Datenbankdesign und -implementierung (Climate Change Datastory)

Team:
Lukas Gehrig, Simon Stähli, Vincenzo Timmel, Eric Winter

Generelle Fragen

Welches LE hast Du in der Challenge abgedeckt?

- LE1 (Auswahlverfahren und -kriterien)
- LE2 (Relationale Datenbanken (SQL))

Was sind Deine Vorkenntnisse in diesem LE aus Deiner beruflichen Erfahrung?

Wir haben beruflich keine Erfahrungen mit Datenbanken gesammelt.
Jedoch haben wir im vorherigen Semester schon etwas mit SQL-Abfragen gearbeitet.

Was war für Dich die wichtigste Erkenntnis?

Datenbanken sind ein gutes Werkzeug um mit grossen Datenmengen zu arbeiten. Wenn man sich beim Design der Datenbank Gedanken macht, spart man sich später sehr viel Zeit. Etwas zeit in die SQL-Abfragen investieren lohnt sich auch, da es viel effizienter ist als die Daten lokal zu bearbeiten (bei sehr grossen Datenmengen ist die lokale Bearbeitung (z.b. JOIN und UNION kaum mehr möglich).

Gibt es andere wichtige Punkte, die auf Deinen Lernerfolg einen wichtigen Einfluss hatten?

Es sticht nichts besonders heraus, alles hat zusammengespielt.

Spezifische Fragen, abhängig vom LE

LE1 (Auswahlverfahren und -kriterien):

Welche Datenbank oder andere Lösung hast Du gewählt?

Wir haben uns für eine relationale Datenbank entschieden. Da wir im vorherigen Semester bereits etwas mit PostgreSQL gearbeitet haben, entschlossen wir uns für diese Datenbank.

Welche Kriterien waren dafür ausschlaggebend?

Grund dafür war, dass wir mehrere Daten für die gleichen Zeitpunkte haben (und weil wir es ausprobieren wollten und lernen wollten).

Würdest Du bei einer gleichen Aufgabe dasselbe Datenbanksystem wählen und wieso?

Obwohl wir mit Zeitreihen gearbeitet haben, haben wir uns für eine relationale Datenbank (postgreSQL) entschieden. Wenn dies wieder der Fall wäre, dann würden wir uns für das gleiche System entscheiden.

LE2 (Relationale Datenbanken (SQL)):

Wie sieht das Datenmodell aus und wieso und wie bist Du auf dieses gekommen?

Das ERD des Datenmodells findet ihr auf der letzten Seite.
Da wir keine Erfahrung mit dem Erstellen von Datenbanken hatten, sind wir mit unserem ersten ERD erstmal in die Sprechstunde gekommen. Dort hat Simon uns dann einige Tipps mit auf den Weg gegeben (DB aus “Sensorsicht” aufbauen) und wir haben diese umgesetzt. Wir wollten die Vorteile einer relationalen Datenbank nutzen und so stand relativ schnell fest, welche Tables wir verwenden wollten und welche Spalten diese haben.

Würdest Du bei einer gleichen Aufgabe dasselbe Datenmodell erarbeiten und wieso?

Wir haben uns Gedanken gemacht, wie wir unsere Klimadaten aus diversen Quellen am besten Ablegen können. Der ‘locations’ Table und der ‘sensors’ Table standen fest, einzig fpr die ‘sensor_readings’ gab es zwei verschiedene Möglichkeiten:
Entweder für die verschiedenen Datentypen jeweils einen einzelnen Table, oder alle in einen grossen Table und dann NA in den beiden Typen die nicht benötigt werden (sichergestellt mit CONSTRAINT).

Wie sehen die Ausgangsdaten aus und was für eine Struktur weisen sie auf?

Bei den Klimadaten wird auf bestehende Daten zugegriffen. Diese kommen aus unterschiedlichen Quellen. Wir haben für die Kompetenz Webdatenbeschaffung Scripts geschrieben die diese Daten herunterladen und als pandas DataFrame zurückgeben (auch speicherbar als csv). Die Daten waren noch nicht normalisiert (wurde im Rahmen von Data Wrangling angepasst).
Nach der Normalisierung konnten die Daten direkt in die Datenbank geschrieben werden.

Gab es "Probleme" beim Import der Daten und wie bist Du diese angegangen?

Wir haben die Daten mit einem Python-Skript in die Datenbank eingepflegt. Mithilfe von SQLAlchemy und der Funktion pandas.DataFrame.to_sql() aus der pandas Library haben wir die Daten nach der Normalisierung an die Datenbank geschickt.
Im Hintergrund macht SQL-Alchemy für jede Zeile im DataFrame ein einzelnes INSERT-Statement. So hätte Einlesen von über 400’000 Temperaturmessungen etwa 2 Stunden gedauert.
Aus diesem Grund haben wir die Daten mit pgAdmin aus einem csv-File eingelesen (Dauer: etwa 15 Sekunden).
Auch das Updaten der Daten in der Datenbank findet per Funktion statt. Wenn unsere Datenquellen geupdatet werden, können wir problemlos unsere Funktionen verwenden um die Daten in der Datenbank aktuell zu halten.