CS50

SQL

Overview

Imagine that you have a sheet of paper with someone’s username on it and a separate sheet of paper with the matching password. It wouldn’t be very difficult to keep them together. But now imagine that you have thousands of these papers, all with corresponding pieces of information. How would you keep them organized? That’s where databases come in. A **database** is a program that stores data in an easily accessible, manageable, and updatable form. By organizing data in a database, programs can effectively and efficiently keep track of enormous amounts of information.

Key Terms

- database
- field
- record
- primary key
- persistent
- SQL

Databases

Databases look very similar to spreadsheets, like those found in Excel or Google Sheets. The data is organized in a table in **fields** (columns) and **records** (rows). Fields describe the data in a column and records link related pieces of information. In the example on the right we’ve stored the names, ages, and favorite foods of a few dogs. The field names of this table are id, name, age, and favorite food. Each row ties pieces of information together; we know that Elphie is age 2, and loves vanilla ice cream. In this table, the id is set to be the **primary key**, or a unique identifier for each record. Each table can only have one primary key. Similar to variables, each field has a particular data type. In this table, id and age are of the type INTEGER, and name and favorite food are of the type TEXT. There are also other SQL data types, such as BLOB (binary data), NULL (no value), REAL (floating-point value), DATETIME (dates and times), and NUMERIC (any kind of number).

id	name	age	favorite food
1	Elphie	2	vanilla ice cream
2	Milo	6	duck dog treat
3	Mochi	3	mochi

Say we also wanted to store the number of calories of these favorite foods. Instead of recreating my table to have an additional field, we can create a new database that stores a list of foods with their corresponding calorie count. Since these tables have the same foods on them, the information from both tables can be linked together. Databases work with programs, but they are separate files from your code. For this reason, databases are **persistent**: any changes made to the database remain after the program exits.

SQL

Structured Query Language, or **SQL**, is the standard language for managing, or “talking” with, a database. Using SQL, we can request, search, and filter data from our database. Take a look at these common commands for manipulating databases:

```
CREATE TABLE ‘dogs’ (‘id’ INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, ‘name’ TEXT, ‘age’ INTEGER, ‘favorite food’ TEXT)
```

This creates a table named dogs with specified fields and data types. The id is set to be the primary key and is automatically created when a new record is inserted. This way, we will never have a record with no id and no ids will ever repeat.

```
INSERT INTO “dogs” (“name”, “age”, “favorite food”)
VALUES (“Willow”, 4, “watermelon”)
```

The INSERT command inputs data for a new record, specifying field names and their corresponding values. Because we set id to be autoincremented, we don’t have to worry about assigning it ourselves.

```
SELECT * FROM “dogs”
```

The asterisk (*) means all in SQL. This allows us to select all records from dogs.

```
UPDATE “dogs” SET “age” = 2
WHERE id = 4
```

This updates the name of the record with the id 4. The primary key comes in handy whenever we need to retrieve a particular record.

```
DELETE FROM “dogs” WHERE id = 4
```

Delete the record at id 4