Foundations of Data Science for Biologists

More SQL

BIO 724D

04-MAR-2024

Instructors: Greg Wray and Paul Magwene

Creating and maintaining an SQL database

SQL is limited to a small set of operations

Only four kinds of operations are permitted with relational databases

Create, **R**ead, **U**pdate, **D**elete (CRUD)

`CREATE` to create a new database, table, or relation; `INSERT` to add rows to a table

`SELECT` to query a database ✔

`UPDATE` to change the information within a table

`DELETE` to remove rows from a table; `DROP` to remove a database, table, or relation

SQL has some additional keywords, but most work with the above set

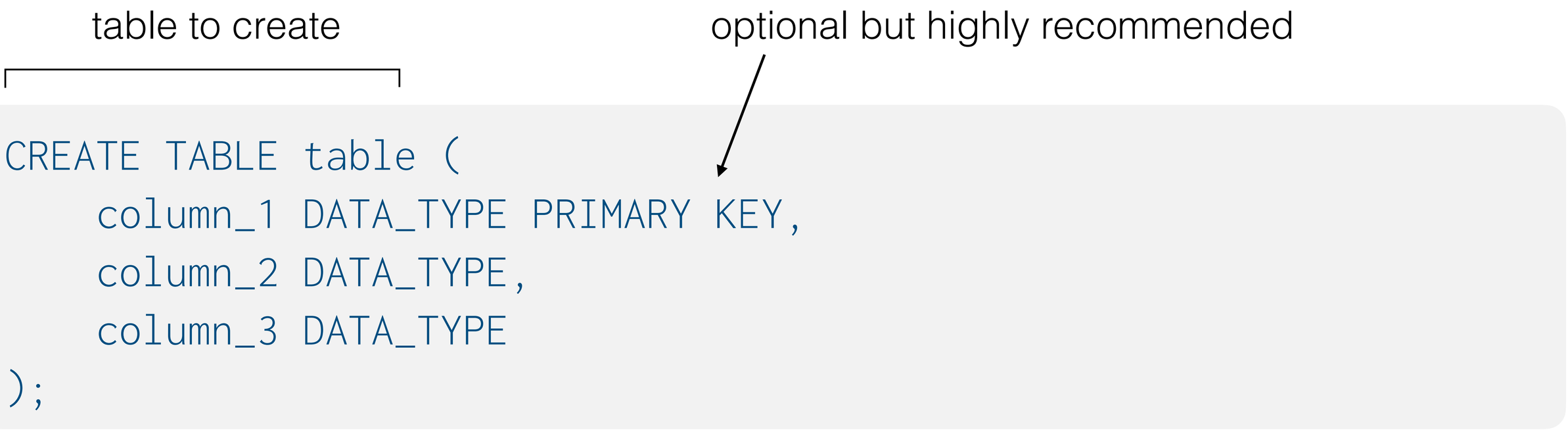
Clauses within statements: `WHERE`, `JOIN`, `GROUP BY`, `LIMIT`, etc.

Operators within statements: `=`, `>`, `IN`, `NOT`, `LIKE`, `BETWEEN`, etc.

Functions within statements: `MIN`, `MEAN`, `COUNT`, `DISTINCT`, etc.

`Pandas` and `dplyr` implement many SQL-like operations (and were inspired by it)

CREATE syntax



Data types typically include: `INTEGER`, `FLOAT`, `BOOLEAN`, `CHAR(n)`, `VARCHAR`, `DATE`, etc.

Optional qualifiers: `PRIMARY KEY`, `NOT NULL`, `UNIQUE`, etc.

INSERT and UPDATE syntax

table to update

```
INSERT INTO table VALUES (v1, v2, v3);
```

required new values clause

table to update

optional yet almost always needed clause

```
UPDATE table SET column = new_value WHERE condition;
```

required new value clause

DELETE and DROP syntax

table to update

```
DELETE FROM table WHERE condition;
```

required condition for row(s) to drop

table to delete

```
DROP table;
```

Constraints

Constraints are designed to ensure data integrity

Constraints are rules that govern what data can be present and how data interact

SQL provides several mechanisms that are easy to implement but powerful

Constraints should be defined when you create a table

- Some SQL implementations allow constraints to be added later
- But SQL will complain if the existing data don't conform to the new rule!

Single-table constraints

`UNIQUE`: ensures that every value in a column is distinct from all others

`NOT NULL`: ensures that a column contains no NULL values

`PRIMARY KEY`: both `UNIQUE` and `NOT NULL`; max of 1; also important for **normalization** (later)

`DEFAULT`: inserts a default value if no value is specified during `INSERT` or `UPDATE` operations

`CHECK`: ensures that the value in a column meets a specified condition(s)

Syntax for adding constraints

```
CREATE TABLE t1 (
    uid INTEGER PRIMARY KEY,
    genus VARCHAR NOT NULL UNIQUE,
    species VARCHAR NOT NULL,
    b_date DATE UNIQUE,
    age INTEGER DEFAULT = 42,
    entry_date DATE DEFAULT CURRENT_DATE,
    p1 INTEGER CHECK (c6 > 10 AND c6 != 0),
    p2 INTEGER CHECK (6 IN ('red', 'green', ‘blue’)),
);
```

Note: some versions of SQL use a different syntax for `CHECK` constraints and `CURRENT_DATE`

Two-table constraints

`FOREIGN KEY`: prevents actions that would destroy a relation between two tables

a foreign key depends on the values in a column in another table

the referenced column in the other table must contain unique values

`observations`

seq	genus	species	date	location
1023	Pycnonotus	tricolor	2007-06-14	Fairview Hotel
1024	Milvus	migrans	2007-06-14	Fairview Hotel
1032	Chalcomitra	amethystina	2007-06-14	Sheldrick Centre
1033	Pycnonotus	tricolor	2007-06-14	Sheldrick Centre

`locations`

location	prov	country	clim	elev	geolocation
Everard Reserve	Vic	Australia	Csb	90	-37.68,145.49
Everglades NP	FL	USA	Aw	2	25.39,-80.63
Fairview Hotel	Nb	Kenya	Cfb	1715	-1.29,36.80
Faskrudsfjordur	Au	Iceland	ET	20	64.93,-14.01

A foreign key prevents:

- Adding an observation with a location that does not exist in the locations table
- Removing a location from the locations table that is referenced by an observation

Syntax for adding constraints

```
CREATE TABLE t1 (
    uid INTEGER PRIMARY KEY,
    genus VARCHAR NOT NULL UNIQUE,
    species VARCHAR NOT NULL,
    b_date DATE UNIQUE,
    age INTEGER DEFAULT = 42,
    p1 INTEGER CHECK (c6 > 10 AND c6 != 0),
    p2 INTEGER CHECK (6 IN ('red', 'green', ‘blue’)),
    loc_name VARCHAR REFERENCES locations(loc_name)
);
```

keyword to create the foreign key

specifies the table and column to reference

Note: some versions of SQL use a slightly different syntax

NULL is a special value in SQL

Database design and normalization

Database design

A database should be organized to:

- Minimize mistakes when entering and updating data
- Minimize redundant information
- Simplify the query process

The solution to all these goals: database **normalization**

Normalization helps you decide:

- How many tables?
- What should be their primary keys?
- What relations should link the tables?

First normal form

Do not use row order to convey information

- Row order is neither stable nor predictable
- If consistent row order is needed for output, include an index column

Do not mix data types within a column

- Some spreadsheet programs allow this (looking at you, Excel)
- As a result, you may encounter errors when importing data

Do not use repeating groups

- Examples of repeating groups: lists, tuples, and dictionaries
- These values need to go into another table

A primary key is required

- Values must be unique within a single column or together across multiple columns
- The most natural way to think about a PK is as a row identifier

Second and third normal forms

Second normal form

- Concerns how non-PK columns relate to the PK
- Each non-PK must depend on the PK (i.e., its value is not a function of something else)
- Each non-PK must depend on the entire PK (i.e., *both* columns if a multi-column PK)
- If this is not the case, the column in question should be in another table

Third normal form

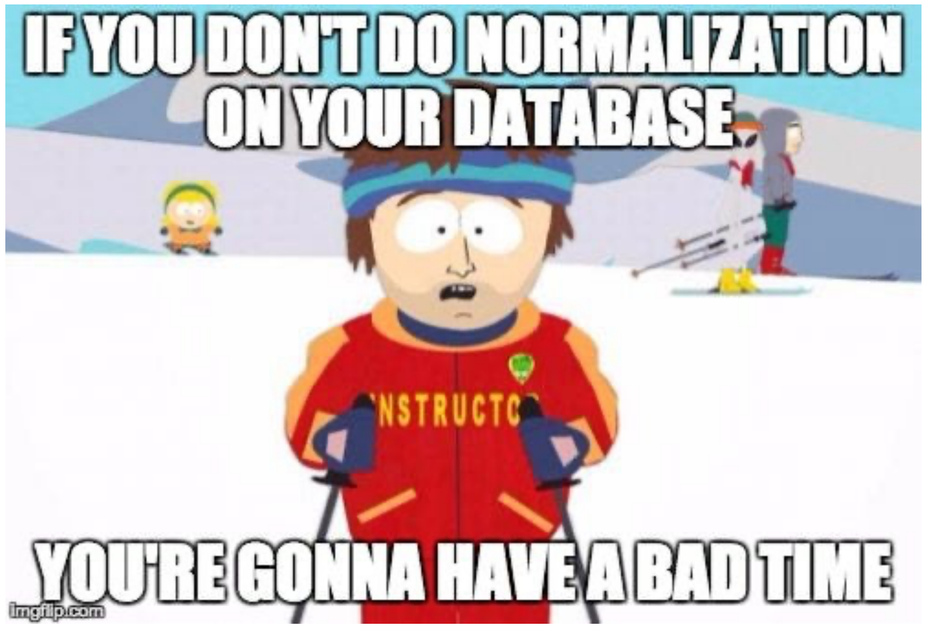
- Concerns transitive dependencies: non-PK column depends on another non-PK column
- Each non-PK must depend only on the PK
- If this is not the case, the column in question should be in another table

Most databases in third normal form are fully normalized

- There are some exceptions, involving fourth and fifth normal forms
- However, these are unusual and outside the scope of what we will cover

Summing up normalization

Every non-PK column must depend on the PK, the entire PK, and nothing but the PK



Database design

Figure out what information is fundamental and what is in a supporting role

Bird database: field observations are fundamental, the entire reason for its existence

Each observation is a separate unit and gets its own identifier

- This is the primary key of the most fundamental table

Several pieces of information are directly tied to specific observations

- E.g., species, time of day, date, location
- This information belongs in the observations table

Other pieces of information are independent of individual observations

- E.g., conservation status of the species seen, which family it belongs to
- This information belongs in other tables
- Those form relations and define the primary keys of the supporting tables

Using SQL within Python

Why would you use SQL within Python?

The best of both worlds: combine the flexibility of a GPL with the power of DSL

- Efficiently extract subsets of data using SQL
- Move immediately into analysis and visualization
- This is an example of using Python as a “wrapper” language


SQL BASICS DONE
ADVANCED SQL? DUSTED
imgflip.com