IN (“sam”,
“sophie”, “connor”);
```

people

id (INTEGER)	name (TEXT)	senior (INTEGER)
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

Query for the names of all non-seniors

```
— TODO
```

people

id (INTEGER)	name (TEXT)	senior (INTEGER)
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

Query for the names of all non-seniors

```
SELECT name
FROM people
WHERE senior = FALSE;
```

people

id (INTEGER)	name (TEXT)	senior (INTEGER)
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

What's actually returned?

```
SELECT name
FROM people
WHERE senior = FALSE;
```

people

name (TEXT)
sophie
sam
jason

More Complex Conditions

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

From W3 Schools!

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

More things you can do

- LIMIT keyword limits the number of results you'll get
- ORDER BY allows you to choose how the results are sorted
- Perform operations on data (MIN, MAX, AVG, COUNT, SUM)
- SELECT DISTINCT to get only unique values
- **LOTS MORE!!!**

Searching across tables

- Subqueries: Using results of one query in another one
  - Easier
  - Slower
  - Limited
- JOINs: Combining two tables together (temporarily)
  - Harder
  - Faster
  - More Powerful

Searching across tables

- **Subqueries**
- JOINs

```
SELECT id
FROM people
WHERE name = “connor”
```

people

id (INTEGER)	name (TEXT)	senior (INTEGER)
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

students

student (INTEGER)	course (INTEGER)	score (REAL)
1	3	23.6
3	1	96.7

```
SELECT score
FROM students
WHERE student in (
    SELECT id
    FROM people
    WHERE name = "connor"
)
```

people

id (INTEGER)	name (TEXT)	senior (INTEGER)
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

students

student (INTEGER)	course (INTEGER)	score (REAL)
1	3	23.6
3	1	96.7

How do we find a score in just one class?

```
SELECT score
FROM students
WHERE student in (
    SELECT id
    FROM people
    WHERE name = "connor"
) AND course in (
    SELECT id
    FROM courses
    WHERE code = "CS50"
)
```

id (INTEGER)	name (TEXT)	senior (INTEGER)
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

student (INTEGER)	course (INTEGER)	score (REAL)
1	3	23.6
3	1	96.7

Searching across tables

- Subqueries
- **JOINs**

```
<left_table>
<JOIN_TYPE> <right_table> ON
table.column=table.column
```

people

id (INTEGER)	name (TEXT)	senior (INTEGER)
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

students

student (INTEGER)	course (INTEGER)	score (REAL)
1	3	23.6
3	1	96.7

```
<left_table>
<JOIN_TYPE> <right_table> ON
table.column=table.column
```

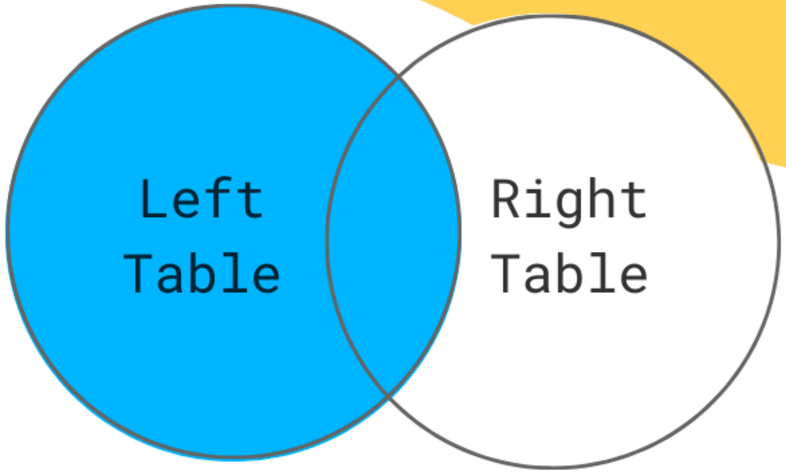
people

id (INTEGER)	name (TEXT)	senior (INTEGER)
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

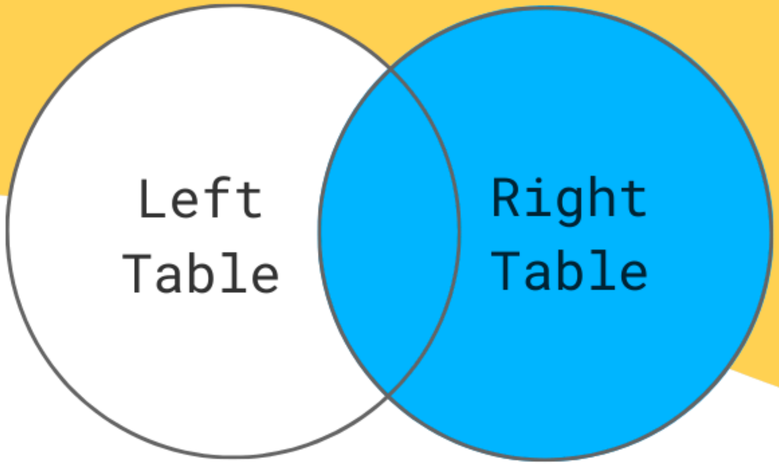
students

student (INTEGER)	course (INTEGER)	score (REAL)
1	3	23.6
3	1	96.7

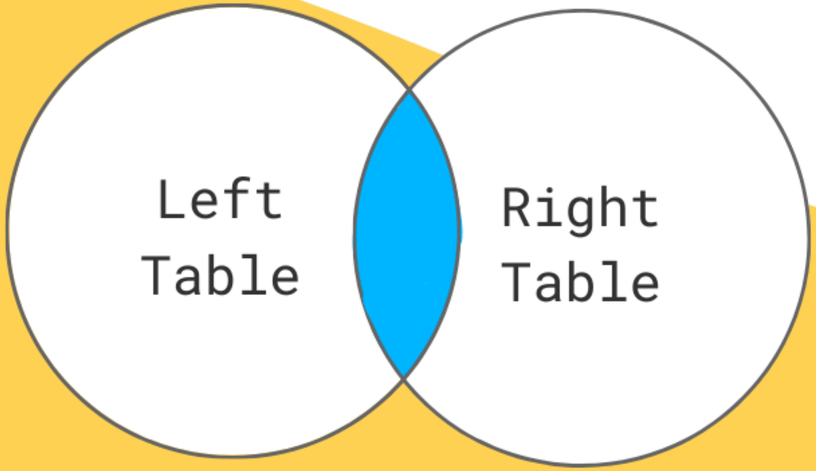
LEFT JOIN



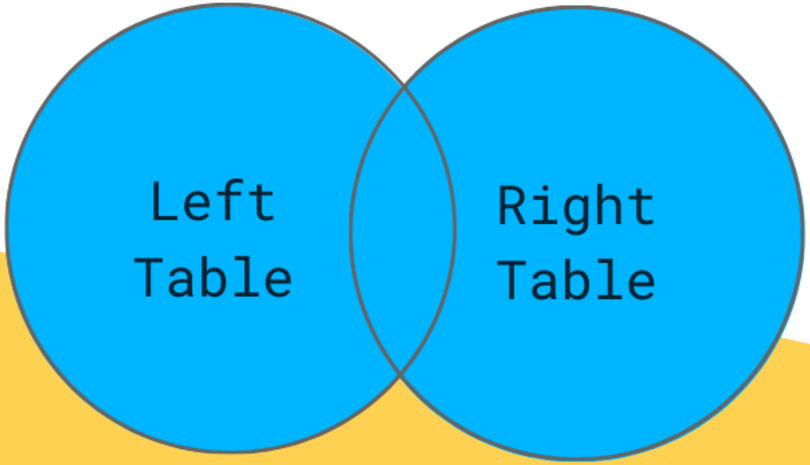
RIGHT JOIN



INNER JOIN



FULL JOIN



LearnSQL.com

```
people
INNER JOIN students ON
people.id=students.student
```

people

id	name	senior
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

students

student	course	score
1	3	23.6
3	1	96.7

```
people
INNER JOIN students ON
people.id=students.student
```

people

id	name	senior
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

students

student	course	score
1	3	23.6
3	1	96.7

```
people
INNER JOIN students ON
people.id=students.student
```

people

id	name	senior
1	connor	TRUE
3	sam	FALSE

students

student	course	score
1	3	23.6
3	1	96.7

```
people
INNER JOIN students ON
people.id=students.student
```

id	name	senior	student	course	score
1	connor	TRUE	1	3	23.6
3	sam	FALSE	3	1	96.7

```
SELECT name, score
FROM people
INNER JOIN students ON
people.id=students.student
```

name	score
connor	23.6
sam	96.7

Editing (3:50)

Editing

- Sometimes we’ll want to change a specific row or rows
- We do this using the UPDATE command

Making Edits

```
UPDATE <table_name>
SET column = value, column2 = value, …
WHERE condition
```

id	name	senior
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

Making Edits

```
UPDATE people
SET name = samuel
WHERE id = 3;
```

id	name	senior
1	connor	TRUE
2	sophie	FALSE
3	sam	FALSE
4	jason	FALSE

Making Edits

```
UPDATE people
SET name = samuel
WHERE id = 3;
```

id	name	senior
1	connor	TRUE
2	sophie	FALSE
3	**samuel**	FALSE
4	jason	FALSE

Compounding!

```
UPDATE students
SET score = 99.6
WHERE student in (
    SELECT id
    FROM people
    WHERE name = “Connor”
);
```

student (INTEGER)	course (INTEGER)	score (REAL)
1	3	23.6
3	1	96.7

Compounding!

```
UPDATE students
SET score = 99.6
WHERE student in (
    SELECT id
    FROM people
    WHERE name = “Connor”
);
```

student (INTEGER)	course (INTEGER)	score (REAL)
1	3	99.6
3	1	96.7

Be careful!

- Make sure your condition is specific enough?
- What if connor is taking more than one class?
- What if there’s more than one connor?

Python and SQL (3:55)

Python and SQL

- Many larger-scale programs are written using Python
- There are several methods we could use to run SQL code within Python.
- If you’re interested, some options are SQLAlchemy and the MySQL library
- For simplicity, in this class we’ll work with the **cs50** library

SQL in Python

```
from cs50 import SQL
```

SQL in Python

```
from cs50 import SQL

# Connect to dadabase
db = SQL("sqlite:///database_name.db")
```

SQL in Python

```
from cs50 import SQL

# Connect to dadabase
db = SQL("sqlite:///database_name.db")

# Insert new person
id = db.execute(
    "INSERT INTO people (name) VALUES (?)",
    "connor"
)
```

What does the execute function do??

- Let’s look it up!
- https://cs50.readthedocs.io/libraries/cs50/python/#cs50.SQL.execute

Exercises

github.com/cjleggett/2022-section

If There's Time: Test Prep

cs50.harvard.edu/college/2020/fall/test/

The Test

- Released 11/14, Due 11/20
- You can take as much time as you want, and stop/start as much as you want
- You can use all **non-human** resources!
- There will be a mix of coding and conceptual questions
- READ INSTRUCTIONS CAREFULLY!!!

Tutorials

Ed