FH HAGENBERG

PROJEKTARBEIT

Weather Tracer - Dokumentation

Autor:
Daniel ENGLISCH
Andreas ROITHER

Übungsleiter:
Daniel SKLENITZKA

16. November 2018

Version 1.0

Inhaltsverzeichnis

1 Datenbankdesign **2**
1.1 Beispieldaten Generierung . . . 3
1.1.1 Stationen, Addressen und Communities . . . 3
1.1.2 Messdaten . . . 3
1.1.3 Andere Daten . . . 3

2 Visual Studio Projektmappe **4**
2.1 Common.Dal.Ado . . . 4
2.2 Wetr.Domain . . . 4
2.3 Wetr.Dal.Interface . . . 4
2.4 Wetr.Dal.Ado . . . 4
2.5 Wetr.Dal.Factory . . . 4
2.6 Wetr.Test.Dal . . . 4
2.7 Wetr.Generator . . . 4

3 Installationsanleitung **6**
3.1 Benötigte Programme und Voraussetzungen . . . 6
3.2 Datenbank . . . 6

1 Datenbankdesign

Dieses Projekt wurde mit eines MySQL Datenbank auf Version 5.7.23 realisiert. Es wurden zuerst die geforderten Entitäten aus der Angabe extrahiert und mithilfe eines grafischen Modellierungswerkzeugs namens MySQL Workbench 8[1] modelliert.

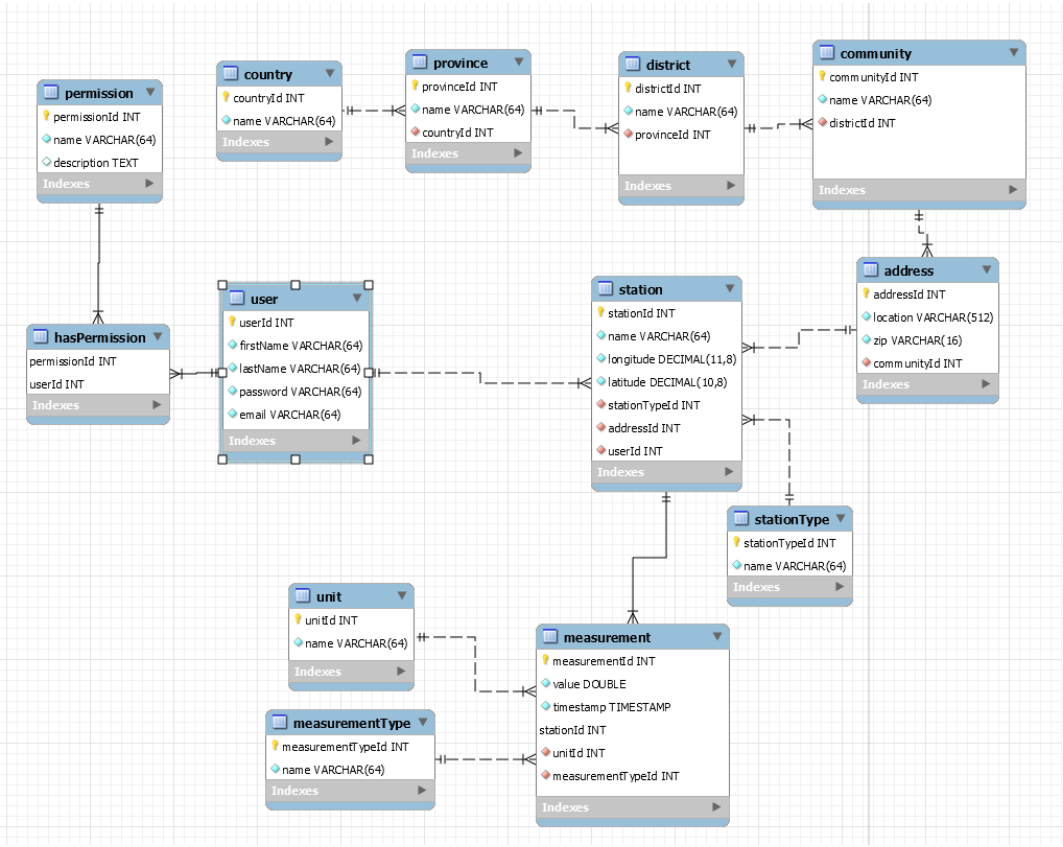

Abbildung 1: Datenbankschema des Wetr-Projekts in der ersten Ausbaustufe.

Wie in Abbildung 1 zu sehen wird zur Verwaltung des Standortes einer Station eine Reihe von abhängigen Entitäten verwendet. Um die Flexibilität zu erhöhen wurde neben zusätzlich ein *Country* modelliert. In einem *Country* befinden sich *Provinces*, welche Bundesländer darstellen. Jede *Province* wird in mehrere *Districts* unterteilt, ähnlich wie Bezirke. In jedem *District* gibt es mehrere *Communities*, welche mit Gemeinden vergleichbar sind. Als kleinste Entität in dieser Kette gibt es die *Address*, welche einen einfachen String zur Angabe von genaueren Addressdaten (rein zur Anzeige oder falls anderswo benötigt) und eine Zuordnung mittels Postleitzahl enthält.
Im Datenbankschema gibt es *User*, welche, falls benötigt, verschiedene *Permissions* zugewiesen haben können. Ein *User* kann mehrere *Stations* betreiben, welche wiederum neben der *Address* auch einen Namen und die Geokoordinaten in Form von Latitide und Longitude gespeichert hat. Der Typ der Station wurde in eine eigene Entität *StationType* ausgelagert.
Jede *Station* kann beliebig viele *Measurements* generieren, welche neben den ebenfalls ausgelagerten Entitäten *MeasurementType* und *Unit*, auch einen Zeitstempel und dazugehörigen Messwert besitzen.

[1] https://dev.mysql.com/downloads/workbench/

1.1 Beispieldaten Generierung

1.1.1 Stationen, Addressen und Communities

Der *Extractor* (befindet sich im extractor Ordner) wurde mit Python[2] geschrieben und verwendet die Stationsliste der *Zentralanstalt für Meteorologie und Geodynamik*[3]. Die " .csv" Datei wird vom Extractor eingelesen und für jede *Station* wird eine Insert Anweisung in eine " .txt" Datei geschrieben. Zusätzlich wird anhand des Längen- und Breitengrades der Ort mit einem geolocator der Aufenthaltsort der Station ermittelt. Anhand dieser Daten werden SQL Anweisungen für die *Community* und *Address* Tabellen erstellt die von der jeweiligen Stationen referenziert werden.

1.1.2 Messdaten

Für die erste Ausbaustufe dieses Projekts wurde ein Generator implementiert, der über eine Millionen Messdaten generiert. Diese Messdaten sind jedoch nicht realitätsnahe, sondern haben als Basis den Jahresdurchschnitt in Österreich laut Klimatabelle[4]. Für eine genauere Beschreibung siehe Abschnitt 2.7.

1.1.3 Andere Daten

Die Beispieldaten der restlichen Tabellen wurden per Hand mithilfe vom Internet zusammengestellt. *Provinces*, *Districts* und weiter Standortbezogene Daten sind höchstwarhscheinlich nicht vollständig übernommen worden.

[2]https://www.python.org/
[3]https://www.zamg.ac.at/cms/de/klima/messnetze/wetterstationen
[4]https://www.klimatabelle.info/europa/oesterreich

2 Visual Studio Projektmappe

Das gesamte Projekt befindet sich in einer Visual Studio Projektmappe, welch in folgene Unterprojekte gegliedert ist:

- **Common.Dal.Ado**: Hier befinden sich Hilfsklassen, die den Umgang mit ADO.NET leichter gstalten.
- **Wetr.Domain**: Dieses Projekt beinhaltet die Domainobjekte, welche als einfache Datenbehälter genutzt werden.
- **Wetr.Dal.Interface**: In diesem Paket befinden sich die Interfaces für die Datenzugriffschricht für jedes Domainobjekt bzw. Tabelle.
- **Wetr.Dal.Ado**: Die ADO.NET Implementierung der Datenzugriffsschicht Interfaces.
- **Wetr.Dal.Factory** Beinhaltet Factories um die Instanziierung der benötigen komkreten Klassen zu abstrahieren.
- **Wetr.Test.Dal** Dieses Projekt beinhaltet Unit-Tests für die Datenzugriffsschicht.
- **Wetr.Generator**: Zuständig für das Generieren von Messdaten für die einzelnen Stationen.

2.1 Common.Dal.Ado

2.2 Wetr.Domain

2.3 Wetr.Dal.Interface

2.4 Wetr.Dal.Ado

2.5 Wetr.Dal.Factory

2.6 Wetr.Test.Dal

2.7 Wetr.Generator

Der Generator generiert pro *MeasurementType* eine eigene Datei, in der sich die generierten Messdaten befinden. Pro Typ werden nicht genau gleich viele Messdaten generiert, somit ergibt sich eine Summe von etwa 1.2 Millionen Messdaten. Das Format der erzeugten Dateien ist so gestaltet, damit es mit einem speziellen SQL Befehl mittels Bulk-Insert[5] sehr schnell in die Datenbank aufgenommen werden kann. Die generierten Daten haben einen Zeitstempel, der sich innerhalb von einem Jahr bewegt.

Temperatur
Es wird jede Stunde ein Messwert generiert, der je nach Jahreszeit die Temperatur nach natürlichem Verlauf, sprich zur Mittagszeit ist es am wärmste und in der Nacht am

[5] https://stackoverflow.com/questions/14330314/bulk-insert-in-mysql

kältesten, gestaltet. Die Werte beinhalten eine zufällige Abweichung von ± 1.25. Der Minimal- bzw. Maximalwert wird pro Jahreszeit nach klimatabelle.info festgelegt.

Luftfeuchtigkeit
Die Berechnung der Luftfeuchtigkeit funktioniert gleich, wie die der Temperatur, nur, dass die durchschnittlichen Luftfeuchtigkeitswerte hergenommen wurden. Als zufällige Abweichung wurden $\pm 10\%$ gewählt.

Niederschlag
Da nur der jährliche Durchschnittsniederschlag pro Jahreszeit zur Verfügung stand, wurden nicht stündlich, sondern täglich ein Messert generiert. Dieser Wert bewegt sich zwischen 0 und der Anzahl des durchschnittlichen Tagesniederschalag Mal zwei.

Luftdruck
Jede Stunde wird ein Luftdruckwert generiert, der zufällig zwischen 900 und 1100 Hektopascal liegt.

Windrichtung
Die Windrichtung ändert sich in dieser einfachen Simulation jede Stunde und kann von 0 bis 360 Grad betragen.

Windstärke
Die Generierung der Windstärke ist in der aktuellen Ausbaustufe sehr primitiv gehalten und ändert sich stündlich zufällig von 0 und 20 km/h

3 Installationsanleitung

3.1 Benötigte Programme und Voraussetzungen

Für dieses Projekt wird zusätzliche Software benötigt:

- Docker [6]
- Visual Studio [7]
- MySql-Connector (optional, nur falls notwendig) [8]

Um die Datenbank zu erstellen muss Docker gestartet sein. Die Datenbank kann mit dem *Powershell-Script* "run.ps1 " automatisch generiert werden. Falls dieses Script wegen fehlender Berechtigungen nicht ausgeführt werden kann, muss eine *Shell* (Git-Bash oder ähnliches) im Ordner mit der Docker Compose Datei "docker-compose.yaml " geöffnet und folgenden Befehle nacheinander ausgeführt werden:

```
docker stop $(docker ps -a -q)
docker rm $(docker ps -a -q)
docker-compose up --build --force-recreate
```

3.2 Datenbank

Für die Datenbank muss die SQL Datei "create-wetr.sql " im Ordner sql/Create ausgeführt werden. Falls das Unit Testing auch ausgeführt werden soll, wird die zweite SQL Datei "create_wetr-unit.testing.sql " auch benötigt.

Für die Füllung der Datenbank kann die SQL Datei "InsertEverythingWithoutMeasurement.sql" verwendet werden. Für die Measurement Daten muss spezieller vorgegangen werden dadurch kann aber die Insert Zeit drastisch verringert werden:

- Wetr.Generator.exe im Ordner sql/Insert starten
- Die resultierenden ".bulk" Dateien in den sql Ordner kopieren (der phpMyAdmin Container wurde mit dem sql Ordner verbunden um diese Dateien verwenden zu können)
- http://localhost:8080/ im Browser aufmachen (phpMyAdmin sollte starten)
- Benutzer: root Passwort: 0c1cd84e
- Datenbank wetr auswählen
- Zum SQL Tab wechseln
- Die nachfolgenden SQL Anweisungen eingeben und ausführen

[6] https://www.docker.com/
[7] https://visualstudio.microsoft.com/de/
[8] https://dev.mysql.com/get/Downloads/Connector-Net/mysql-connector-net-8.0.13.msi

```
LOAD DATA LOCAL INFILE "/tmp/sql/insert/measurementsDownfall.bulk" INTO
 ↪  TABLE measurement
FIELDS TERMINATED BY ', ' ENCLOSED BY "'"
LINES TERMINATED BY '\r\n';

LOAD DATA LOCAL INFILE "/tmp/sql/insert/measurementsHumidity.bulk" INTO
 ↪  TABLE measurement
FIELDS TERMINATED BY ', ' ENCLOSED BY "'"
LINES TERMINATED BY '\r\n';

LOAD DATA LOCAL INFILE "/tmp/sql/insert/measurementsTemperature.bulk" INTO
 ↪  TABLE measurement
FIELDS TERMINATED BY ', ' ENCLOSED BY "'"
LINES TERMINATED BY '\r\n';

LOAD DATA LOCAL INFILE "/tmp/sql/insert/measurementsWind.bulk" INTO TABLE
 ↪  measurement
FIELDS TERMINATED BY ', ' ENCLOSED BY "'"
LINES TERMINATED BY '\r\n';

LOAD DATA LOCAL INFILE "/tmp/sql/insert/measurementsWindDirection.bulk"
 ↪  INTO TABLE measurement
FIELDS TERMINATED BY ', ' ENCLOSED BY "'"
LINES TERMINATED BY '\r\n';

LOAD DATA LOCAL INFILE "/tmp/sql/insert/measurementsPreassure.bulk" INTO
 ↪  TABLE measurement
FIELDS TERMINATED BY ', ' ENCLOSED BY "'"
LINES TERMINATED BY '\r\n';
```