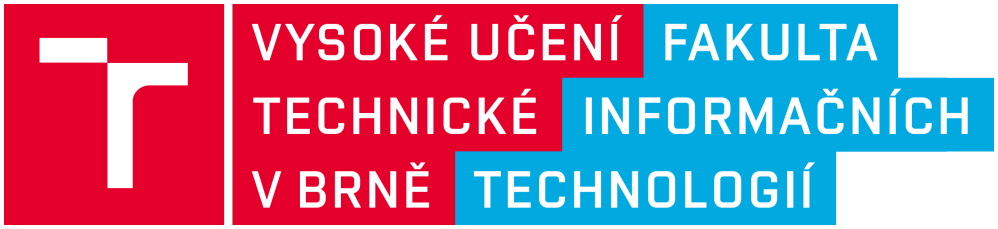


Ukládání a příprava dat

Ukládání rozsáhlých dat v NoSQL databázích

Tereza Burianová (xburia28)
Mário Harvan (xharva03)

13. listopadu 2023

1 Sloupcová wide-column databáze

URL: Plynulost dopravy
Zdroj dat: Waze, Magistrát města Brna

Pro reprezentaci sloupcové wide-column databáze byl vybrán produkt Apache Cassandra, demonstrován na datové sadě "Plynulost dopravy". Datová sada obsahuje různé datové typy, jako například číselné datové typy, řetězce, nebo časové značky daných záznamů. Jedná se o jednoduché datové typy, které existují i pro relační databáze, ovšem dokumentace datové sady připouští možnost přidání nových polí, tedy proměnlivost její celkové struktury. Proto je použití produktu Apache Cassandra, nebo podobného, vhodné pro tato konkrétní data: databáze je připravena na přidání nových polí s různými datovými typy, ovšem nejedná se o zcela nestrukturovaná data, která by vyžadovala použití například dokumentové databáze. Každý záznam taktéž obsahuje unikátní identifikátor, jak se touto databází vyžadováno.

Dalším z důvodů, proč je tento produkt pro vybranou datovou sadu vhodný, je použitá architektura. Apache Cassandra využívá peer-to-peer architekturu, data jsou tedy téměř vždy dostupná. Pro aplikace, které slouží řidičům jako navigace nebo aktuální zdroj informací, je dostupnost dat důležitou vlastností. Cassandra je schopná efektivně pracovat s velkým množstvím dat a umožňuje nastavení rovnováhy mezi výkonem, tedy rychlostí ukládání dat a přístupu k nim, a konzistencí dat. Při datech z aplikace Waze se vzhledem k počtu uživatelů, kteří při jejím používání nahlašují případná narušení dopravní plynulosti, a také vzhledem k častému obnovování informací, které probíhá v řádu sekund, jedná o vysoké množství dat, která jsou vkládána do databáze, a taktéž o vysoký počet dotazů nad nimi. Zachování konzistence použitých dat v tomto případě není prioritou, je tedy možné vytvořit kompromis pro vyšší výkonost.

Přestože datová sada obsahuje data počínající v roce 2020, pro účely aplikace používající vytvořenou databázi by mohlo být vhodné data po dané době odstranit a uchovat pouze aktuální informace. Tento přístup Cassandra umožňuje pomocí speciální hodnoty "tombstone", zapsané k záznamům, které již např. na základě hodnoty TTL (Time To Live) nejsou aktuální a mohou tedy být postupně odstraněny. [1] [2]

Definice úložiště, import dat, dotazování

Vytvoření "keyspace":

```
cqlsh
CREATE KEYSPACE IF NOT EXISTS plynulostDopravy
WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : 1};
DESCRIBE KEYSPACE plynulostDopravy;
USE plynulostDopravy;
```

Vytvoření tabulky:

```
CREATE TABLE IF NOT EXISTS nepravidelnostDopravy (
    country text,
    level tinyint,
    city text,
    speedKMH smallint,
    length int,
    turnType text,
    globalid text,
```

[1] https://cassandra.apache.org/doc/latest/cassandra/cql/
[2] https://www.influxdata.com/comparison/databases/cassandra/

```
    type text,
    uuid int,
    endNode text,
    speed int,
    blockingAlertUuid text,
    roadType smallint,
    delay int,
    street text,
    objectid int,
    pubMillis timestamp,
    PRIMARY KEY ((street), pubMillis)
) WITH comment = ' Liniová vrstva plynulosti dopravy na cestách v Brně.';
```

Primární klíč se skládá ze dvou částí: *partition key*, který je v uvedeném příkladu ulice, a clustering key, který je časové razítko daného záznamu. Důvod zvolení tohoto složení klíče bude dále vysvětlen po uvedení možných dotazů. [3]

Import dat:

```
COPY nepravidelnostDopravy (country,level,city,speedKMH,length,turnType,
globalid,type,uuid,endNode,speed,blockingAlertUuid,roadType,delay,street,
objectid,pubMillis) FROM '/usr/Plynulost_dopravy___Traffic_delays.csv'
WITH HEADER = true;
```

Cassandra podporuje import dat ze souboru csv, ovšem zvolená data bylo třeba upravit. Pro jejich úpravu byla zvolena metoda vyhledání a nahrazení textu pomocí regulárních výrazů. Úprava formátu data:

```
    ([0-9]{4})\/([0-9]{2})\/([0-9]{2}) nahrazeno $1-$2-$3
```

Nahrazení prázdných polí ulice hodnotou "null":

```
    ,(,[0-9]+,[0-9]{4}-[0-9]{2}-[0-9]{2}) nahrazeno ,null$1
```

Vložení nových dat, případně úprava již existujících se stejnou ulicí a časem (UPDATE záznam upraví, pokud existuje, případně ho přidá):

```
UPDATE nepravidelnostdopravy SET country='EZ',level=2,city='Brno',speedKMH=30,
length=120,turnType='NONE',globalid='{03B015CD-9D06-19C1-8969-EA03A1ABE037}',
type='NONE',uuid=1124185336,endNode='Kosmova',roadType=1,objectid=636286
WHERE street = 'Božetěchova' AND pubMillis = '2023-10-28 16:20:10.000+00';
```

Dotaz na nepravidelnosti dopravy na ulici Božetěchova:
Historická data řazena dle času

```
SELECT pubMillis,level,speedKMH FROM nepravidelnostdopravy
WHERE street = 'Božetěchova' ORDER BY pubMillis;
```

Nejnovější záznam

```
SELECT pubMillis,level,speedKMH FROM nepravidelnostdopravy
WHERE street = 'Božetěchova' ORDER BY pubMillis DESC LIMIT 1;
```

Tabulka je navržena tak, aby počítala s vyhledáváním záznamů na základě ulice, neboť používá ulici jako partition key. Je tak umožněno, aby se v architektuře data se stejnou ulicí soustředila do stejných uzlů, dotaz se odesílá na menší množství uzlů a proces je efektivnější. Pro zajištění unikátnosti a také umožnění práce s daty při dotazování je datum použito jako clustering key.

[3] https://rychly-edu.gitlab.io/dbs/nosql/cassandra-lab/

2 Dokumentová databáze

URL: Indukční smyčky a objekty se znakujícím personálem
Zdroj dat: Magistrát města Brna

Pro reprezentaci Dokumentové databáze byl vybrán produkt Atlas MongoDB, demonstrovaný na datové sadě "Indukční smyčky". Datová sada obsahuje GPS souřadnice míst, na kterých se nacházejí indukční smyčky. Každý datový záznam obsahuje doplňující údaje o dané datové smyčce, např. typ, název, popis, webová adresa. Dokumentová databáze MongoDB je vhodná pro takový typ dat z několika důvodů. Prvním významným faktorem je podpora geolokačních indexů. Tyto indexy umožňují efektivní vyhledávání geografických dat, včetně GPS souřadnic, což je nezbytné pro mapové služby, navigace a aplikace zaměřené na lokalizaci. Geospatiální dotazy jsou dalším výhodným prvkem. MongoDB poskytuje rozsáhlou sadu geospatiálních dotazů a operátorů, které umožňují provádět různé operace na geografických datech, jako je hledání bodů uvnitř určité oblasti nebo výpočty vzdáleností mezi body. Díky podpoře těchto funkcí by bylo snadné postavit nad databází aplikaci, která by uživatelům umožnila vyhledávat nejbližší Indukční smyčky a následně je k nim navigovat. Aplikace by také mohla zobrazovat oblasti, kde jsou indukční smyčky dostatečně rozšířeny, a následně umožnit městu zaměřit se na oblasti, kde je třeba vybudovat nové indukční smyčky.

Flexibilita schématu databáze je také velmi užitečná, protože geografická data mohou mít různé formáty a schémata. MongoDB je schopna pracovat s nestálými schématy dat, což umožňuje přizpůsobovat strukturu geografických dat podle konkrétních potřeb aplikace. Zvolená datová sada by mohla být v budoucnu jednoduše rozšířena o doplňující údaje pro každou indukční smyčku. Doplňující údaje by mohly být např. typ zařízení, aby mohli uživatelé snadno zjistit, zda je daná indukční smyčka kompatibilní s jejich naslouchacím zařízením.

V neposlední řadě, MongoDB poskytuje podporu pro různé typy dat spojených s geografickými daty, jako jsou popisy míst a fotografie. Rozšíření datové sady o tyto údaje by mohlo pomoci uživatelům snáze vybrat, která lokalita je pro ně atraktivní. Díky těmto vlastnostem je MongoDB výbornou volbou pro aplikace, které pracují s geografickými daty a umožňuje efektivní ukládání, správu a vyhledávání těchto dat s ohledem na škálovatelnost a flexibilitu. [4] [5]

Definice úložiště, import dat, dotazování

Vytvoření databáze a kolekce:

```
use test

db.createCollection("indukcnismycky")
```

Import dat:

```
mongoimport --db test --collection indukcnismycky --type csv
--headerline --file geograficke_data.csv
```

Pro importování dat z csv použijeme jednoduchou aplikaci mongoimport, která dokáže importovat data přímo z csv souboru. Data bude třeba následně upravit, aby byly GPS souřadnice ve vhodném tvaru pro použití Geospatial funkcí.

Úprava formátu data:

```
db.tollbooth.find().forEach(function (item) {
    loc = [item.X, item.Y]
    item.loc= loc
```

[4] https://www.mongodb.com/basics/examples
[5] https://www.w3schools.com/mongodb/

```
    db.tollbooth.save(item)
})
```

Vložení nových dat:

```
db.indukcnismycky.insertOne({
  \loc" : [
        1844526.3142,
        6306773.6649
    ],
  "ObjectId": 1,
  "ogcfid": 15574,
  "typ": "indukční smyčka",
  "nazev_cely": "Dopravní hřiště Riviéra",
  "popis": "Indukční smyčka je instalována v areálu dopravního hřiště.",
  "web_1": "http://www.dopravnihristebrno.cz",
  "web_2": "",
  "adresa": "Bauerova 7",
  "upresneni_": "",
  "datum_exportu": new Date("2023-08-17T00:00:00.000Z"),
  "GlobalID": "{F122BBEA-10D1-4CC6-9731-791FDA6C65A7}"
});
```

Dotaz na polohu nejbližší smyčky od polohy uživatele s omezením maximální vzdálenosti:

```
db.collection("indukcnismycky")
    .find({
      lokacia: {
        $near: {
          $geometry: {
            type: "Point",
            coordinates: [1844526.3142, 6306773.6649],
          },
          $maxDistance: 1000, // Maximalni vzdalenost v metrech (uprava dle potreby)
        },
      },
    })
```

Dotaz na polohu všech smyček v dané oblasti:

```
db.collection("indukcnismycky")
    .find({
      lokacia: {
        $geoWithin: {
          $geometry: {
            type: "Polygon",
            coordinates: [
              [
                [longitude1, latitude1],
                [longitude2, latitude2],
                [longitude3, latitude3],
                // pridejte dalsi souradnice podle potreby
```

```
            ],
          ],
        },
      },
    },
  })
```

Databáze je navržena tak, aby se dalo velmi jednoduše pracovat s geografickými daty. V dotazech jsme ukázali, jak jednoduché je vyhledávat indukční smyčky v blízkosti uživatele, případně zobrazit všechny smyčky v definované oblasti. MongoDB samozřejmě nabízí také jiné funkce pro práci s geografickými daty.

3 Grafová databáze

URL: Jízdní řád IDS JMK ve formátu GTFS
Zdroj dat: KORDIS, Magistrát města Brna (data.brno.cz)

Pro reprezentaci grafové databáze byl vybrán produkt Neo4j, demonstrován na datové sadě "jízdní řád IDS JMK ve formátu GTFS". Grafové databáze jsou oproti ostatním druhům databází, demonstrovaným v tomto dokumentu, velmi specifické, neboť se soustředí tolik na samotné uložení dat, jakožto na znázornění vztahů mezi daty. Proto je Neo4j obzvláště vhodný pro znázornění vybrané datové sady, popisující jízdní řády, která disponuje mnoha složitými vztahy mezi entitami jako například trasy, jízdy, zastávky a časy příjezdů a odjezdů, často s kardinalitou M:N. Vztahy jsou uloženy podobným způsobem, jako samotné entity, vyhledávání vztahů v databázi je tedy efektivnější, než například při relační databázi, která vyhledává vztahy pomocí operace "join". Neo4j navíc podporuje indexaci pro efektivnější dotazování. [6]

Definice úložiště, import dat, dotazování

Následující příkazy byly vytvořeny podle návrhu (Obrázek 1), který odpovídá referenční dokumentaci formátu GTFS [7]. Veškeré příkazy jsou napsány v jazyce Cypher. [8]
Vytvoření omezení pro unikátní hodnoty:

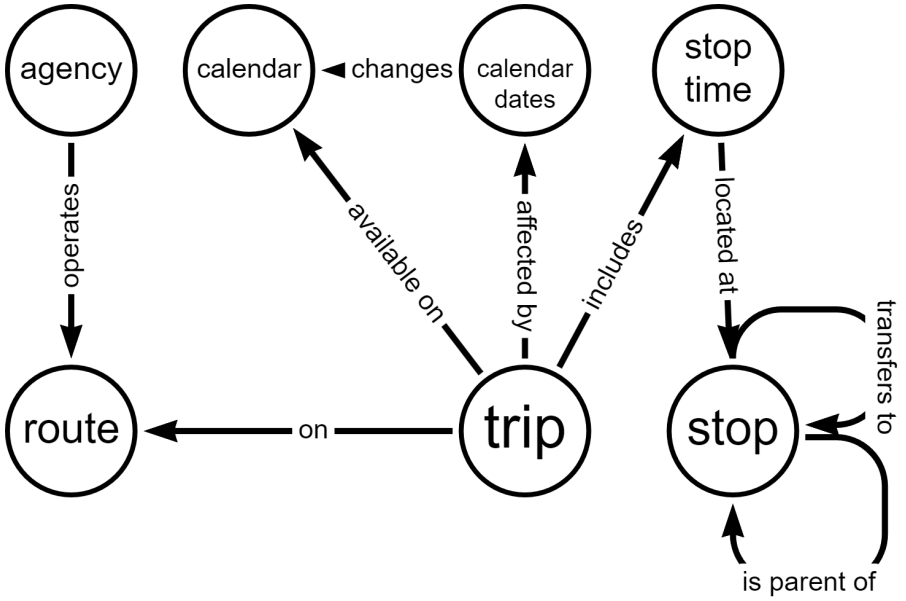
```
cypher-shell
create constraint for (a:Agency) require a.agency_id is unique;
create constraint for (s:Stop) require s.stop_id is unique;
create constraint for (r:Route) require r.route_id is unique;
create constraint for (r:Calendar) require r.service_id is unique;
create constraint for (t:Trip) require t.trip_id is unique;
```

Neo4j podporuje vložení z formátu csv. Vytvoření uzlů a vztahů, import dat: [9]

```
load csv with headers from
'file:///agency.txt' as csv
create (a:Agency {id: csv.agency_id, name: csv.agency_name, url: csv.agency_url});

load csv with headers from
'file:///stops.txt' as csv
create (s:Stop {id: csv.stop_id, name: csv.stop_name, lat: toFloat(csv.stop_lat),
```

[6] https://neo4j.com/developer/graph-database/
[7] https://developers.google.com/transit/gtfs/reference/
[8] https://neo4j.com/docs/cypher-cheat-sheet/5/neo4j-community
[9] https://neo4j.com/docs/getting-started/data-import/csv-import/



Obrázek 1: Návrh Neo4j grafové databáze pro jízdní řády IDS JMK ve formátu GTFS.

```
lon: toFloat(csv.stop_lon), zone_id: csv.zone_id, location_type:
toInteger(csv.location_type), parent_station: csv.parent_station,
wheelchair_boarding: toInteger(csv.wheelchair_boarding),
platform_code: csv.platform_code});

load csv with headers from
'file:///stops.txt' as csv
with csv
where not (csv.parent_station is null)
match (ps:Stop {id: csv.parent_station}), (s:Stop {id: csv.stop_id})
create (ps)-[:PARENT_OF]->(s);

load csv with headers from
'file:///routes.txt' as csv
with csv
where not (csv.agency_id is null)
match (a:Agency {id: csv.agency_id})
create (a)-[:OPERATES]->(r:Route {id: csv.route_id, short_name: csv.route_short_name,
long_name: csv.route_long_name, type: toInteger(csv.route_type), color: csv.route_color,
text_color: csv.route_text_color});

load csv with headers from
'file:///calendar.txt' as csv
create (c:Calendar {id: csv.service_id, monday: toInteger(csv.monday), tuesday:
toInteger(csv.tuesday), wednesday: toInteger(csv.wednesday), thursday:
toInteger(csv.thursday), friday: toInteger(csv.friday), saturday:
toInteger(csv.saturday), sunday: toInteger(csv.sunday), start_date:
csv.start_date, end_date: csv.end_date});

load csv with headers from
```

```
'file:///calendar_dates.txt' as csv
match (c:Calendar {id: csv.service_id})
create (c)<-[:CHANGES]-(cd:CalendarDate {date: csv.date, exception_type:
toInteger(csv.exception_type)});

load csv with headers from
'file:///trips.txt' as csv
match (r:Route {id: csv.route_id}), (c:Calendar {id: csv.service_id})
create (r)<-[:ON]-(t:Trip {id: csv.trip_id, headsign: csv.trip_headsign,
wheelchair_accessible: toInteger(csv.wheelchair_accessible), block_id: csv.block_id,
direction_id: csv.direction_id, bikes_allowed: toInteger(csv.bikes_allowed)}),
(t)-[:AVAILABLE_ON]->(c);


load csv with headers from
'file:///stop_times.txt' as line
call {
 with line
 match (t:Trip {id: line.trip_id}), (s:Stop {id: line.stop_id})
 create (t)-[:INCLUDES]->(st:StopTime {arrival_time: line.arrival_time,
 departure_time: line.departure_time, stop_sequence: toInteger(line.stop_sequence),
 pickup_type: toInteger(line.pickup_type), drop_off_type: toInteger(line.drop_off_type),
 shape_dist_traveled: toFloat(line.shape_dist_traveled), timepoint:
 toInteger(line.timepoint)}),
 (st)-[:LOCATED_AT]->(s)
} in transactions of 100000 rows;

load csv with headers from
'file:///transfers.txt' as csv
match (s1:Stop {id: csv.from_stop_id}), (s2:Stop {id: csv.to_stop_id})
create (s1)-[:CONNECTS_TO {transfer_type: toInteger(csv.transfer_type),
min_transfer_time: toInteger(csv.min_transfer_time), from_trip_id: csv.from_trip_id,
to_trip_id: csv.to_trip_id}]->(s2)
with csv
where not (csv.from_trip_id is null) and not (csv.to_trip_id is null)
create (t1:Trip {id: csv.from_trip_id})-[:CONNECTS_TO]->(t2:Trip {id: csv.to_trip_id});
```

Vložení nové jízdy, případně úprava již existující, na základě ID jízdy:

```
merge (t:Trip {id: "36068"})
 on create set
  t.headsign = 'Semilasso',
  t.wheelchair_accessible = 0,
  t.direction_id = "1",
  t.bikes_allowed = 0
 on match set
  t.headline = 'Semilasso';
```

Dotaz na jízdy spoje linky č. 12, obsahující ID jízdy, název trasy, název destinace a dostupnost dané jízdy v neděli:

```
match (r:Route)<-[o]-(t:Trip)-[a]->(c:Calendar)
 where r.short_name = "12"
 return t.id as ID, r.long_name as route, t.headsign, c.sunday as sunday;
```

Dotaz je oproti použití např. relační databáze velmi efektivní, neboť vyhledá vztahy, které jsou uloženy stejným způsobem, jako samotné uzly.

4 Databáze časových řad

URL: Kvalita ovzduší
Zdroj dat: Magistrát města Brna

Pro reprezentaci databáze časových řad byl vybrán produkt influxdata InfluxDB, demonstrovaný na datové sadě "Kvalita ovzduší". Datová sada obsahuje naměřená data kvality ovzduší z různých stanic. Každá stanice sbírá data vícekrát do dne. Kvalita ovzduší je určena pomocí více veličin, které každá stanice měří. V datové sadě jsou také údaje o každé stanici. Data obsahují adresu, vlastníka a geografické souřadnice stanice. Tuto datovou sadu je vhodné uložit do databáze InfluxDB, jelikož obsahuje data ze senzorů měnící se v době, na což je tento typ databáze určen. Pokud databázi správně navrhneme a určíme tagy, bude možné v databázi snadno vyhledávat hodnoty kvality ovzduší jednotlivých stanic v daném časovém úseku. Kromě toho InfluxDB nabízí rychlé čtení a zápis dat, což je klíčové pro aplikace, které potřebují rychle přistupovat k čerstvým datům nebo provádět analýzu historických dat. InfluxDB také umožňuje definovat retenční politiky, které určují, jak dlouho se data uchovávají. Tato funkce je užitečná pro data s různými úrovněmi důležitosti a archivace. InfluxDB je součástí ekosystému nástrojů a knihoven, které usnadňují analýzu a vizualizaci dat. Uživatelská aplikace postavená na této databázi by mohla být snadno použita k analýze změny kvality ovzduší v čase.

InfluxDB je navržena tak, aby byla snadno rozšiřitelná a mohla zpracovávat velké objemy dat. To je pro naši datovou sadu vhodné, jelikož existuje mnoho stanic pro měření kvality ovzduší. Pokud bude každá stanice vyhodnocovat kvalitu ovzduší víckrát denně, objem dat velmi rychle naroste. Také můžeme předpokládat, že aplikace, která bude pracovat s těmito daty, bude chtít zobrazovat poměrně velký počet těchto dat v různých grafech. Kvůli těmto specifikům je databáze InfluxDB vhodným řešením pro zvolenou datovou sadu. [10] [11]

Definice úložiště, import dat, dotazování

Vytvoření databáze:

```
CREATE DATABASE airQuality

USE airQuality
```

Import dat: Importování dat přímo z formátu csv v tomto případě nebude možné. Proto jsme připravili jednoduchý Python skript, který data z csv načítá, upraví do požadovaného tvaru a zapíše do databáze.

```
import pandas as pd
from influxdb import InfluxDBClient
from datetime import datetime

# Definice parametru na pripojeni se k databazi
csv_file = 'kvalita_ovzdusia.csv'
```

[10] https://docs.influxdata.com/influxdb/v1/
[11] https://blog.ippon.tech/a-beginners-guide-to-influxdb-a-time-series-database/

```
influxdb_host = 'localhost'
influxdb_port = 8086
database_name = 'airQuality'
measurement_name = 'airStations'

client = InfluxDBClient(host=influxdb_host, port=influxdb_port)

data = pd.read_csv(csv_file)

# Konvertovani csv do InfluxDB formatu
points = []
for index, row in data.iterrows():
    tags = {
        "code": row["code"],
        "name": row["name"],
        "owner": row["owner"],
        "globalid": row["globalid"]
    }

    fields = {
        "X": row["X"],
        "Y": row["Y"],
        "objectid": row["objectid"],
        "lat": row["lat"],
        "lon": row["lon"],
        "so2_1h": row["so2_1h"],
        "no2_1h": row["no2_1h"],
        "co_8h": row["co_8h"],
        "pm10_1h": row["pm10_1h"],
        "o3_1h": row["o3_1h"],
        "pm10_24h": row["pm10_24h"],
        "pm2_5_1h": row["pm2_5_1h"]
    }

    timestamp = row["actualized"]
    if timestamp:
        timestamp = datetime.strptime(timestamp, "%Y/%m/%d %H:%M:%S+00")

    point = {
        "measurement": measurement_name,
        "tags": tags,
        "time": timestamp,
        "fields": fields
    }
    points.append(point)

# Zapis dat do InfluxDB
client.switch_database(database_name)
client.write_points(points)
```

```
client.close()
```

Jako tagy dat jsme zvolili název, vlastníka, globalID a kód dané stanice. To nám umožní indexovat data z každé stanice zvlášť.
Dotaz na všechna měření pro jednu stanici v určitém časovém intervalu:

```
SELECT * FROM airStations
WHERE name = 'Brno - Dětská nemocnice'
  AND time >= '2023-01-01T00:00:00Z'
  AND time <= '2023-02-01T00:00:00Z'
```

Dotaz na měření všech stanic v daném časovém intervalu:

```
SELECT name, no2_1h FROM airStations
WHERE time >= '2023-01-01T00:00:00Z'
  AND time <= '2023-02-01T00:00:00Z'
```

Databáze je navržena tak, aby se dalo velmi jednoduše vyhodnocovat data z jednotlivých stanic v čase. V dotazech jsme ukázali, že lze snadno vyhledávat data v různých časových rozmezích. InfluxDB samozřejmě podporuje i složitější možnosti filtrace dat, například dle limitních hodnot měření.