UPA: Ukládání a příprava dat – dokumentace

Zvolené téma: **Databáze meteorologických dat**

Řešitelé: Bc. Josef Kolář (*xkolar71*), Bc. Timotej Halás (*xhalas10*), Bc. Vojtěch Hertl (*xhertl04*)

1 Zvolené dotazy a formulace vlastního dotazu

A vytvořte žebříček nejdeštivějších/nejsušších a nejteplejších/nejchladnějších meteorologických stanic/lokalit

B najděte skupiny meteorologických stanic s podobným počasím

C vizualizujte průměrnou teplotu vzduchu na kontinentu Austrálie i případnou interpolací hodnot do regionů bez měřících stanic

2 Stručná charakteristika zvolené datové sady

Na FTP serveru australského meteorologického úřadu se nachází datově i formátově široká sada mapující počasí po celém kontinentu Austrálie – pro účely tohoto projektu, resp. dle jeho zadání, bude k dalšímu zpracování použit výběr sedmi datových souborů ve formátu XML.

Jejich sběr ze vzdáleného serveru bude mít na starost samostatný Docker kontejner s příznačným názvem `scraper` – jeho implementace bude realizována v jazyce Python a během svého běhu se připojí na FTP server, detekuje aktualizované soubory od poslední kontroly, ty novější stáhne na lokální úložiště a následně je uloží do databáze MongoDB. Ta běží v samostatném kontejneru a její kolekce jsou popsány v sekci Zvolený způsob uložení uložení surových dat.

Jednou z důležitých částí datových XML souborů je specifikace stanice, ve které probíhají měření – ukázka toho fragmentu datové sady je umístěna níže v Ukázka způsobu uložení informací týkajících se konkrétní stanice. Mezi atributy důležité pro řešení úloh v tomto projektu je především dvojice `[lat, lon]` určující geografické umístění stanice, unikátní identifikátor stanice `wmo-id` (který bude použit pro navázání jednotlivých měření) a samotné jméno stanice dostupné z dvojice atributů `stn-name` a `description`.

```
<station
    wmo-id="94648" forecast-district-id="SA_PW001"
    stn-name="ADELAIDE (WEST TERRACE / NGAYIRDAPIRA)" type="AWS"
    lat="-34.9257" lon="138.5832" stn-height="29.32"
    description="Adelaide (West Terrace / ngayirdapira)"
><!-- all measurements from this station based on time --></station>
```

Kód 1: Ukázka způsobu uložení informací týkajících se konkrétní stanice.

Obsahem těchto elementů jsou poté elementy typu `<period />`, jehož atribut `time-utc` určuje čas konkrétního měření. Při zanoření do těchto elementů se poté dostáváme přímo ke změřeným datům, která jsou uložena v elementech typu `<element />` – ty ve všech případech obsahují atribut `type`, který značí, o jaký typ změřených dat se jedná. Samotná data jsou poté uložena jako obsah elementu a jejich jednotka, je-li to nutné, je doplněna v atributu `units`. Krácená varianta tohoto uložení se nachází níže v kódu Ukázka uložení meteorologických dat změřených v *21:50:00 UTC 26.9 .2020*.

```
<period time-utc="2020-09-26T21:50:00+00:00">
    <level type="surface">
        <element units="Celsius" type="apparent_temp">6.4</element>
        <element type="cloud">Partly cloudy</element>
        <element type="cloud_oktas">4</element>
        <element units="Celsius" type="delta_t">2.1</element>
        <element units="km/h" type="gust_kmh">9</element>
```

```
        <element units="knots" type="wind_gust_spd">5</element>
        <element units="Celsius" type="air_temperature">9.0</element>
        <element units="hPa" type="pres">1027.4</element>
        <element units="%" type="rel-humidity">72</element>
        <element units="km" type="vis_km">61</element>
        <element type="wind_dir">ENE</element>
        <element units="deg" type="wind_dir_deg">69</element>
        <element units="km/h" type="wind_spd_kmh">7</element>
    </level>
</period>
```

Kód 2: Ukázka uložení meteorologických dat změřených v *21:50:00 UTC 26.9 .2020*

3 Zvolený způsob uložení uložení surových dat

V rámci této databáze se bude jednat o kolekce `station` a `measurement`. První jmenovaná ukládá informace o samotné meteorologické stanici provozující měření. Kolekce `measurement` pak obsahuje všechna vykonaná měření a na stanici, která vykonala konkrétní měření, se odkazuje pomocí identifikátoru stanice.

- **station**
  - `wmo_id` – unikátní identifikátor stanice
  - `location` – stát a město, kde se stanice nachází
  - `station_name` – název stanice
  - `station_height` – nadmořská výška v metrech, ve které se stanice nachází
  - `latitude`, `longitude` – zeměpisná poloha stanice
- **measurement**
  - `station` – odkaz na unikátní identifikátor stanice
  - `time_period` – čas měření
  - `delta_t` – indikátor rychlosti vypařování
  - `dew_point` – teplota, na kterou musí být vzduch zchlazen, aby došlo k jeho kondenzaci[1]
  - `rel_humidity` – relativní vlhkost vzduchu
  - `vis_km` – viditelnost v kilometrech
  - `weather` – typ počasí slovně
  - `pres`, `msl_pres`, `qnh_press` – údaje popisující atmosferický tlak
  - `rain_hour`, `rain_ten` – množství srážek v milimetrech
  - `air_temperature`, `apparent_temp` – pocitová a reálná teplota vzduchu
  - `cloud`, `cloud_oktas`, `cloud_type_id` – typ oblaků, jejich pokrytí oblohy a slovní popis oblačnosti
  - `wind_dir`, `wind_dir_deg` – směr větru popsán slovně a úhlem
  - `wind_spd`, `wind_spd_kmh` – rychlost větru v uzelch a kilometrech za hodinu získáná průměrem za 10 minut
  - `wind_gust_spd`, `gust_kmh` – rychlost větru v uzelch a kilometrech za hodinu získáná ze 3sekundových měření
  - `rainfall`, `rainfall_24hr` – počet srážek v milimetrech od ranních 9:00 a historický údaj z předešlého dne

[1] též tzv. rosný bod – https://en.wikipedia.org/wiki/Dew_point

- `minimum_air_temperature`, `maximum_air_temperature` – minimální a maximální naměřená teplota od 18:00 do 9:00
- `maximum_gust_spd`, `maximum_gust_kmh`, `maximum_gust_dir` – maximální naměřená rychlost větru v uzlech a kilometrech za hodinu a jeho směr od půlnoci do půlnoci z 3sekundových měření

4 Implementace projektu a workflow

Projekt byl rozdělen do několika Docker kontejnerů, popsaných v souboru `docker-compose.yml` – konkrétně se jedná o hlavní kontejnery `scraper`, `mongo`, `computer`, `postgres`, `superset`, pomocné kontejnery `django-admin` a `redis`, a administrační kontejnery `mongo-admin` a `postgres-admin`. V následující části budou popsány základní zodpovědnosti hlavních kontejnerů a důvody pro zavedení dalších kontejnerů.

mongo

V Docker kontejneru `mongo` běží instance nerelační databáze Mongo DB ve verzi 4.4.1 k uložení načtených dat ze zdrojových souborů. Konkrétní podoba kolekcí odpovídá popisu v kapitole 3, jedná se o kolekci `station` s daty o stanicích a kolekci `measurement` s daty týkajících se konkrétních měření.

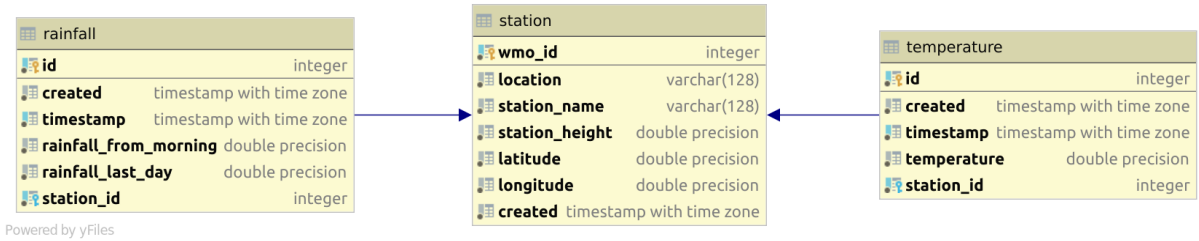
Do této databáze jsou vytvořeni dva uživatelé, jeden pro kontejner `scraper` s identickým názvem určeným pro import dat – druhý pak analogicky `computer` pro načítání dat pro další zpracování. K tomuto kontejneru také slouží administrační kontejner `mongo-admin` s nástrojem mongo-express verzi `0.54.0`, pomocí kterého lze administrovat tuto NoSQL databázi přímo skrz webový prohlížeč.

scraper

Kontejner s identifikátorem `scraper` slouží k importování dat ze zdrojových XML souborů do nerelační databáze – jako zdroj slouží buď lokální adresář, ze kterého jsou zdrojové XML soubory, nebo vzdálený FTP server, z jehož specifické složky XML soubory tento kontejner stáhne. Jeho implementace je založena na jazyku Python ve verzi 3.8.5 a používá sdílenou definici dokumentů (popsaných v 3) pro nerelační databáze, konkrétně na základě podpory ze strany knihovny `mongoengine`.

postgres

Tento kontejner slouží pro instanci relační databáze PostgreSQL ve verzi `12.4` – pro přístup do ní jsou nakonfigurováni tři uživatelé, jeden pro výpočetní kontejner, další pro administraci databázového schématu z kontejneru `django-admin` a třetí pro čtení dat kontejnerem `superset` s publikační vrstvou.



Obrázek 1: Databázové schéma pro uložení předpočítaných výsledků v relační databázi

computer

Výpočetní kontejner založený na jazyku Python s identifikátorem `computer` slouží ke spouštění agregačních dotazů nad nerelační databází v kontejneru `mongo`, zpracování výsledků a jejich následné uložení do databáze v kontejneru `postgres`.

Podrobnější popis dotazů a jejich řešení pomocí agregací následuje:

Dotaz A

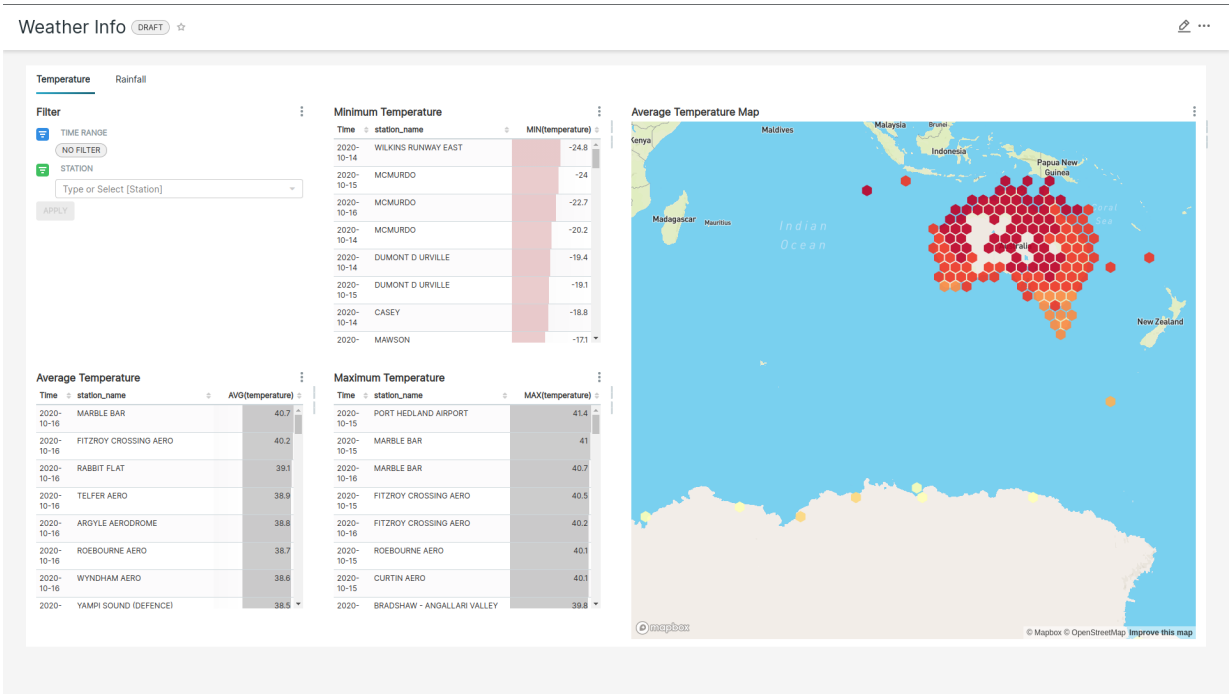
Vytvořte žebříček nejdeštivějších/nejsušších a nejteplejších/nejchladnějších meteorologických stanic/lokalit.

Získání dat v odpovídající struktuře a splňující podmínky prvního dotazu vychází z použití agregační *pipeline* odeslané na kolekci `measurement` do databáze MongoDB. Každá část dotazu je realizována samostatnou agregací, avšak obě sdílí následující strukturu:

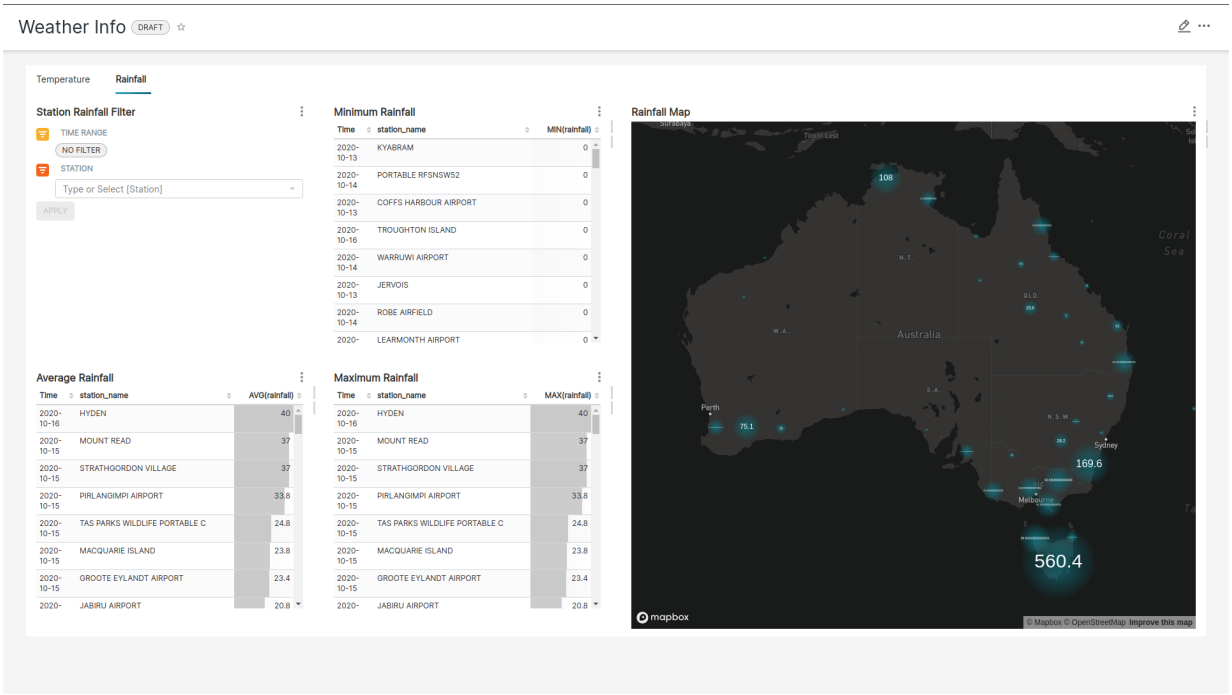
- **$match** – do dalšího zpracování se vezmou pouze dokumenty, která jsou validní pro danou část dotazu (tedy obsahují teplotu vzduchu, resp. úhrny srážek)
- **$lookup** – k datům měření se připojí data o stanici na základě klíče `station_id`
- **$unwind** – tento krok zajistí rozexpandování nalezených stanic k každému měření – vzhledem k povaze dat však ke každému měření odpovídá právě jedna stanice
- **$project** – do dalšího zpracování jsou vzaty v potaz pouze některé atributy – zejména samotná data identifikující měření (stanice, měsíc měření, čas měření) a také samotná hodnota měření
- **$sort** – dokumenty jsou seřazeny dle stanice a času pořízení (pro snazší ladění)
- **$group** – koncové seskupení do dávek měření, které odpovídající kompozitnímu klíči `[station_id, timestamp]` a obsahují všechny měření z konkrétní stanice a času – v samotných záznamech se poté již vyskytuje pouze změřená hodnota (resp. hodnoty) a konkrétní časový otisk

V případě nejdeštivějších/nejsušších míst byl problém s tím jak hledět na data, protože srážky jsou spojitá veličina a data obsahují počet srážek v čase od 9:00 do 9:00 následujícího dne v časovým pásmu stanice. Pro jednoduchost je čas měření počtu srážek brán jak koncový čas tohoto měření.

V případě odpovědi na otázku nejdeštivějších/nejsušších stanic se naskytl problém časové lokality údajů – úhrn srážek je z pohledu datové sady spojitá veličina, ovšem uložen je diskrétně jako kumulativní množství úhrnu srážek od ranních 09:00 do 09:00 dalšího dne ve specifických časových intervalech. Pro zachování jednoduchosti byla v tomto případě vzata v potaz pouze koncová hodnota předcházející vynulování kumulované hodnoty pro následující den – tedy poslední hodnota před ranními 09:00.



Obrázek 2: Výsledná vizualizace teploty



Obrázek 3: Výsledná vizualizace srážek

Dotaz B

Najděte skupiny meteorologických stanic s podobným počasím.

Před samotným řešením problému je vhodné lokálně definovat *stanice s podobným počasím* – pro účely tohoto dotazu mají dvě stanice podobné počasí, jestliže lze prokázat pozitivní korelaci mezi daty změřenými v těchto stanicích v odpovídajícím čase a zároveň v rámci stanovené odchylky odpovídají statistické metriky těchto měřených dat. Tedy pro příklad, pro konkrétní den odpovídá střední hodnota a směrodatná odchylka teplot vzduchu pro dvě *stanice s podobným počasím* a zároveň je srovnatelný týdenní vývoj tohoto změřeného parametru.

Možný SQL dotaz, který splňuje specifickou část požadavku je zobrazen v 4 a je založen na technice

CTE implementované v použité databázi PostgreSQL. Jeho funkce je založena na vytvoření pomocné datové sady obsahující agregovaná data tak, že pro každou stanici a týden v roce je uložena standardní odchylka a střední hodnota teplot vzduchu v tom konkrétním týdnu. Takto připravená data jsou následně agregovaná dle obou statistických ukazatelů do množin dvojic (stanice, týden v roce), které již odpovídají *stanicím s podobným počasím* – aby došlo úspěšně ke generalizaci, jsou oba statistické ukazatele zaokrouhleny na celá čísla.

Problém *podobného počasí* lze přímo v modelové vrstvě řešit i mnohem sofistikovanějšími metodami, například pomocí n-dimenzionálního rozmístění stanic dle změřených příznaků a následného hledání skupin stanic klasifikátorem `k-means` či jiného klasifikátoru určeného ke *clusterizaci*. Takto fungující klasifikátor by mohl, dle znalostí autorů tohoto projektu, být založen na vlastní `window` funkci založené na technice CTE v rekurzivní variantě.

```
with measurements as (
    select
        s.wmo_id,
        extract(weeks from t.timestamp) as week_number,
        ROUND(cast(stddev(temperature) as numeric), 0) as temperature_stddev,
        ROUND(cast(avg(temperature) as numeric), 0) as temperature_avg
    from temperature t
        inner join station s~on s.wmo_id = t.station_id
    where
        extract(year from t.timestamp) = 2020
    group by extract(weeks from t.timestamp), s.wmo_id
)
select
    m.temperature_avg,
    m.temperature_stddev,
    array_agg(cast(ARRAY [m.wmo_id, m.week_number] as int[2])) as stations_in_time
from measurements m
where
    temperature_stddev is not null and temperature_avg is not null
group by m.temperature_avg, m.temperature_stddev;
```

Obrázek 4: SQL dotaz pro nalezení *stanic s podobným počasím* v konkrétních týdnech – analogicky by vznikaly skupiny stanic na základě dalších parametrů *počasí*.

Dotaz C

Vizualizujte průměrnou teplotu vzduchu na kontinentu Austrálie i případnou interpolací hodnot do regionů bez měřících stanic.

Vizualizace změřených hodnot pro jednotlivé stanice na dotčeném území již byla realizována jako součást odpovědi na dotaz A, viz snímky prezentační vrstvy 2 a 3. Zbytek této sekce se bude věnovat návrhu algoritmu pro interpolaci změřených hodnot i pro území, na kterých se měřící stanice nenachází a tím pádem nejsou z těchto lokalit data.

Následuje konceptuální návrh algoritmu pro interpolaci počasí pro celé území.

1. vytvoření hodnotové matice matice nad zemepisnou šířkou a délkou – tedy $latitude \times longitude$ pro rozsah území našeho zájmu s hustotou polí dle vstupního parametru (parametru, co určuje úhel/délku polí)
2. pro pole matice, ve kterých se nachází nenulové množství měřících stanic, provést agregaci těchto stanic na základě interpolovaného parametru do střední hodnoty a směrodatné odchylky
3. pole s nulovým počtem stanic seřadit dle množství sousedních polí s hodnotami sestupně, stanovit první jako interpolované pole a aplikovat interpolaci:
   (a) z průměru středních hodnot a směrodaných odchylek okolních neprázdných polí simulovat normální rozdělení pro interpolované pole

(b) použít toto rozdělení pro vygenerování hodnoty, která je stanovena jako střední hodnota interpolovaného pole

(c) směrodatnou odchylku tohoto rozdělení stanovit jako směrodanou odchylku interpolovaného pole

4. iterativně doplňovat hodnoty do matice v rámci bodu 3.
5. ukončit iteraci, jestliže jsou všechna pole vyplněna
6. výstupem algoritmu jsou následně střední hodnoty z každého pole z původního rozdělení

superset

Tento kontejner slouží pro instanci nástroje Apache Superset, který je určen pro publikaci a vizualizaci zpracovaných dat uložených v databázi v kontejneru `postgres`. Jako pomocný kontejner pro cache slouží `redis` s instancí stejnojmenné databáze typu `key-value`.

5 Spuštění a práce s projektem

Požadovaný software

- Docker engine 19.0 a vyšší
- Docker compose 1.25 a vyšší

Příprava prostředí

Pro lokální běh projektu je nutné vytvořit soubor s lokálním nastavením `.local.env` – šablona pro jeho vytvoření je uložena v `.local.env.template` a kromě přístupového API klíče pro externí služby prezentační vrstvy je zde nutné nastavit klíče pro bezpečné uložení proměnných webových sezení nebo nastavit specifickou hodnotu pro určení úrovně logování `LOG_LEVEL`.

Inicializace služeb

1. Zapnutí potřebných kontejnerů:
   - spuštění kontejnerů
     ```
     $ make up
     ```
   - spuštění kontejnerů na pozadí
     ```
     $ make upd
     ```
2. Po spuštění příkazu je nutné vyčkat na nastartovanání služeb v kontejnerech a následně je možné inicializovat projekt následujícím příkazem (při běžících kontejnerech):
   ```
   $ make init
   ```
3. A následně je možné načíst výchozí konfiguraci pro prezentační vrstvu:
   ```
   $ make restore-superset
   ```

Zastavení služeb

Pro zastavení služeb je možné použít příkaz `$ make down`.

Správa dat

Pro přidání dat do NoSQL databáze je nutno v kořenu projektu vytvořit adresář `pocasi` a umístit do něj datovou sadu souborů určenou pro načtení – to lze realizovat pomocí `$ make scrape`.

Následné spuštění výpočetní vrstvy je možné pomocí příkazu `$ make compute`.

Správa uživatelů

Vytvoření uživatelů a administrátorů lze docílit dvěma způsoby – buď v grafickém rozhraní Apache Superset nebo přes příkazovou řádku – druhé pomocí `$ make create-admin`, resp. `$ make create-user`.

Porty služeb

Služba	Port
mongo-admin	8081
postgres-admin	8082
superset	8088
postgres	5432
mongo	27017
redis	6379