ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ

FAKULTA ELEKTROTECHNICKÁ

Zpřístupnění technických síťových podkladů využitím cloudových služeb

Accessing technical network data sources via cloud services

Bakalářská práce

Studijní program: Elektronika a komunikace

Vedoucí práce: Ing. Robert Bešťák Ph.D

Luka Jovanović

Praha 2020

Čestné prohlášení

Tímto prohlašuji, že jsem zadanou práci vypracoval samostatně pod vedením vedoucího práce a že jsem uvedl veškeré použité zdroje v souladu s metodickým pokynem o dodržování etických principů při tvorbě vysokoškolských prací.

Datum: 22.5.2020 ……..…

Luka Jovanović

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Jovanović** Jméno: **Luka** Osobní číslo: **473459**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra telekomunikační techniky**

Studijní program: **Elektronika a komunikace**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Zpřístupnění technických síťových podkladů prostřednictvím cloudových služeb

Název bakalářské práce anglicky:

Accessing Technical Network Data Sources via Cloud Services

Pokyny pro vypracování:

Cílem práce je odzkoušet a vyhodnotit metody zpřístupnění technických informací o mobilní síti s využitím uživatelsky dostupných nástrojů cloudových služeb AWS. Jedná se zejména o mapy pokrytí mobilním signálem, polohu a konfiguraci antén v návaznosti na předpokládané pokrytí a intenzitu využití buněk.

Seznam doporučené literatury:

[1] Sauter, M.: From GSM to LTE-Advanced Pro and 5G: An Introduction to Mobile Networks and Mobile Broadband. Wiley. 2017. ISBN: 978-1-119-34686-9.
[2] Walke, B.H.: Mobile Radio Networks: Networking, Protocols and Traffic Performance, 2nd Edition. Wiley. 2002. ISBN: 978-0-471-49902-2.
[3] Okabe, A.; Boots, B.; Sugihara, K; Chiu, S.N.: Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, 2nd Edition. Wiley. 2000. ISBN: 978-0-471-98635-5.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Robert Bešťák, Ph.D., katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **07.01.2020** Termín odevzdání bakalářské práce: **22.05.2020**

Platnost zadání bakalářské práce: **30.09.2021**

Ing. Robert Bešťák, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

Datum převzetí zadání	Podpis studenta

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu práce Ing. Robertovi Bešťákovi, Ph.D za ochotu, trpělivost, pomoc a rady a při zpracovávání této bakalářské práce.

Abstrakt

Tato práce se zabývá návrhem struktury Amazon DynamoDB databáze pro ukládání dat o pokrytí mobilní sítí. Jsou v ní popsány základní principy mobilních sítí, databází a nástrojů ke zpracování geografických dat. Dále je uvedeno několik konkrétních návrhů struktury v DynamoDB. Dva návrhy jsou empiricky testovány a výsledky jsou zpracovány do podoby tabulek a grafů. V závěru práce jsou uvedeny analýzy testů a alternativní řešení.

Klíčová slova

GIS, Databáze, Mobilní síť, Data, Amazon DynamoDB, Geodata

Abstract

The aim of this bachelor thesis is designing an Amazon DynamoDB structure for storing data from the coverage of mobile networks. The first part consists of basic principles of mobile networks, databases in general and tools for handling geographical data. The thesis contains several designs of the structure in DynamoDB. Two designs were selected and tested. The results of these tests are processed into a form of tables and graphs. An analysis of the tests and alternative solutions are located at the end of the thesis.

Key words

GIS, Database, Mobile network, Data, Amazon DynamoDB, Geodata

OBSAH

Seznam obrázků

Seznam tabulek

Seznam zkratek

GIS	Geographic Information System
FDMA	Frequency Division Multiple Access
TDMA	Time Division Multiple Access
CDMA	Code Division Multiple Access
OFDM	Orthogonal Frequency Division Multiplexing
BTS	Base Transciever Station
MS	Mobile Station
GPRS	General Packet Radio Services
LAN	Local Area Network
XML	eXtensible Markup Language
DBMS	Database Management System
PK	Partition Key
SK	Sort Key
SQL	Structured Query Language
AWS	Amazon Web Services

1 Úvod

V dnešní době je téměř každé zařízení připojeno do určité sítě. Je zřejmé, že směřujeme k absolutně bezdrátovému světu technologií a mobilní sítě jsou nezbytnou součástí této vize. Veškeré funkce, jež každé zařízení v síti zprostředkovává, jsou zaznamenány do podoby dat, ukládány do databází či uložišť a zpracovány v souladu s architekturou sítě, zatímco požadavky na rychlosti zpracování stále rostou. Tempo, jakým se toto odvětví technologie vyvíjí a množství požadavků, jež jsou kladeny na zařízení implikuje fakt – s postupem času bude stále růst množství dat, které je potřeba zpracovávat či ukládat.

Tato práce se zabývá návrhem struktury databáze, sloužící pro zpracování a ukládání dat z mobilní sítě, která pokrývá území České republiky, využitím cloudových technologií *Amazon Web Services* (AWS). První část práce se skládá z teoretického úvodu do problematiky mobilních sítí. V úvodu je vysvětlen princip buňkových sítí a všech dosavadních generací mobilních sítí. Dále jsou vysvětleny základy geografických dat a nástrojů, pomocí kterých jsou tato data zpracována, konkrétně principy geografického informačního systému (GIS). V druhé části je uveden jak obecný princip databází, tak i princip využité databáze od firmy Amazon s názvem *DynamoDB* (Dynamo databáze). Dále jsou uvedeny navržené struktury pro zadanou problematiku. Byly vybrány dvě struktury, které jsou v závěru práce empiricky testovány s účelem zjištění, zda se jedná o efektivní navržení. Testování je prováděno tak, že je zkoumána vždy jedna entita (obec, buňka pokrytí) a pomocí dotazů je získáván počet překryvů (počet buněk v obci/počet obcí v buňce). V závěru se nachází výsledky analýz, zhodnocení a alternativní řešení.

2 Mobilní sítě

Za posledních pár let zažívá bezdrátová industrie obrovský rozmach. Bezdrátové technologie širokého využití, rostoucí počet tzv. *user-friendly* zařízení a přístup veškerým multimédiím způsobuje požadavky na efektivní navržení sítě. Krom rádiových komponent obsahují mobilní sítě i dedikovanou infrastrukturu, která umožňuje zpřístupnění právě zmíněným službám. Každá nová generace mobilní sítě tak přináší novou, vylepšenou službu (v době, kdy byla vynalezena) a zároveň jsou kladeny větší požadavky na další generace.

2.1 Generace mobilních sítí

Veškeré generace mobilních sítí přináší velké množství služeb, které přeformovaly způsob naší komunikace a sdílení dat. Každá nová generace přinesla technologický vývoj a tento vývoj pokračuje dodnes.

2.1.1 První generace

Úplně první generace mobilní sítě je téměř nerozpoznatelná při pohledu na služby, které se dnes denně využívají. První generace sítí využívala technologie analogového broadcastu, který pracuje v pásmu 800MHz a využívá 30kHz kanály, které jsou rozděleny pomocí frekvenčního dělení (*Frequency Division Multiple Access*, *FDMA*). Mezi standardy patří *Nordic Mobile Telephone* (NMT), *Advanced Mobile Telephone System* (AMTS) a *Total access Communication System* (TACS). Tato generace zprostředkovala pouze hovorové služby [1].

2.1.2 Druhá generace

Druhá generace se dala považovat za zlomovou. Zatímco 1G systémy byly využívány jen určitým počtem lidí, tzv. *Global System for Mobile Communications* (GSM) se dodnes využívá na celém světě. GSM je nejrozšířenějším standardem v dnešní době. Funguje na principu buňkové sítě (viz. sekce 2.2). V České republice je využito dvou frekvencí technologie GSM uvedených v tabulce 1 a to konkrétně 900 MHz a 1800 MHz.

Jako přístupovou metodu se v GSM používá kombinace frekvenčního dělení (FDMA) a časového dělení (*Time Division Multiple Access,* TDMA) [2][3].

Mezi 2. a 3. generací existují ještě generace 2.5 a 2.75. Jedná se o *General Packet Radio Services* (GPRS), což je nadstavba GSM umožňující datový přenos. Technologie *Enchanced Data rates for GSM Evolution* (EDGE) přináší vylepšení datových přenosových vlastností [2][3].

2.1.3 Třetí generace

Cílem 3. generace bylo poskytnutí vyšších přenosových rychlostí a umožnění dalších multimediálních služeb. Jedná se o přímé vylepšení GSM. *Universal Mobile Telecommunications System* (UMTS) je technologie využívající pásma 2GHz, což je frekvence, na které jsou zachovány dobré přenosové vlastnosti, je zaručen nízký útlum signálu a dobrá penetrace do venkovních i vnitřních prostorů. Příděl pásem závisí na využité technologii a na směru signálu. UMTS využívá časový duplex (TDD) a frekvenční duplex (FDD), přičemž směr je buď vzestupný nebo sestupný. Vzestupný směr (*uplink)* je směr od uživatele, zatímco sestupný (*downlink*) je směr k uživateli. V České republice je technologie UMTS provozována konkrétně na frekvenci 2100MHz [3][4].

UMTS má širší pásmo při UMTS FDD, jelikož tato technologie je upřednostňována. Využitím kódového dělení (*Code Division Multiple Access,* CDMA) je zaručena větší kvalita služeb, a proto je možno využít tuto technologii na sdílení multimédií (obrázky, videa) či k jiným službám [1][3][6].

Přístupové metody použité v GSM technologiích nejsou pro UMTS dostatečné, a proto je využito širokopásmového kódového dělení (*Wideband Code Division Multiple Access,* WCDMA). Jedná se o kódové dělení, které způsobuje dělení do širokých pásem, způsobující zlepšení vlastnosti kapacity a přenosové rychlosti.

Navazující, tzv. 3.5 generací je *High-speed Downlink Packet Access* (HSDPA), což je technologie, vylepšující parametry ve směru k uživateli jak pro UMTS FDD, tak i UMTS TDD. Analogicky existuje i 3.75 generace *High-speed Uplink Packet Access* (HSUPA), což je rovněž vylepšení technologií, ale ve směru od uživatele.

2.1.4 Čtvrtá generace

Čtvrtá generace mobilních sítí představuje pokročilou verzi 3. generace. Jedná se o přímý vývoj technologie UMTS nazývaný *Long Term Evolution* (LTE), někdy označován jako 3.95G. LTE, na rozdíl od UMTS, ve směru k uživateli využívá ortogonální frekvenční dělení (*Orthogonal Frequency Division Multiplexing,* OFDM). Ve směru od uživatele není již využito WCDMA nýbrž *Single Carrier – Frequency Division Multiple Access* (SC-FDMA). Jelikož LTE není oficiálně považováno za technologii 4. generace, existuje *Long Term Evolution Advanced* (LTE-A). Ta již splňuje normy mezinárodní telekomunikační unie ITU a přináší vylepšení například v podobě slučování nosných frekvencí a mnoho dalších. V České republice je technologie 4G (LTE, LTE-A) provozována na frekvencích 800, 900,1800,2100,2600MHz, přičemž dvě frekvence jsou stále využívány primárně staršími technologiemi a to UMTS (2100MHz) a GSM (900MHz) [2][3][4].

Pomocí těchto technologií je možno např. pořádat videokonference, sdílet 3D animace a využívat veškeré audio služby. Primární cíl této generace je však globalizace. Tato technologie má za úkol poskytnout globální mobilitu, přenosnost služeb a škálování za snížené ceny [1].

V tabulce 1 je uvedeno shrnutí všech generací a technologií, zmíněných v této kapitole. Protože se generace 1G dnes již prakticky nepoužívá, není v tabulce uvedena. Jsou v ní uvedeny veškeré standardy pro danou generaci a frekvence, na kterých jsou technologie provozovány v České republice [4].

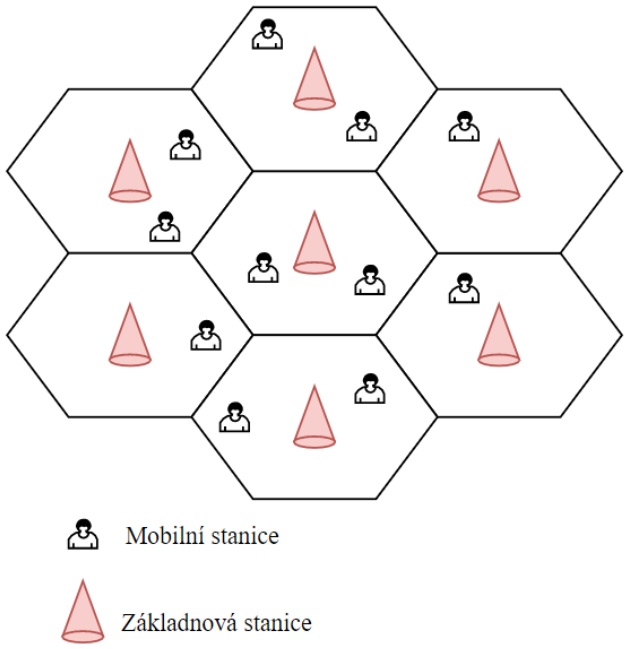
Tabulka 1 - Přehled generací mobilních sítí.

Generace	Technologie	Frekvence [MHz]
2G	GSM, GPRS, EDGE	900,1800
3G	UMTS, HSPDA, HSPUA	2100
4G	LTE, LTE-A	800,900,1800,2100,2600

Na těchto frekvencích jsou provozovány i technologie shromažďující data, která jsou v mé práci použita. Vymezení sady použitých dat je uvedeno v kapitole 5.

2.2 Struktura buňkové sítě

Princip buňkové sítě je založen na rozdělení území pokrytého mobilní sítí na dílčí buňky různých velikostí. Velikosti buněk se liší na základě toho, jakou funkci mají splňovat. Jsou-li seřazeny podle velikosti sestupně, jedná se o satelitní buňky, makrobuňky, mikrobuňky, pikobuňky či femtobuňky. Jednoduché schéma buňkové sítě je uvedeno na obrázku 1. Síť je založena na základnových stanicích (*Base Transciever Station,* BTS), což je struktura obsahující vysílací/přijímací antény a na mobilních stanicích (*Mobile Station,* MS), což jsou zařízení připojená k síti. Na obrázku 1 je ilustrován princip buňkové sítě. Každá buňka má vlastní základnovou stanici a vyskytují se v ní pohybující se/stacionární zařízení připojená k síti. V principu platí, že větší buňky mají na starost pohybující se MS (auto na silnici), zatímco menší buňky mají na starost spíše stacionární MS (prostředí kanceláře či školy) [2][3].

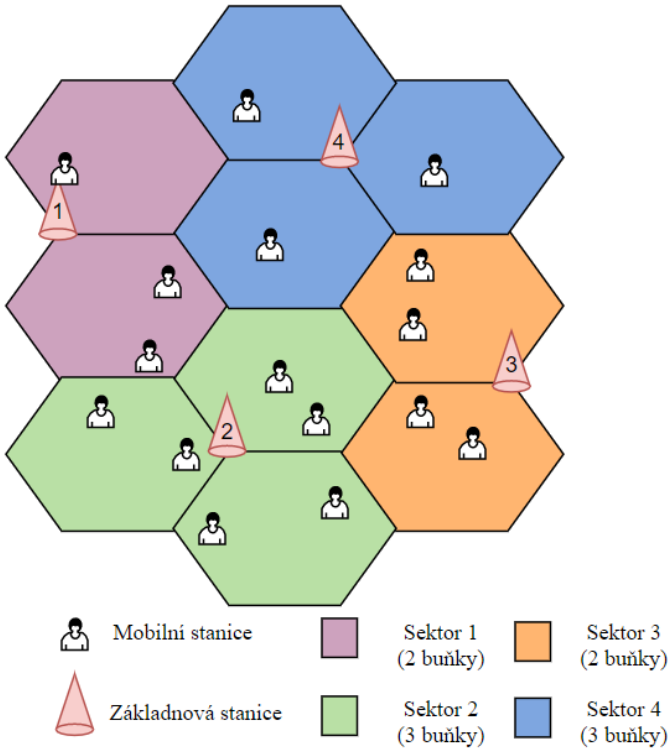


Obrázek 1 – Struktura buňkové mobilní sítě.

Při rozsáhlé buňkové síti by struktura z obrázku 1 znamenala velké množství BTS. K tomu existuje takzvaná sektorizace. Princip spočívá v tom, že místo toho, aby byla stanovena jedna BTS na každou buňku, navrhne se síť takovým způsobem, aby jedna základnová stanice pokrývala několik sousedících buněk. Tato skupina buněk je označována za *sektor,* přičemž sektor může tvořit libovolný počet buněk. Pokrytí

sektoru spočívá v tom, že na BTS jsou umístěny antény příslušných technologii (viz. sekce 2.1.4 tabulka 1) a pokrývá několik buněk ve stejnou chvíli.

Princip sektorizace je znázorněn na obrázku 2. Pro názornou ukázku, mějme sektor 1, sektor 2, sektor 3 a sektor 4. Každý ze sektorů se skládá ze tří nebo ze dvou buněk sítě. Máme tedy čtyři sektory, které můžou být pokryty GSM, UMTS či LTE technologiemi na frekvencích, uvedených v sekci 2.1.4 v tabulce 1. Je vhodné podotknout, že na jedné základnové stanici, lze mít více antén různých technologií, tudíž neplatí, že v jednom sektoru (ani v buňce) je jednotná technologie.



Obrázek 2 - Princip sektorizace.

Sektorizace je vhodná především k úspoře základnových stanic. Při identifikaci, která se řídí podle systému, který je uveden v sekci 5.2, je nutno specifikovat lokalitu základnové stanice, typ vysílače (antény), využitou technologii (včetně použité frekvence) a daný sektor (buňku), kterou pokrývá.

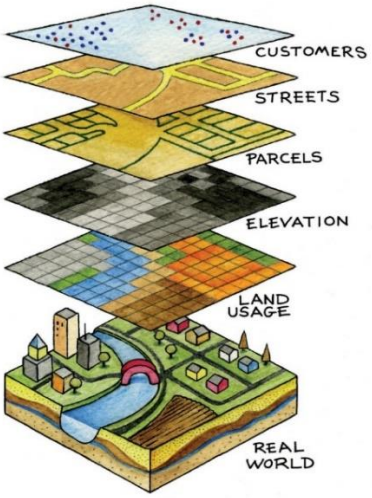
3 Geografický informační systém

Náš svět konstantně prochází změnami. Ať už se jedná o změny přírodní, nebo ty, způsobené člověkem, nezůstává nic stejné delší dobu. Poháněni touhou po znalostech se tyto změny zaznamenávají a dále zkoumají. Jedním z nástrojů ke zaznamenání dat je tzv. Geografický Informační Systém (GIS). Ten lze definovat jako počítačově založený systém, který poskytuje následující aplikace při práci s daty, která závisí na geografických parametrech:

- zaznamenávání a příprava dat,
- řízení dat, včetně ukládání a spravování,
- manipulace a analýza dat,
- vizualizace či jiná prezentace dat.

GIS jako celek se skládá z hardware komponent, na kterých je zprovozněn příslušný software, obsahující veškeré potřebné nástroje. V minulosti byly využívány analogové zdroje dat, kde zpracování a vizualizace byla prováděna manuálně. GIS umožňuje tato data ukládat v digitální formě použitím souřadnic. Škála využití GIS je z pohledu geografických dat prakticky neomezená. Využívá se téměř při každé geografické změně či analýze, ať už se jedná o plánování cest, či jiných pozemních komunikací, evidenci nemovitostí nebo například zkoumání tlakových hladin. V této práci je analyzováno pokrytí území mobilní sítí, přičemž je konkrétně zkoumána vazba mezi geografickými daty území (obce) a mapou pokrytí mobilní sítí různých technologií [7][8].

Na obrázku 3 je ilustrována škála využití GIS. Příkladem je jeho využití k mapování určité oblasti. Je znázorněno použití rastrových i vektorových reprezentací dat (sekce 3.4.1 a 3.4.2) na principu mapování území (rastr), nadmořské výšky (rastr), jednotlivých parcel nemovitostí (vektor), silnic (vektor) či osob (vektor).



Obrázek 3 – GIS [9].

3.1 Princip

3.1.1 Geodata

Jak bylo zmíněno, GIS využívá a pracuje s určitými daty, která jsou vázána k vlastnostem Země a jsou označována za tzv. geodata. Tato data reprezentují reálný svět a je nutno je od sebe odlišovat. Z toho důvodu má každý záznam hlavní složky:

- geometrická – tvar a umístění objektu,
- atributy – specifické vlastnosti objektu,
- vztah vůči jiným objektům,
- změny objektu v čase.

Krom těchto parametrů je velmi důležité tyto objekty rozměrově rozlišovat, jelikož není dána jednotná dimenze. Lze tedy objekty z tohoto hlediska členit následovně:

Dimenze 0D je dimenze, u které nelze měřit žádný rozměr – jedná se o údaje reprezentované **bodem**.

Dimenze 1D je dimenze, u které je možno měřit pouze jeden parametr – délku. Jedná se tedy o data reprezentována **linií**.

Dimenze 2D je dimenze, kde jsou reprezentovány objekty, u nichž lze měřit dva rozměry, a to délku ve dvou směrech, tj. jedná se o **plochy**.

Dimenze 3D je nejvyšší dimenzí, objekty jsou zde určeny třemi rozměry, a to délkou a plochou – jedná se o **tělesa**.

Při rekonstrukci objektů v GIS samozřejmě dochází do určité míry ke zkreslení, a tudíž žádný z objektů ve finále není shodný s reálným světem [8].

3.1.2 Metadata

Krom zmíněných dat existují také tzv. metadata. V praxi je lze označit za „data o datech“. Jinými slovy se jedná o informace, popisující veškeré nezbytné parametry o samotných datech, a to:

- identifikační informace – zdroj dat, čas získání atd.,
- kvalitu obdržených dat – přesnost atributů, pozice, času,
- informace o atributech a entitách – např. jednotky měření.

Metadata jsou klíčová pro údržbu kvality a informací o datech v GIS. Je to z toho důvodu, že mohou sloužit nejen jako popis dat, ale zároveň i jako návod na práci s daty. Jsou důležitým parametrem při výběru sady dat, jelikož jsou využívány při dotazech typu „kdo“, „co“, „kde“, „kdy“ atd. Příkladem je, chceme-li získat sadu dat, která splňuje určité požadavky (existence v určité oblasti) [7].

3.2 Datové modely

Máme-li sadu dat, je možno s nimi manipulovat mnoha způsoby. Častým požadavkem GIS je vizualizace dat. Jak bylo zmíněno v sekci 3.1.1, rozlišujeme data podle dimenzí, na základě kterých, je třeba zvolit datový model, jenž bude využit k rekonstrukci těchto dat.

Při volbě modelu je nutno brát v potaz, o jaký objekt se jedná, jelikož tyto modely neposkytují ekvivalentní efektivitu při rekonstrukci. K tomu máme k dispozici modely vektorové a rastrové. Oba modely mají své výhody i nevýhody, tudíž nelze říci, že nezáleží na volbě modelu. Například vektorová reprezentace je výhodnější při tvorbě mapy libovolných komunikací neboli dat, jež lze reprezentovat linií či bodem, zatímco rastrová reprezentace je lepší například při tvorbě mapy srážek. I přesto lze mezi sebou tato data převádět s možným vznikem chyb či nepřesností.

3.2.1 Vektorová reprezentace

Tato reprezentace využívá souřadnicového systému k reprezentaci objektů, to znamená, že bod je reprezentován jako souřadnice [x,y], linie je dána intervalem souřadnic [x,y] jdoucích za sebou a plocha je definována uzavřenou posloupností linií (označováno

jako polygon). Zvláštním případem jsou tzv. *multipoints, multilinestrings a multipolygons*, což jsou množiny vícero prvků, které nemají společný bod (jedná se například o jakoukoliv pozemní komunikaci, která je v určitém úseku přerušena). Jedním z hlavních standardů pro ukládání dat je tzv. *Shapefile*. Ten spočívá v tom, že každý prvek stejného charakteru (např. veškerá elektrická vedení) se ukládají do databáze se stejným názvem, ale jinou příponou. Tento zápis je složen z několika souborů [8][10]:

- hlavní soubor – záznam, popsaný pomocí daného souřadnicového systému,
- indexový soubor – poukazuje na místa v Shapefile souboru, odkud se příslušná data mají načítat (indexace),
- databázový soubor – tabulka osahující jednotlivé atributy prvků,
- doplňkové soubory – další informace o datech, použitý souřadnicový systém, prostorový index.

Tento standard má několik omezení (uložení do více souborů, omezení počtu znaků atributů atd.), ale fakt, že je kompatibilní se všemi softwary je důvodem jeho využití dodnes. Jiné vektorové formáty jsou rozebrány v sekci 3.4.1. Pomocí těchto reprezentací lze rekonstruovat především objekty tzv. diskrétního charakteru. Jedná se například o určité významné body, hranice všech typů (státní hranice, hranice krajů), průběhy elektrického vedení či například hranice jezer nebo rybníků.

Důležitým pojmem je tzv. topologie. Topologie je efektivní způsob vyjadřování vztahů mezi jednotlivými objekty v prostorovém smyslu. Využívá se především v analýzách, kdy není nutno využívat souřadnic daných objektů k řešení problému. Příkladem je výpočet optimální trasy mezi dvěma body. Použitím seznamu známých spojů mezi těmito body a výpočtem ceny výstavby je možno získat ideální trasu. Souřadnice jsou tedy využity teprve při tvorbě takto získané trasy. Pro tento výpočet se používají tři základní koncepty topologie [10]:

- konektivita,
- sousednost,
- definice plochy (jedná-li se o plošný výpočet).

3.2.2 Rastrová reprezentace

Rastrová reprezentace se vyznačuje tím, že veškeré prvky jsou vyjádřeny v určitých buňkách (pixelech). Hojně se využívá pro reprezentaci plošných jevů, které se spojitě

či nespojitě mění. Jednotlivé buňky mohou mít stejnou, pravidelnou velikost a tvar nebo nepravidelnou velikost a tvar. Dané buňky mohou mít čtvercový, trojúhelníkový či hexagonální tvar. Celková reprezentace je tedy matice buněk neboli přesněji pixelů, nejčastěji čtvercového tvaru, z důvodu kompatibility s datovými strukturami programovacích jazyků používaných v GIS software. Rozměry jednotlivých buněk (tj. délka strany čtverce) udává rozlišení výsledné mapy, tudíž čím menší strana čtverce, tím větší je rozlišení. Každá buňka mřížky nese v sobě určitý druh informace, která je reprezentována číselně a jejíž hodnota se konkrétním způsobem projeví ve vizualizované formě. Příkladem je frekvence výskytu určitého objektu v mřížce. Pro každou buňku je evidována číselně hojnost výskytu a ta je reprezentována např. intenzitou barvy v mapě na výstupu. Rastrová reprezentace se využívá především pro jevy spojitého charakteru jako je například nadmořská výška, mapa srážek či využití mobilní sítě v určité lokalitě (obci, kraji) [8][10].

3.3 Převod dat

Jak bylo uvedeno, jednotlivé reprezentace jsou výhodnější pro různé typy analýz. GIS systémy umožňují převod mezi těmito reprezentacemi tzv. vektorizaci či rasterizaci.

3.3.1 Rasterizace

Rasterizace je převod z vektorového na rastrovou reprezentaci. Je nutné stanovit velikost buňky rastru optimálním způsobem, aby se do ní "vešly" veškeré informace, a zároveň, aby nezabírala zbytečný prostor, který zatěžuje hardware. Poté se vykoná prakticky překryv vektorové vrstvy na rastrovou mřížku. Jelikož rastrová buňka má pouze jeden atribut, nastává problém ve chvíli, kdy buňka má v sobě více objektů. Pro převod na rastrový formát dat lze využít jednoho z následujících způsobů [11]:

Metoda dominantního typu spočívá v tom, že v buňce, kde nastala kolize objektů se vybere ten, který má hojnější zastoupení (velikost obsazené plochy buňky) a ten je buňce přiřazen [11].

Metoda nejdůležitějšího typu je vykonána po evaluaci, kdy se buňce přiřadí objekt, který je z hlediska dané analýzy nejdůležitější [11].

Centroidová metoda je založena na průmětu středu buňky do vektorové reprezentace, který stanoví, jaký objekt bude přiřazen [11].

3.3.2 Vektorizace

Tento proces je komplikovanější než rasterizace a to z toho důvodu, že je nutno vytvořit vektorové objekty ze spojité rastrové podoby. K tomu je využito několik metod:

Ruční metoda je manuální tvorba do vektorové podoby. Výhodou tohoto přístupu je malý dopad na hardware a software, jedná se o nenáročný způsob. Avšak značnou nevýhodou je doba trvání převodu, jelikož je vše děláno ručně [12].

Poloautomatická metoda je „spolupráce“ administrátora a systému, kde osoba zvolí začátek rastrové linie, posléze systém navrhuje směr vektorizace, který musí administrátor schválit a vykonává vektorizaci do doby, kdy narazí na problém/překážku. V tu chvíli je nutné schválení od administrátora, aby se pokračovalo v operaci [12].

Automatická metoda probíhá tak, že systém vykonává převod samostatně, kdy k tomu využívá umělou inteligenci a známé algoritmy zpracování obrazu [12].

3.4 Datové formáty

Veškerá geodata mají určitou formu, ve které jsou uložena. Každý z formátů nese jisté výhody v některých aplikacích, a proto existuje rozdělení formátů na vektorové a rastrové.

3.4.1 Vektorové formáty

Existuje několik vektorových formátů, přičemž každý z nich má specifické vlastnosti. Mezi ně patří například a) Shapefile b) KML c) GML d) GeoJSON e) OGC GeoPackage f) GDB.

a) **Shapefile** je jedním z nejčastěji používaných formátů. Skládá se z několika souborů, ve kterých jsou uložena vektorová data (především jednoduššího charakteru), jak je zmíněno v sekci 3.2.1. Sice se v dnešní době jedná o méně používaný formát, ovšem jeho výhodou je kompatibilita se všemi programy či nástroji, které GIS nabízí [13].

b) ***Keyhole Markup Language*** **(KML)** je datový formát sloužící především k práci s geodaty. Je psán v jazyce *eXtensive Markup Language* (XML) a obsahuje vektorové formáty všech dimenzí, pomocí kterých identifikuje místa, či například tvoří překryv textury [10].

c) ***Geograpgy Markup Language*** **(GML)** je rovněž formát psaný v jazyce XML, který slouží k zobrazení geografických prvků. Skládá se z několika částí, které tvoří celek (objekt, v praxi například silnice, mosty apod.), který pak lze využívat v různých aplikacích v zakódované podobě [14].

d) **GeoJSON** je formát geodat, založený na objektové notaci JavaScript (JSON), tj strukturovaný takovým způsobem, že jsou textem zapsané atributy daného objektu. Tento formát nese informace a vlastnosti určitých geografických objektů a je schopen definovat prakticky veškeré dimenze vektorové reprezentace (bod, linie, polygon, multipoint, multilinestring, multipolygon). Výhodou tohoto formátu je jeho jednoduchost a velikost – jedná se o nenáročný formát na zpracování, což způsobuje jeho efektivní využití v oblasti webových prohlížečů. Využívá souřadnicového systému a lze manuálně manipulovat s atributy záznamu [15].

Příkladem GeoJSON-u, který popisuje nějaký bod na mapě je uveden na obrázku 4.

```
{
"type":"Feature",
  "geometry": {
        "type":"Point",
        "coordinates": [133.1,22.3]
        },
         "properties":{
               "title": "SmallVillage"
               }
        }
```

Obrázek 4 - Příklad GeoJSON.

e) **OGC GeoPackage** je datový formát, implementovaný jako druh SQL databáze (viz. sekce 4.1). To znamená, že poskytuje možnost ukládat vektorová i rastrová data, všechny atributy a některá rozšíření. GeoPackage standard stanovuje veškerá pravidla a omezení pro ukládání dat (definice tabulek, formátová omezení, doporučený obsah GeoPackage apod.). Jelikož se jedná o tzv. *database container*, lze přímo přistupovat datům a lze je upravovat bez překladů. [16].

f) ***GeoDatabease*** **(GDB)** je alternativní způsob, jak ukládat geodata do jednoho většího souboru, který může obsahovat několik vrstev vektorových či rastrových zápisů. Jedná se o výhodnější způsob ukládání dat v porovnání s formou několika Shapefile souborů,

protože poskytuje přehlednější uspořádání při velkém množství dat (data stejného charakteru či problému tvoří jednu databázi). Na rozdíl od Shapefile nabízí GDB možnost tvořit vnitřní topologii [17].

3.4.2 Rastrové formáty

Rastrové formáty mají širokou škálu využití a velké množství lidí je používá na denní bázi. Mezi ně patří například a) GeoTIFF b) JPEG c) GIF d) PNG.

a) **GeoTIFF** je, de facto, veřejná doména skládající se z tzv. metadat, což jsou v praxi data, nesoucí v sobě informace o jiných datech (viz. sekce 3.1.2). Specifická jsou v tom, že umožňují ukládat a přistupovat geodatům, která jsou uložena v souboru typu *Tagged Image File Format* (TIFF), který slouží k ukládání rastrových dat. Informace uložené v metadatech jsou především souřadnicové systémy, projekce map, elipsoidy, čas uložení dat a jiné [8].

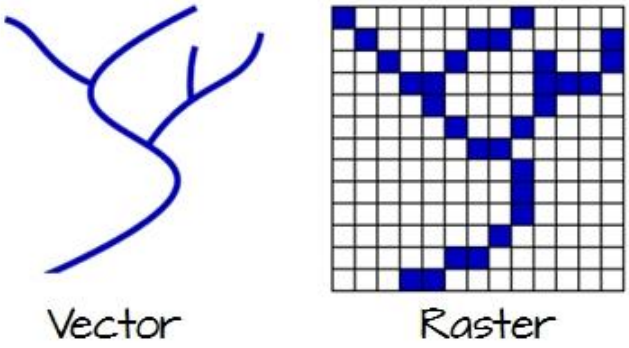
b) ***Joint Photographic Experts Group*** **(JPEG)** je velmi běžný formát používaný zejména například u digitálních zařízení. Podporuje různé úrovně komprese a 16 miliónů barev. Ty je JPEG schopen produkovat a jsou tvořeny 8 bity pro každou barvu v RGB spektru [18].

c) ***Graphics Interchange*** **Format (GIF)** je častý formát používaný pro webové nebo animované obrázky. Využívá bezeztrátovou kompresi, a proto nedochází k poškození kvality výstupu. GIF ukládá obrazová data v indexové podobě barev, což znamená, že je velikostně omezen na 256 barev. Z toho důvodu se nejedná o vhodný formát pro ukládání digitálních fotografií [20].

d) **PNG** je formát, který rovněž využívá bezeztrátovou kompresí. Na rozdíl od JPEG či GIF formátů, má podporu tzv. alfa kanálu neboli barevného prostoru RGBA. Byl vytvořen za účelem vylepšení a záměny za GIF, jelikož je běžné, aby PNG formát byl o 10-30 % více komprimován, což má za výsledek menší velikost souboru a lepší barevnou reprodukci [21].

3.4.3 Srovnání formátů

Je uvedeno několik formátů obou reprezentací, které lze mezi sebou jednoduše porovnat. Je zmíněno, že každý formát má své výhody i nevýhody, stejně jako účel využití. Obrázek 5 znázorňuje rozdíl v reprezentaci určitého objektu.



Obrázek 5 - Vektorová a rastrová reprezentace [19].

Tabulka 2 obsahuje porovnání vektorové a rastrové reprezentace.

Tabulka 2 - Srovnání rastrové a vektorové reprezentace.

	Vektor	**Rastr**
Struktura dat	body, linie, polygony	pixely
Velikost dat	malá	velká
Geometrická přesnost	vysoká	nízká
Převod (rasterizace/vektorizace)	jednoduchá	složitější
Využití	+ mapy, tratě - nevhodné pro obrázky	+ obrázky - nevhodné pro křivky

3.4.4 Zapisování dat

Důležitým nástrojem v zapisování dat je *Well-known Text* (WKT), což je prakticky způsob zaznamenávání vektorových objektů na mapách. Je úzce spojen s tzv. *Well-known binary* (WKB), který je prakticky totožný, akorát využívá 1-byte *unsigned integer*, 4-byte *unsigned integer* a 8-byte *floating point* čísla k zápisu dat. Pomocí WKT lze zapsat veškeré vektorové objekty od bodu po polygony či multipolygony. Syntaxe spočívá v tom, že se stanoví, o jaký objekt se jedná (POINT, LINESTRING, POLYGON) a určí se jeho rozsah pomocí souřadnic [22].

Jednoduchý příklad WKT je uveden na obrázku 6. Jedná se o definování každého druhu vektorového objektu pomocí souřadnic.

```
POINT (45,15)

LINESTRING (30 10, 30 20, 50 10)

POLYGON ((45 10, 35 15, 40 15, 50 10, 45 10))
```

Obrázek 6 - Příklad WKT.

4 Databáze

Obdržená data je nutno skladovat takovým způsobem, aby byla zachována přehlednost, nedošlo ke ztrátě informací a aby data byla připravena k opakovaném použití. Databáze jako taková je pojem, označující skupinu informací, primárně orientovanou kolem stanovené problematiky. Tato definice slouží k separaci databází od skupiny nesouvisejících informací. V praxi je-li nutno projít každou informaci za účelem nalezení jedné konkrétní, nejedná se o databázi [23].

Každý druh databáze má určitou strukturu. Ve valné většině se jedná o formu tabulky, obsahující entity, které nesou informaci a jsou jednoznačně definované. Každá entita může mít libovolný počet atributů (stanoveno tvůrcem, či administrátorem). Tyto atributy slouží ke specifikaci datových entit. Právě to zapříčiňuje organizovaný charakter, jelikož se pomocí těchto atribut zápisy mohou řadit, filtrovat či získávat. Jednoduchý příklad databáze znázorňuje tabulka 3.

Tabulka 3 - Příklad databáze.

ID_prod	ID_zák	Adresa	Cena	Stav
A123	JN.457	Nová 42	6990	Doručeno
B123	AL.787	Celá 74	8850	Doručeno
C123	OK.789	Dlouhá 3	1239	Zpracovávání
D123	SN.661	Malá 48	999	Na cestě
E123	LU.452	U Kraje 7	599	Zrušeno

Tabulka ilustruje jednoduchou databázi, která je využita například v internetových obchodech. Obsahuje zápisy, které reprezentují fiktivní objednávky zboží. Jednotlivé entity jsou uvedeny v řádcích a atributy představují sloupce. Z takovéto struktury lze jednoduše filtrovat data při tvorbě dotazů na databázi. Pokud bychom chtěli například získat pouze entity, představující vyřízené objednávky, využijeme k tomu filtrování pomocí parametru *stav*.

Obsahuje-li databáze pouze jednu tabulku s prvky, jedná se o tzv. *flat-file* databázi. Tabulka 3 představuje příklad takové databáze – veškeré vlastnosti o primárním parametru (v tomto případě ID produktu) jsou uloženy na jednom místě. Avšak často

jsou požadavky (a množství dat) velmi obsáhlé a tento typ databází může být nepraktický a nepřehledný [23].

V tom případě lze takové tabulky rozdělit do několika menších, které jsou navzájem vázané a mají alespoň jeden společný parametr. Takový druh databází se nazývá *relational databases* neboli relační databáze. Využijeme uvedený příklad pro názornou ukázku. Rozdělíme tabulku na dvě menší, kdy jedna bude obsahovat pouze identifikátory a adresy kupců, zatímco druhá bude obsahovat cenu a status objednávky. Obě tabulky budou navzájem vázané primárním parametrem – ID produktu, což je v tabulce 4 zvýrazněno zelenou barvou.

Tabulka 4 - Rozdělení databáze.

ID_prod	Cena	Stav
A123	6990	Doručeno
B123	8850	Doručeno
C123	1239	Zpracovávání
D123	999	Na cestě
E123	599	Zrušeno

ID_prod	ID_zák	Adresa
A123	JN.457	Nová 42
B123	AL.787	Celá 74
C123	OK.789	Dlouhá 3
D123	SN.661	Malá 48
E123	LU.452	U Kraje 7

Toto je jednoduchá ilustrace principu relačních databází, v praxi jsou však tyto vazby většinou komplexnější (může být vázáno více parametry, vázaných více tabulek).

Existuje veliké množství databází, dělících se podle funkce, aplikace či principu. Pro pochopení je třeba si vymezit pojem *Database Management System* (DBMS). Ten slouží ke tvorbě vhodného prostředí pro práci s databází. DBMS vytváří podporu pro jednoho i více uživatelů. Mezi funkce DBMS mimo jiné patří [23][24][25]:

- skladování, získávání a aktualizování dat,
- kontrola simultánního využívání – schopnost produkovat chybné hlášení v situaci, kdy se snaží více uživatelů manipulovat s daty ve stejnou chvíli,
- zotavení při kolapsu systému,
- bezpečností omezení.

Přestože se databáze jeví jako efektivní a velmi praktický nástroj pro práci s daty, má také určitá negativa. Jedním z nich je cena provozování databází. Jelikož může být množství dat obrovské, je k jeho zpracování potřebné mít výkonný hardware a software a rovněž tak i osoby, které mají bohaté zkušenosti s údržbou databází. Takovéto databáze jsou totiž na údržbu a řízení komplexní, obzvlášť v situacích, kdy je nutno aktualizovat velké množství informací v krátkých intervalech [23][24][25].

4.1 SQL Databáze

Již bylo zmíněno, že existuje systém sloužící k práci s databázemi. Tento systém je ovládán určitými příkazy ve formě jazyka a jedním z nich je právě *Structured query language*(SQL). SQL je specifický jazyk pro manipulaci s daty, uloženými ve zmíněných relačních databázích. Slouží k vložení, vyhledání, aktualizaci či vymazání záznamu z databáze. Krom toho slouží i k optimalizaci a údržbě databází. Konkrétní příkaz *SQL query* nebo dotazu je uveden v sekci 7.2. A právě databázím, které jsou ovládané prostřednictvím jazyka SQL se říká SQL databáze [26].

4.2 NoSQL databáze

Analogicky existuje i tzv. NoSQL jazyk (No = *not only*), což je rozšiřující se kategorie jazyku používaného v databázích. Jedná se o koncept, který vytváří odchylku od relačních databází. Hlavním rozdílem mezi SQL a NoSQL databází je takzvané schéma. Zatímco SQL jsou databáze „tabulkové“ orientace a mají striktně danou strukturu a průběh veškerých funkcí uvnitř databází, NoSQL databáze mají variabilní strukturu. Lze vytvořit NoSQL databáze se schématem, které vyhovuje dané sadě dat. S tím je spojena i možnost tzv. škálování, a to konkrétně horizontálního, což v praxi znamená, že lze vkládat neomezené množství dat, aniž by to výrazně ovlivnilo strukturu databáze. SQL databáze takovou možnost nemají, jelikož při vkládání nových záznamů, mohou nastat potíže a vznik požadavku na změnu schématu [27].

NoSQL se vyvíjel především u technologických gigantů dnešní doby, kterými jsou Google, Amazon či Facebook a to proto, že tyto firmy musí zpracovávat ohromná množství dat. Pokud by je zpracovávali použitím SQL jazyku, nastalo by velké množství operací a systém by znatelně zpomalil [27].

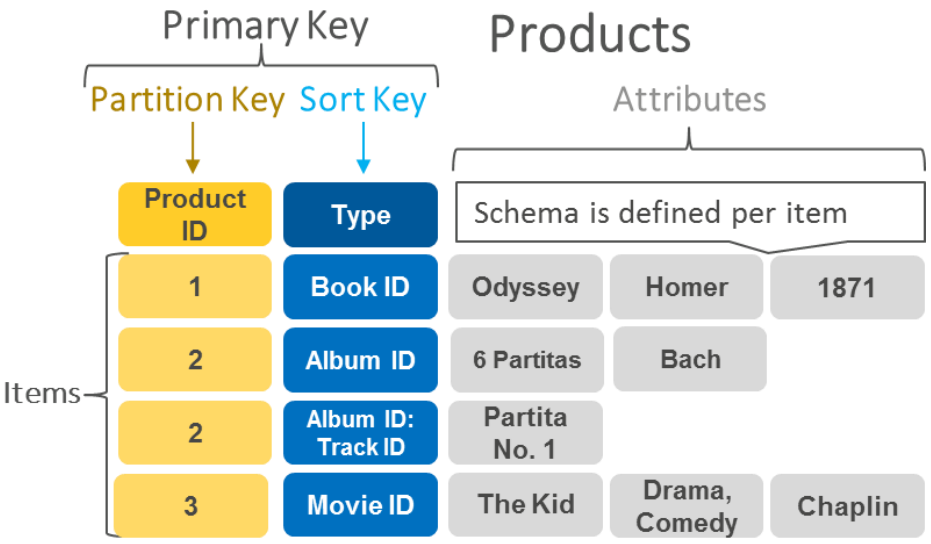
Jedním z řešení je vylepšení hardware daných systémů, aby byly schopny zpracovávat rychleji data. To by do určité míry proces zrychlilo, ale jedná se o řešení finančně náročné a především krátkodobé, jelikož dříve nebo později by bylo nutno rychlosti zpracování opět zvyšovat s rostoucími nároky. Optimálnějším řešením je tzv. *scaling out*, což je prakticky rozdělení práce na několik zařízení. A právě NoSQL databáze mají lepší schopnost škálování než klasické relační databáze. NoSQL databáze nepoužívají SQL pro "tázání" na určitá data a nedodržují striktní schéma funkcí na rozdíl od relačních databází, což způsobuje určité nevýhody. Díky těmto vlastnostem nelze zaručit tzv. ACID (*atomicity, consistency, isolation, durability*). Jinými slovy, nelze garantovat nedělitelnost operací, důslednost, izolaci nebo separaci a odolnost [27].

NoSQL databáze jsou především specifickým nástrojem, který se aplikuje hlavně v situacích, je-li nutno zpracovat velká množství dat, v jiných případech je lepší používat klasické *relation databases* [27].

4.3 Dynamo databáze

Databáze, která byla zvolena pro tuto práci se nazývá DynamoDB. Jedná se o specifický druh databáze, kterou nabízí firma Amazon prostřednictvím svých webových služeb AWS. Vyznačuje se především tím, že se jedná o tzv. *key-value* databázi tj. databázi definovanou pomocí klíčových hodnot. Ta spočívá na principu ukládání dat do entit, které jsou jednoznačně určeny jedním nebo dvěma „klíči“, podle kterých lze jednotlivé zápisy identifikovat. Tyto hodnoty mohou být jak triviální objekty, tak i velmi komplexní objekty. Jak je vyobrazeno na obrázku 7, každý zápis je určen pomocí tzv. *Primary key* (primární klíč). Existují dva typy primárního klíče – jednoduchý klíč, který se skládá pouze z tzv. *Partition key* (PK). V tomto případě je požadováno, aby každý zápis měl unikátní PK. Druhým způsobem, vyobrazeném na obrázku 7, je tzv. *Composite key* (složený klíč), který se skládá z již zmíněného PK a řadícího klíče *Sort key* (SK).

V tomto případě PK nemusí být unikátní – lze mít více zápisů se stejným PK, avšak SK musí být odlišný. Zjednodušeně, každý zápis musí mít unikátní dvojici PK a SK. Dodatečné atributy další omezení nemají, jedná se jen o parametry entity [29][30].



Obrázek 7 - Struktura DynamoDB [28].

4.3.1 Hlavní výhody

Využívání dynamo databází nese několik zásadních výhod. Lze je rozdělit do tří základních částí:

- výkon a rychlost operací v poměru na množství dat,
- virtuální podoba (tj bez fyzických serverů),
- aplikovatelnost a zabezpečení.

Výkonnost je úzce spjata s již zmíněným pojmem škálování, který nabízí flexibilnější schéma funkcionality, což je výhodné pro situace, kdy může například dojít ke změně požadavků na účel databáze. Krom toho také dynamo databáze od Amazonu využívají nástrojů AWS, které efektivně pracují s pamětí a výrazně zvyšují výkon databází (až 10x). To se jeví jako vhodné, především při používání dynamo databází v aplikacích, kde je nutno zachycovat a vykonávat změny v době, kdy probíhají [29].

Pro příklad, u určité aplikace, využívané v globálním měřítku, dojde ke změně dat v některé z tabulek databáze, která je z určitého regionu. V jiném regionu je tato změna zaznamenána a je automaticky vytvořena kopie této tabulky a zároveň je na ni vázána takovým způsobem, že se sama aktualizuje, dojde-li ke změně v původní tabulce. Tudíž aktualizace a "spolupráce" probíhá v reálném čase a automaticky. Krom toho, pro Dynamo databázi není nutno spravovat nebo dohlížet na servery či jiný software,

protože škálování, tolerance chyb a podobné vlastnosti jsou zabudované a není je nutno jako administrátor kontrolovat [29][30].

Flexibilita těchto databází může vypudit nejistoty v oblasti bezpečnosti či zabezpečení dat. To je řešeno šifrováním uživatelských dat, kdy jsou veškerá uživatelská data šifrována automaticky a zároveň lze tvořit aplikace se striktními požadavky na bezpečnost. Krom toho lze zálohovat i načítat data až do velikosti několika stovek *terabyte,* aniž by došlo k ovlivnění výkonu [29][30].

Naopak nevýhodou Dynamo databází je fakt, že je lze provozovat pouze prostřednictvím AWS. Dalším nedostatkem by se dalo považovat dotazování integrované v databázi přímo, avšak existují jiné nástroje (AWS AppSync, viz. 9.1), pomocí kterých lze tento problém minimalizovat.

5 Data

Má práce je součástí většího projektu, probíhajícího v rámci univerzity. Používám v ní data, která jsem obdržel při zapojení se do projektu. Tato kapitola pouze definuje způsob identifikace záznamů v datech, žádná data nejsou nově vytvářena. Data se skládají ze sady dvou databází, obsahující geodata o území ČR a pokrytí mobilní sítí všech technologii uvedených v sekci 2.1, které jsou provozovány na frekvencích, uvedených v tabulce 1 v sekci 2.1.4. Jedná se o databázi s informacemi o obcích ČR a databázi s pokrytím mobilní sítí. V následujících kapitolách jsou tyto databáze vysvětleny, stejně jako princip identifikátorů a parametrů jednotlivých záznamů.

5.1 Databáze obcí

Tato databáze obsahuje záznamy o všech obcích ČR. Parametry, které jsou pro moji práci důležité jsou identifikátory obce a její polygon ve formátu *JSON* (sekce 3.4.1, formát *d*)).

5.1.1 Klasifikace území ČR

Každé území je určitým způsobem klasifikováno. Mezinárodní klasifikace územních jednotek nese název Nomenklatura územních statistických jednotek (NUTS). Pomocí této klasifikace, jsou definovány územní oblasti od států, či regionů až po kraje. V České republice se tato klasifikace označuje za CZ-NUTS, jelikož klasifikuje pouze území ČR. Součástí NUTS je i menší soustava s názvem *Local Administrative Units* (LAU), která slouží k identifikaci okresů, obcí či základních sídelních jednotek. Tabulka 5 představuje, jaký typ klasifikačního standardu CZ-NUTS zodpovídá za dané území [32].

V mé práci je zkoumáno pouze území typu obec, tudíž je důležitá klasifikace LAU 2, která je v tabulce 5 zvýrazněna. V databázi s obcemi je každý záznam o obci jedinečným způsobem identifikován. Veškeré záznamy obcí, které jsem v práci použil mají zmíněný identifikátor a polygon, který obsahuje geometrické údaje o obci samotné. V databázi neexistují dva záznamy, které by popisovaly stejnou obec.

Tabulka 5 - NUTS v ČR[5].

	Význam	Počet území v ČR
NUTS 0	stát	1
NUTS 1	území (ČR)	1
NUTS 2	regiony	8
NUTS 3	kraj	14
LAU 1	okres	76 + Praha
LAU 2	obec	6253
LAU 3	základní sídelní jednotka	22606

Zavedený systém identifikátoru obce v této databázi je složen z předpony *la2*, značící fakt, že se jedná o obec a jedinečné kombinace čísel, definující konkrétní obec. Předpona *la2* je jedinečná pro tento systém, ale čísla za ní jsou identifikátory obcí podle ČSÚ. Příkladem identifikátoru obce je:

la2545848

Tento parametr eviduji do databází ve své práci do sloupců s názvem *PK*. V databázi s obcemi jsem dále použil parametr s názvem *pol* obsahující informaci o polygonu obce ve formátu *JSON*. Tento parametr je využit jak v PostGIS databázi (kapitola 7), tak i v DynamoDB (kapitola 8). Celkový příklad záznamu obce je uveden v tabulce 6.

Tabulka 6 – Záznam obce v databázi.

PK	pol
la2545848	JSON

Sloupec *PK* značí primární klíč (viz. sekce 4.3), pomocí kterého jsou tato data později implementována do databáze DynamoDB (viz. kapitola 8).

5.2 Databáze pokrytí

Druhou použitou databází je databáze s pokrytím ČR mobilní sítí. Obsahuje záznamy o buňkách pokrytí všemi technologiemi (viz. sekce 2.1.4, tabulka 1). Tyto záznamy mají také jedinečné identifikátory, ze kterých lze určit vlastnosti buňky pokrytí.

Vytvořený systém identifikace, který byl v databázi zaveden a je použit v mé práci je založen na následujícím principu. Identifikátor je složen z devítimístného řetězce formátu *AAAbbbCDEE*. Lze z něj vyčíst lokalita buňky (*AAAbbb*), velikost buňky a použitá technologie (*C*), provozovaná frekvence (*D*), popřípadě odlišení od jiných záznamů (*EE*). Příkladem identifikátoru buňky pokrytí je:

KVRacr11CA

Z tohoto identifikátoru lze vyčíst, že se jedná o 2G, makro/mikro buňku na frekvenci 900 MHz s vysílačem v Karlových Varech.

Identifikátor buňky pokrytí je rovněž uváděn do sloupce *PK*, z důvodu efektivní implementace do DynamoDB. Stejně jako u obcí, jsem i u pokrytí použil polygony buněk pokrytí, které mají opět stejný formát *JSON* v PostGIS (kapitola 7) i DynamoDB databáze (kapitola 8). Příklad záznamu o pokrytí mobilní sítí je v tabulce 7. Sloupec *PK* opět značí primární klíč, pomocí kterého je definována databáze DynamoDB.

Tabulka 7 – Záznam o pokrytí v databázi.

PK	pol
KVRacr11CA	JSON

Krom zmíněných dvou databází byla zpřístupněna i databáze s údaji o anténách, ale ta není pro moji práci zásadní.

6 Implementace

Jak bylo zmíněno v úvodu práce, GIS je nástroj pro reprezentaci geografických dat, který lze efektivně využívat v oblasti mobilních sítí. V kapitole 3 je uvedeno, že v GIS můžeme vizualizovat veškerá geodata, tudíž je intuitivní zobrazit pokrytí území mobilní sítí ve formě mapy. I když je v GIS (a PostGIS, viz. kapitola 7) možno skladovat velké množství dat bez omezení, nastává problém ve chvíli, kdy tuto databází používá větší množství uživatelů najednou. V tomto aspektu nejsou tyto nástroje ideální, a proto je vhodné využít jiných nástrojů pro skladování a GIS pouze jako vizualizační prostředek, jelikož jiné databáze tento problém řeší a zároveň jsou navrženy tak, aby efektivně zpracovávaly velké množství dat. Z toho důvodu je třeba převést veškerá data do jiné databáze. Jelikož se množství datových objektů mění rychle (a ve velkém počtu) databáze Dynamo od Amazonu se nabízí jako možné řešení. Cílem této práce je tato data do DynamoDB vhodným způsobem převést a otestovat některé dotazy, které mohou být na data pokládány. Problematikou je fakt, že tato databáze není primárně navržena na geodata (umí ukládat jen obecný JSON, viz. sekce 3.4.1) a nelze data jednoduše přesunout. Je nutné zavést vazbu mezi geografickými atributy a definicí tabulky tak, aby byla v mezích možností DynamaDB a zároveň nijak neomezovala dotazování.

Důvodem ke zvolení konkrétně DynamoDB pro tuto práci je fakt, že nad zkoumanými daty bude v budoucnu tvořena webová aplikace, u které dochází k variabilnímu počtu uživatelů. Krom toho je nutno zajistit, aby zdroj dat byl škálovatelný a tato kritéria DynamoDB splňuje.

7 PostGIS

Struktura databáze či její ideální aplikace, není u všech databází stejná. Jak je zmíněno v sekcích 4.1 a 4.2, lze dělit databáze na několik typů. Krom zmíněných SQL a NoSQL databází existují např. objektově-orientované, hierarchické a jiné. PostgreSQL, což je databáze, ve které jsou často uložena geodata, je typ objektově relační. Ten se v jednoduchosti vyznačuje tím, že veškeré informace jsou reprezentovány formou objektů, se kterými se pracuje pomocí jazyku SQL. Jednou z jeho výhod, krom toho že se jedná o tzv. *open source* program, je možnost rozšíření. Jedním z těchto rozšíření je tzv. PostGIS. Toto rozšíření s sebou nese podporu geografických objektů a možnost SQL dotazů na tyto objekty [31].

Pro tuto práci je tento nástroj nezbytný, jelikož slouží k lokaci překryvů geografických údajů a map pokrytí mobilní sítí. Ke zpracování byl využit program QGIS. V něm je možnost připojit se ke vzdálené databázi PostGIS. Je nutno vytvořit bezpečné spojení se sítí prostřednictvím *Virtual Private Network* (VPN), což je typ připojení, který se vymezuje bezpečným komunikačním kanálem. Veškerý provoz uvnitř kanálu je kódovaný a zdroje jsou sdíleny exkluzivně mezi autorizovanou skupinou uživatelů. Připojení se realizuje tímto způsobem, jelikož se jedná o citlivá data a v klasickém připojení přes veřejnou síť lze data odchytávat [32].

7.1 Rozlišení využitím barev

V práci se vyskytuje několik záznamů, které se často liší jen minimálně, i když představují odlišné věci. Proto je zavedena konvence barev, sloužící k odlišení typu záznamu, či odlišení toho, co daný parametr představuje. Identifikace záznamů obcí a pokrytí sítí jsou v souladu se sekcemi 5.1 a 5.2. Konvence barevného odlišení bude využita v sekcích 7.2, 8.1, 8.2, 8.3, 8.4 a 8.5.

Je-li v záznamu parametr *PK* či *SK* znázorněn zelenou barvou  ,jedná se o identifikátor obce.

Je-li v záznamu parametr *pol* znázorněn zeleným písmem, jedná se o *JSON*, který přísluší obci. Příklad je v tabulce 8. Ve sloupci *PK* je uveden identifikátor obce a ve sloupci *pol* se nachází její polygon.

Tabulka 8 - Odlišení obce.

PK	*pol*
la2575828	JSON

V tabulce 8 je tedy záznam o obci s identifikátorem *la2575828* a její polygon.

Je-li v záznamu parametr *PK* či *SK* znázorněn modrou barvou ,jedná se o identifikátor buňky pokrytí, který je vytvořen podle sekce 5.2.

Je-li v záznamu parametr *pol* znázorněn modrým písmem, jedná se o *JSON*, jež přísluší buňce uvedené v záznamu. Příklad je ilustrován v tabulce 9.

Tabulka 9 - Odlišení pokrytí.

PK	*pol*
KVRacr11CA	JSON

V tabulce 9 je záznam o pokrytí sítí s identifikátorem *KVRacr11CA* a polygon buňky s tímto identifikátorem.

7.2 Příklad dotazu na pokrytí v PostGIS

V kapitole 7 je vysvětlen princip fungování databáze v PostGIS zatímco v sekcích 5.1 a 5.2 jsou uvedeny použité datové zdroje. Je důležité podotknout, že databáze o pokrytí obsahuje buňky všech technologií, tudíž záznamy získané dotazováním mohou být různých technologií (sekce 5.2). Jinými slovy, položením libovolného dotazu nezískám vždy záznamy pokrytí jednou technologií.

Zjednodušený příklad s reálnými daty o pokrytí mobilní sítí je uveden v tabulce 10. Barevné odlišení záznamů se řídí konvencí, která je vymezena v sekci 7.1. Tabulka je ukázkou databáze se záznamy o pokrytí mobilní sítí, přičemž v tabulce 10 je uvedeno jen několik záznamů (v realitě je jich mnohem více). Obsahuje identifikátor pokrytí v *PK* a *JSON* s polygonem pokrytí v *pol*.

Tabulka 10 - Zápisy buněk pokrytí v PostGIS.

PK	pol
SOKabj11AA	JSON
KVRacr11CA	JSON
JIHabs31AA	JSON

Stejně tak je součástí databáze tabulka se záznamy obcí, která je strukturována podobným způsobem. Zjednodušený příklad s obcemi je uveden v tabulce 11. Ta má definované také dva parametry: jedinečný identifikátor obce (vytvořený na základě sekce 5.1.1) v *PK* a *JSON* s polygonem obce.

Tabulka 11 - Zápisy buněk obcí v PostGIS.

PK	pol
la2545848	JSON
la2554782	JSON
la2587486	JSON

Jelikož PostGIS podporuje jazyk SQL, můžeme ho využít k tvorbě dotazů. Protože cílem analýzy je utvořit vazbu mezi pokrytím a geografickými lokalitami, využijeme následující SQL příkaz:

*SELECT * FROM Tabulka 10 n, Tabulka 11 a, WHERE a.PK = 'la2587486' AND ST_intersects(n.pol,a.pol);*

Vybíráme tedy všechny parametry z obou tabulek (vyjádřeno znakem'*') se specifikací *PK* jednoho záznamu. V tomto případě jsme využili tabulky obcí a omezili jsme dotazování pouze na obec s uvedeným *PK* (*la2587486*). Příkaz *ST_intersects*, vrací pouze takové záznamy, jejichž polygony mají alespoň jeden společný bod (vyjma hranic) a splňují stanovené podmínky. Pomocí tohoto příkazu jsme obdrželi veškeré buňky všech technologií pokrytí z databáze *Tabulka 10*, které zasahují do obce s identifikátorem *la2587486*.

Reálným výsledkem dotazu v prostředí QGIS je 16 záznamů. To znamená, že do zkoumané obce zasahuje 16 buněk pokrytí (různých technologií). Jak již bylo zmíněno, GIS je efektivním nástrojem vizualizace geodat. Pro názorný příklad bude stačit, pokud si zobrazíme jeden ze záznamů z výsledku dotazování, který se nachází i v tabulce 10.

Byl zvolen záznam o pokrytí, který má identifikátor ve tvaru *JIHabs31AA*. Využijeme příkaz, který bude ve tvaru:

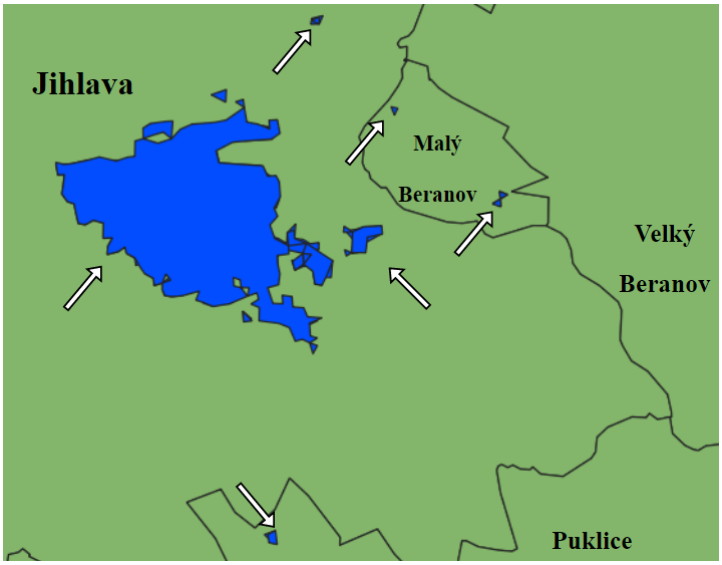
SELECT PK,pol from Tabulka 10 WHERE PK = 'JIHabs31AA';

Výsledkem dotazu je následující záznam:

JIHabs31AA	JSON

Z něj, využitím definice uvedené v sekci 5.2, lze zjistit, o jakou technologii se jedná, kde se buňka nachází a další informace.

V QGIS je provedena vizualizace uvedeného dotazu. Nejprve je načtena mapa obcí ČR jako jedna vrstva (zelené pozadí). Záznam získaný z dotazu načteme do prostředí jako další, oddělenou vrstvu, abychom docílili překryvu těchto vrstev. Na obrázku 8 je uveden výsledek v prostředí QGIS. Buňka pokrytí je vyznačena modře. Bílé šipky poukazují na jev, že tento záznam nemá jednotný polygon, tj skládá se z několika, různě velkých polygonů. To může být způsobeno například strukturou terénu v této oblasti (hory, výšiny, nížiny), které mají rušivé efekty. Tyto bílé šipky ukazují na veškeré části záznamu o pokrytí s identifikátorem *JIHabs31AA*.

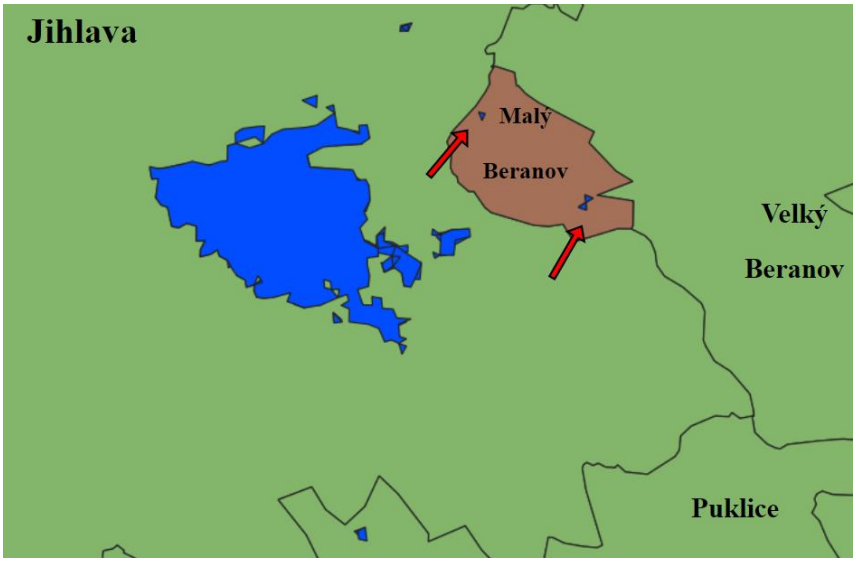


Legenda: Záznam pokrytí, Území ČR

Obrázek 8 - Polygon zkoumané buňky pokrytí.

Na obrázku 8 je tedy vizualizován polygon pokrytí buňky s identifikátorem *JIHabs31AA*. Jelikož původní příkaz byl zadán za účelem zjištění, které záznamy o

pokrytí zasahují do obce s identifikátorem *la2587486*, je vhodné si tuto obec vizualizovat jako samostatnou vrstvu. Zkoumaná obec je zvýrazněna hnědou barvou na obrázku 9. V obrázku 8 ukazovaly bílé šipky na veškeré části polygonu pokrytí. Na obrázku 9 vidíme červené šipky, které poukazují pouze na ty části polygonu pokrytí, které zasahují do zkoumané obce s identifikátorem *la2587486*.



Legenda: Záznam pokrytí, Zkoumaná obec, Území ČR

Obrázek 9 - Polygony buňky pokrytí a obce.

7.3 Převod dat do DynamoDB

PostGIS je sice nástroj, který se pro geodata jeví jako praktický, ale pro velké množství uživatelů, pracujících s daty ve stejnou chvíli, je tato databáze nevhodná pro skladování (viz. kapitola 6). PostGIS databáze je sice schopna bez větších obtíží ukládat velké množství dat, avšak u DynamoDB je toto číslo ještě větší. Zároveň DynamoDB splňuje požadavky na škálování, jak již bylo zmíněno v sekci 4.3. Tato databáze nebyla navržena ke zpracování geodat a tudíž implementace není zdaleka tak jednoduchá jako u PostGIS. Je třeba si stanovit, jaké dotazy by měla databáze, v rámci této práce, být schopna vykonávat. V budoucnu lze očekávat rozšíření množiny dotazů. Ze začátku se bude jednat především o následující dotazy:

- zobrazit veškeré buňky, které zasahují do vybrané obce,
- zobrazit veškeré obce, do kterých daná buňka zasahuje.

Dodatečně lze využít navržená řešení i k zobrazení buněk v obcích s daným filtrem (neboli specifikací, například o jakou technologii se jedná, typ antény, velikost apod).

Je zřejmé, že je nutné navrhnout strukturu, která je "obousměrná", tj. jedná se o vazbu, která má flexibilní parametr, u kterého je možné se dotazovat na lokalitu i na pokrytí, aniž by se zbytečně čerpalo úložiště či zpomalovala rychlost operací.

K převodu je třeba nejprve manuálně vytvořit několik zápisů, kvůli stanovení struktury. Poté využijeme nástroje v rámci AWS (AWS Lambda, *Batch insert*), pomocí kterého jsme schopni do definované struktury nahrát velké množství dat prostřednictvím skriptu.

8 Navržená řešení

V této sekci je vysvětlena konkrétní implementace struktur. Jak je zmíněno v sekci 4.3, tabulky obsahující tzv. *Composite primary key*, mohou mít shodné *PK* (ale odlišné *SK*) pro více zápisů.

Všechna uvedená řešení jsou zkoumána na triviálním příkladu, využitím dostupných tabulek. V reálném použití by výsledky všech uvedených dotazů byly výrazně rozsáhlejší a početnější, princip funkce návrhu však zůstává zachován.

Uvedené tabulky a záznamy se řídí konvencí ze sekce 7.1, nebo je u aplikace přímo uveden smysl a vysvětlení struktury nového záznamu. Zároveň jsou veškeré návrhy určené pomocí *PK* a *SK*, jejichž princip je vysvětlen v sekci 4.3.

Dvě z uvedených struktur (sekce 8.1 a 8.5) jsou později testovány a analyzovány v kapitole 9.

8.1 Duplikáty polygonů

Jak je uvedeno v kapitole 6, podstatou je strukturovat tabulku v DynamoDB tak, aby mezi daty pokrytí a obcí byla utvořena „obousměrná“ vazba. Navržená struktura by měla efektivně zpracovávat dotazy na data, uvedené v sekci 7.3. Prvním řešením je vytvořit dva druhy záznamů.

První typ záznamů představuje obec, který však má v sobě polygon ve formátu *JSON* jedné z buněk pokrytí, která do ní zasahuje. *PK* je vytvořen pomocí identifikátoru obce (sekce 5.1.1). Ve svém *SK* obsahuje identifikátor záznamu pokrytí mobilní sítí, který do ní zasahuje. Záznam obce tedy vypadá následovně:

Tabulka 12 – Záznam obce.

PK	SK	pol
la2540323	BNSaad11CA	JSON

Záznam indikuje tedy, že do obce *la2540323* zasahuje buňka *BNSaad11CA*. V databázi se může vyskytnout více záznamů se stejným *PK* (ale odlišným *SK*), záleží to na počtu záznamů mobilní sítě, které do daných obcí zasahují.

Analogicky jsou vytvořeny záznamy o pokrytí sítí. Od záznamů obcí se liší tím, že mají prakticky prohozené hodnoty v *PK* a *SK* a obsahují polygony obcí, které pokrývají. *PK* je vytvořen pomocí identifikátoru pokrytí. Jsou tedy ve formátu:

Tabulka 13 - Záznam pokrytí.

PK	SK	pol
BNSaad11CA	la2540323	JSON

Záznam nese informaci, že buňka *BNSaad11CA* zasahuje do obce *la2540323*. V tabulce 14 je předvedena názorná ukázka struktury, obsahující několik záznamů obou uvedených typů. Je v ní zároveň ilustrován jev, kdy existují záznamy se stejným *PK*, ale odlišným *SK*.

Tabulka 14 – Duplikáty polygonů.

PK	SK	pol
la2540323	BNSaad11CA	JSON
la2540323	PBRaci51AA	JSON
la2577421	JBNaao52AA	JSON
la2575828	CHRaaj11AA	JSON
BNSaad11CA	la2540323	JSON
JBNaao52AA	la2577421	JSON
CHRaaj11AA	la2575828	JSON
CHRaaj11AA	la2547816	JSON

Princip spočívá v tom, že záznamy obcí mají ve svém *SK* identifikátory o pokrytí sítí, které do nich zasahují (a příslušné polygony). Analogicky platí takovýto vztah pro záznamy o pokrytí. Pro ukázku využijeme tabulku 14. Dotazování v takovéto struktuře probíhá následujícím způsobem:

Chceme-li zjistit, které záznamy o pokrytí zasahují do obce s identifikátorem *la2575828*, vytvoříme dotaz prostřednictvím *PK* s filtrem:

PK = la2575828

získaný záznam:

la2575828	CHRaaj11AA	JSON

Obdržíme veškeré zápisy s *PK* uvedeném v podmínce dotazu, které se budou lišit v *SK*, obsahující identifikátor pokrytí. Zároveň obdržíme i hledané polygony pokrytí.

Dotaz, do kterých obcí zasahuje zkoumaná buňka pokrytí probíhá analogicky. Pokud bychom chtěli zkoumat záznam s *PK = BNSaad11CA,* vytvoříme dotaz s filtrem:

PK = BNSaad11CA

získaný záznam:

BNSaad11CA	la2540323	JSON

Obdržíme tedy záznamy s uvedeným *PK* a v *SK* budeme mít identifikátory hledaných obcí, včetně hledaných polygonů.

Návrh této struktury není příliš složitý, zároveň i dotazování je vcelku intuitivní a rychlé. Nevýhodou je ale fakt, že pro obce, do kterých zasahuje několik záznamů mobilní sítě se polygony obcí duplikují. Stejný jev platí i pro pokrytí – máme-li například 15 různých obcí, do nichž zasahuje stejná buňka pokrytí, bude polygon pokrytí v databázi 15x. To znamená, že tato struktura bude náročná na uložiště.

8.2 Duplikát pokrytí

Nevýhoda předchozí struktury je, že se duplikují polygony všech záznamů. Intuitivně je proto prvním dalším řešením tento problém minimalizovat či úplně odstranit. Cílem je vyhnout se duplikaci polygonů, především obcí, které jsou obsáhlé a ušetřit tak místo. Prvním krokem je utvořit jedinečné zápisy, definující pouze obce, včetně jejich polygonů. Tím je zajištěno, že nedochází k duplikaci těchto dat, jelikož tyto polygony se již v tabulce dále nevyskytují. Celkový počet obcí činí přibližně 6300, zatímco počet buněk pokrytí je značně větší, avšak jejich velikosti polygonů jsou výrazně menší, tudíž duplikáty polygonů pokrytí nemusí způsobovat takové potíže.

V tomto návrhu existují opět dva druhy záznamů. *PK* je u všech záznamů ve formátu vysvětleném v sekci 5.1.1, tedy *PK* obsahuje vždy identifikátor obce.

Jelikož v DynamoDB není možno vytvořit záznam, který má stejný *PK* a SK, je nutno je odlišit. Zároveň není možno vytvořit záznam, který by v *SK* neměl žádný atribut. V případě záznamu o obci, který obsahuje polygon, je toto vyřešeno následovně. Do

sloupce *SK*, který obsahuje identifikátor obce, je přidána předložka *loc*, definující že se jedná o *lokalitu* – neboli obec. Příkladem *SK* takového zápisu je:

loc.la2587842

PK a *SK* jsou odlišené pouze předložkou. Stejná konvence platí pro všechny záznamy obcí v tomto návrhu. Tato konvence není v tabulkách nijak zvýrazněna, jelikož tento parametr slouží pouze k dotazování. V tabulce 15 je uveden příklad záznamu obce. Na rozdíl od návrhu v sekci 8.1 se příslušné polygony vyskytují v databázi jen jednou.

Tabulka 15 - Záznam obce.

PK	SK	pol
la2587842	loc.la2587842	JSON

Druhý typ záznamu značí pokrytí mobilní sítí. Rozdíl mezi touto strukturou a tou v sekci 8.1 je, že v této struktuře je v *PK* vždy uveden identifikátor obce, do které zasahuje a v *SK* je obsažen identifikátor pokrytí. Tento záznam obsahuje i polygon pokrytí, přičemž pro záznamy, které zasahují do více obcí jsou tyto polygony duplikovány. Příklad takového záznamu je v tabulce 16.

Tabulka 16 - Záznam pokrytí.

PK	SK	pol
la2587842	ZDRaag31AB	JSON

Tento záznam značí, že buňka pokrytí s identifikátorem *ZDRaag31AB* zasahuje do obce *la258742*. Celková struktura s několika záznamy je vyobrazena v tabulce 17.

Tato struktura se na první pohled výrazně neliší od té v sekci 8.1, kdy každá má své výhody i nevýhody. Hlavní výhodou této struktury je méně duplikátů polygonů, což ale nemusí automaticky znamenat vhodnější návrh.

Tabulka 17 - Struktura s duplikáty pokrytí.

PK	SK	pol
la2587842	loc.la2587842	JSON
la254848	loc.la2574848	JSON
la2587842	ZDRaag31AB	JSON
la2587842	ZDRaam11BA	JSON
la254848	CHRabz52AA	JSON
la254848	CHRacs52AB	JSON

Využijeme tabulku 17 k názorné ukázce – dotazování v této struktuře probíhá podobně jako v sekci 8.1, tedy chceme-li obdržet například záznamy o pokrytí v obci s identifikátorem *la254848,* zadáme:

PK = la254848

získané záznamy:

la254848	loc.la2574848	JSON
la254848	CHRabz52AA	JSON
la254848	CHRacs52AB	JSON

Výsledkem jsou záznamy pokrytí a jejich polygony, které mají v *PK* identifikátor této obce a v *SK* hledané identifikátory pokrytí. Získáme však i záznam o obci samotné, přičemž v reálné situaci tabulka obsahuje více parametrů a lze využít některého z nich k odfiltrování nechtěných záznamů. Pro názornou ukázku označíme filtrování pomocí jiného parametru výrazem:

dodatečnýFiltr

Pokud bychom tak učinili, výsledkem by bylo:

PK=la254848 AND 'dodatečnýFiltr'

získané záznamy:

la254848	CHRabz52AA	JSON
la254848	CHRacs52AB	JSON

Dotaz opačný, do kterých obcí zasahuje záznam pokrytí, sice postupem působí jako jednoduchý, má ale zádrhel. V praxi by probíhal následovně. Pomocí určení *SK* jsou obdrženy veškeré záznamy s identifikátorem pokrytí.

SK = ZDRaag31B

získaný záznam:

la2587842	ZDRaag31AB	JSON

Hledanou obec bychom poté zjistili jednoduše využitím získaného *PK* a specifikací *SK* (filtrace záznamů o pokrytí), abychom získali pouze záznam o obci. Je nutno tedy vytvořit navazující dotaz, který by vypadal:

PK = la2587842 → PK = la2587842 AND SK = loc.la2587842

získaný záznam:

la2587842	loc.la2587842	JSON

Problém je ale následující – veškeré dotazy jsou vykonávány funkcí *scan*. Ta v jednoduchosti „vezme“ zadaná kritéria pro záznamy a vyhledá je v databází. *Scan* je efektivní pouze ve chvíli, kdy vykonává skenování využitím *PK*. V tomto případě *scan* probíhá částečně využitím *SK*, což je pro databází obsahující obrovské množství záznamů neefektivní, až takřka nerealizovatelné. Zároveň je nutno pro jeden z dotazů použít dvoudílné dotazování. Oproti struktuře v sekci 8.1 však tento návrh zabírá výrazně méně paměti.

8.3 Položky vazby

Obě předchozí řešení obsahují určitou duplikaci údajů. Cílem je tento jev omezit a k tomu lze využít alternativní přístup – přidáním vazebných záznamů mezi pokrytím a obcemi. Definuje se nový druh záznamu, který bude mít pouze *PK* a *SK*, jelikož slouží pouze k poukázání na jednotlivé závislosti. Lze je metaforicky označit za „značky na křižovatce“. Struktura se velmi podobá těm v sekcích 8.1 a 8.2, avšak neobsahuje žádné duplikáty polygonů.

První ze tří druhů záznamů je záznam o obci, který se strukturou nijak neliší od předchozího návrhu. Záznam o obci je:

Tabulka 18 - Záznam obce.

PK	SK	pol
la2597520	loc.la259750	JSON

Rozdílem je, že *PK* nebude v tomto návrhu pro všechny tři druhy záznamů ve formátu identifikátoru obce.

Druhým typem záznamu je pokrytí sítí, který se mírně liší od předchozích. Potýká se se stejným problémem, jako zápis obce v návrhu v sekci 8.2, kterým je omezení v DynamoDB. Nelze tedy utvořit záznam, který má stejný *PK* i *SK*.

Toto je v případě záznamu o pokrytí zde vyřešeno přidáním předpony *net*. Tato předpona nemá žádný jiný účel, krom odlišení od *PK*. Příkladem zápisu *SK* v záznamu o pokrytí je:

net.BRUabu32AA

Co se týče barevného odlišení, v tomto případě není *SK* opět nijak odlišen, protože neobsahuje žádnou potřebnou informaci. Tento záznam obsahuje polygon dané buňky pokrytí a jeho struktura je uvedena v tabulce 19.

Tabulka 19 - Záznam pokrytí.

PK	SK	pol
BRUabu32AA	net.BRUabu32AA	JSON

Hlavním rozlišovacím znamením této struktury je třetí typ záznamu – vazebný prvek. Jak je zmíněno v úvodu návrhu, jedná se o „směrovací značky“ v databázi. Ten se vyznačuje tím, že neobsahuje žádný polygon, ani jakýkoliv další parametr krom *PK* a *SK*. Příklad vazebného záznamu je uveden v tabulce 20.

Tabulka 20 – Záznam vazby.

PK	SK
BRUabu32AA	la2597520

Obsahuje v *PK* identifikátor buňky pokrytí a v *SK* identifikátor obce, do které zasahuje. Záznam v tabulce 20 tedy vyjadřuje, že buňka pokrytí s identifikátorem *BRUabu32AA* zasahuje do obce *la2597520*.

Vazebné záznamy obsahují jen *PK* a *SK* a nejsou náročné na paměť. Jelikož každý dotaz bude vykonán prostřednictvím tohoto typu záznamu, může se tento fakt projevit jako důležitý. Názorná ukázka struktury s několika záznamy je uvedena v tabulce 21.

Tabulka 21 - Struktura se zápisy vazeb.

PK	SK	pol
la2597520	loc.la2597520	JSON
la2571164	loc. la2571164	JSON
BRUabu32AA	net.BRUabu32AA	JSON
CHRaaq11CA	net.CHRaaq11CA	JSON
DECabt11CA	net.DECabt11CA	JSON
BRUabu32AA	la2597520	—
CHRaaq11CA	la2571164	—
DECabt11CA	la2568155	—

Využijeme tabulky 21 k názorné ukázce dotazování využitím vazebných zápisů. Chceme-li získat zápisy pokrytí, které zasahují do obce s identifikátorem *la2571164*, vytvoříme dotaz:

SK = la2571164

získaný záznam:

CHRaaq11CA	la2571164

Obdržený vazebný prvek obsahuje v *PK* identifikátor pokrytí, který hledáme. Navazující dotaz by tedy byl:

PK = CHRaaq11CA

získáme záznamy:

CHRaaq11CA	net.CHRaaq11CA	JSON
CHRaaq11CA	la2571164	—

Pokud bychom chtěli odfiltrovat záznamy vazby, přidáme jednoduše specifikaci *SK*, čímž získáme pouze záznam s polygonem pokrytí.

PK = CHRaaq11CA AND SK = net.CHRaaq11CA

získaný záznam:

CHRaaq11CA	net.CHRaaq11CA	JSON

Dotaz na obce, do kterých zasahuje zvolený záznam o pokrytí by rovněž využíval vazebné záznamy. V tomto případě jsou dotazy vykonány prostřednictvím *PK* (popřípadě s dodatečnou specifikací *SK*), ale parametry pro navazující dotaz jsou získány z *SK*. Ukázka dotazu, do kterých obcí zasahuje záznam *BRUabu32AA* by byla:

PK= BRUabu32AA

získané záznamy:

BRUabu32AA	net.BRUabu32AA	JSON
BRUabu32AA	la2597520	—

Pro tento dotaz není záznam s polygonem pokrytí nutný a lze ho filtrovat jako v předchozích případech. Navazující dotaz by byl založen na obdrženém identifikátoru obce (*la2597520)* v *SK* vazebného záznamu:

SK = la2597520 → PK = la2597520

získaný záznam:

la2597520	loc.la2597520	JSON

Na rozdíl od struktur v sekcích 8.1 a 8.2 zde nedochází k duplikaci polygonů. To sice může na první pohled znamenat nejvhodnější návrh, avšak opět zde dochází k částečnému dotazování prostřednictvím *SK*. Efektivita dotazů se pravděpodobně bude jevit jako jednostranná. Na druhou stranu, z pohledu paměti bude tato struktura určitě šetrnější.

8.4 Jednoduchý primární klíč

Doposud každé představené řešení obsahovalo tzv. *Composite Primary Key* (viz. sekce 4.3), tj. struktura byla definována pomocí dvojice *PK* a *SK*. Tento přístup je efektivní z hlediska přehlednosti, neboť uživatel si může jednotlivé zápisy seřadit podle dvou parametrů. Z hlediska dotazování, to nemusí být nejefektivnější způsob, proto je třeba brát v potaz i verzi, která *SK* neobsahuje a je definována pouze pomocí *PK*.

Navržená struktura spočívá v tom, že veškeré identifikační údaje (o obcích i pokrytí) jsou vloženy do *PK*, čímž, čistě teoreticky, by dotazování mělo být vykonáno rychleji.

Tato struktura obsahuje tři různé druhy záznamů. Mezi nimi jsou „klasické“ záznamy o obci a pokrytí a složitější vazebný zápis. Novým parametrem v této struktuře je parametr *bind*, který je typu *boolean* a značí, zda se jedná o záznam vazebný či záznam s polygonem podle následujícího pravidla:

bind = *true* → **vazebný záznam**

bind = *false* → **záznam obce/pokrytí**

Záznamy o obcích jsou definovány jednoduše pomocí *PK* ve tvaru identifikátoru obce a příslušného polygonu. Parametr *bind* je roven hodnotě *false*, tudíž víme, že se nejedná o vazebný prvek. Tento parametr bude vhodný při dotazování. Příklad záznamu obce je v tabulce 22.

Tabulka 22 - Záznam obce.

PK	pol	bind
la2574848	JSON	false

Záznam o pokrytí sítí je definován na stejném principu. Liší se pouze tím, že v *PK* se nachází identifikátor buňky pokrytí a obsahuje příslušný polygon. Příklad záznamu o pokrytí sítí je uveden v tabulce 23.

Tabulka 23 - Záznam pokrytí.

PK	pol	bind
BRUabu32AA	JSON	false

Posledním typem záznamu je vazebný záznam, který se liší od toho, uvedeného v sekci 8.3. Rozdílem je, že tentokrát není k dispozici *SK* a je cílem této struktury vložit veškeré

potřebné informace do *PK*. Tento vazebný záznam je konstruován tak, aby v *PK* obsahoval identifikátory obce i pokrytí, jenž mají společný průnik. Toho je docíleno tak, že do *PK* jsou tyto dva identifikátory vloženy společně a jsou odděleny tečkou. Barevně je odlišen tento záznam, využitím již známých konvencí ze sekce 7.1. Tj část patřící pokrytí je zvýrazněna modře, část patřící obci zeleně.

BRUabu32AA.la2597520

Zároveň, jelikož se jedná o vazebný záznam, je v tomto případě parametr *bind* roven hodnotě *true*. Tento záznam opět neobsahuje polygon, protože jde o pouhou vazbu. Příklad záznamu je uveden v tabulce 24.

Tabulka 24 - Záznam vazby.

PK	bind
BRUabu32AA.la2597520	true

Z *PK* tohoto záznamu zjistíme, že buňka pokrytí s identifikátorem *BRUabu32AA* zasahuje do obce *la2597520*. Celková struktura s několika záznamy je uvedena v tabulce 25.

Tabulka 25 - Struktura Simple PK.

PK	pol	bind
la2574848	JSON	false
la2597520	JSON	false
BRUabu32AA	JSON	false
CHOaaq51AA	JSON	false
BRUabu32AA.la2597520	—	true

Hlavní rozdíl této struktury je ve formě dotazování. Doposud byl vždy využit operátor „=“, jelikož bylo možno zadat konkrétní *PK* jako parametr vyhledávání. V tomto případě je nutno využít operátoru „*CONTAINS*“. To znamená, že při vyhledávání vložíme úryvek *PK*, který je pro nás důležitý.

Pokud bychom chtěli záznamy pokrytí v obci s identifikátorem *la25975220*, dotaz by byl následující:

PK: CONTAINS{"la2597520"}

získané záznamy:

la2597520	JSON	false
BRUabu32AA.la2597520	—	true

Záznam o obci samotné bychom mohli odfiltrovat využitím nového parametru *bind*.

PK: CONTAINS{"la2597520"} AND bind = true

získaný záznam:

BRUabu32AA.la2597520	true

Navazující dotaz by využíval stejný operátor. Z vazebného záznamu lze vyčíst identifikátor pokrytí, který je použit v dalším dotazu. Vypadal by následovně:

PK: CONTAINS{"BRUabu32AA"} AND bind = false

získaný záznam:

BRUabu32AA	JSON	false

Dotaz opačný probíhá analogicky, jediný rozdíl je zadaný *PK*. Ve zkratce by vypadal následovně:

PK: CONTAINS{"BRUabu32AA"} AND bind = true

získaný záznam:

BRUabu32AA.la2597520	true

Z vazebného prvku lze vyčíst identifikátor obce, do které zkoumaný záznam pokrytí zasahuje.

PK: CONTAINS{"la2597520"} AND bind = false

získaný záznam:

la2597520	JSON	false

Oba dotazy tedy probíhají stejným způsobem. Chceme-li buňky v obci, či obce, které buňka pokrývá, pomocí *PK* zjistíme identifikátor hledané entity a druhým dotazem získáme jejich polygony. Tato struktura ale není nejlepším řešením problematiky, a to z prostého důvodu, že operátor „*CONTAINS“* není vhodný pro velké množství dat. Vyhledávání záznamů probíhá výrazně pomaleji než za využití „=“, takže tuto strukturu nemá smysl testovat. Zároveň pro velké množství dat mohou záznamy vazeb působit nepřehledně.

8.5 Dva druhy vazebných prvků

Posledním zkoumaným řešením je modifikovaná verze již zmíněného řešení s vazebnými prvky. Jak je uvedeno v sekci 8.3 , při dotazování je vždy využíváno vazebných prvku. Rozdíl mezi touto strukturou a tou v sekci 8.3 je, že tato struktura obsahuje dva druhy vazebných prvků, které mají sloužit ke zlepšení přehlednosti a urychlení dotazů. Jedná se opět o návrh, který má *Composite key*.

Záznamy obcí jsou zavedeny na stejném principu, který je vysvětlen v sekci 8.2. Jediným rozdílem je, že mají v tomto případě prohozené *PK* a *SK*. V *PK* mají předponu *loc* z důvodu rozlišení od *SK*, způsobené omezením DynamoDB (viz. sekce 8.2).

Tabulka 26 - Záznam obce.

PK	SK	pol
loc.la2540323	la2540323	JSON

Záznamy pokrytí sítí jsou velmi podobné těm v sekci 8.3. Jediným rozdílem je, že jsou také prohozené *PK* a *SK*. Obsahují opět předponu *net*, jejíž smysl a definice je uvedena v sekci 8.3, avšak tentokrát v *PK*. To je z toho důvodu, že je potřeba mít v *PK* identifikátory s předponou kvůli dotazování. V tabulce 27 je příklad takového záznamu.

Tabulka 27 - Záznam pokrytí.

PK	SK	pol
net.PELabc52CA	PELabc52CA	JSON

Na rozdíl od předchozí vazebné struktury, obsahuje tento návrh dva typy vazebných záznamů, které poskytují lepší přehlednost a potenciálně i efektivitu při dotazování.

První typ vazby specifikuje, do jakých obcí zasahuje záznam pokrytí. Funguje na principu, že v *PK* je vložen identifikátor buňky pokrytí, zatímco do *SK* je vložen identifikátor obce, do které zasahuje. Jelikož se jedná o vazebný prvek, neobsahuje sloupec s polygonem. Příklad záznamu je uveden v tabulce 28.

Tabulka 28 - Záznam prvního typu vazby.

PK	SK
ZLIacl51AA	la2585068

Tento záznam „říká“, že buňka *ZLIacl51AA* zasahuje do obce *la2585068.*

Druhý typ vazeb je definován analogicky. Tvoří vazbu, která specifikuje, které záznamy pokrytí zasahují do dané obce. Má tedy v *PK* identifikátor obce a v *SK* identifikátor buňky pokrytí, která do ní zasahuje. Jedná se prakticky o prohozený první typ vazebného záznamu tohoto návrhu. Příklad druhého typu vazebného záznamu je uveden v tabulce 29.

Tabulka 29 - Záznam druhého typu vazby.

PK	SK
la2577421	JBNaao52AA

Tento záznam naopak nese informaci opačnou - do obce *la2577421* zasahuje buňka *JBNaao52AA.*

Důvod k zavádění dvou typů vazebných prvků je v praxi předveden při dotazování. Z teoretického hlediska jde o způsob, jak načítat záznamy z databází tak, abychom získali potřebné identifikátory, ale nenačítali obsáhlé polygony. To je v porovnání s návrhy v sekcích 8.1 a 8.2 úplně jiný přístup k problematice, který se může projevit jako vhodnější. V tabulce 30 je znázorněna struktura se všemi typy záznamů.

Tabulka 30 – Struktura s modifikovanými vazbami.

PK	SK	pol
net.PELabc52CA	PELabc52CA	JSON
net.JBNaao52AA	JBNaao52AA	JSON
loc.la2529869	la2529869	JSON
loc.la2540323	la2540323	JSON
ZLIacl51AA	la2585068	—
PELabc52CA	la2529869	—
la2577421	JBNaao52AA	—
la2540323	PBRacc51BA	—

Dotazování v této struktuře probíhá velmi podobně jako v sekci 8.3. Rozdíl je, že pro každý dotaz se využívá jiný typ vazebného záznamu. Pro zjištění, které záznamy pokrytí zasahují do obce bude dotaz vypadat následovně:

PK = la2577421

získaný záznam:

la2577421	JBNaao52AA

Následně se využije získaného identifikátoru v *SK* v druhém dotazu k obdržení záznamu s polygonem. Abychom nezískali opět stejné záznamy vazby, přidáme k identifikátoru předponu *net.*

SK = JBNaao52AA → PK = net.JBNaao52AA

získaný záznam:

net.JBNaao52AA	JBNaao52AA	JSON

Dotaz, do kterých obcí zasahuje zkoumaný záznam pokrytí probíhá takřka identicky, akorát je k tomu využito druhého typu vazebného záznamu. Dotaz by vypadal následovně:

PK = PELabc52CA

získaný záznam:

PELabc52CA	la2529869

Stejně jako v předchozím případě je využit obsah *SK* k vytvoření navazujícího dotazu a získání polygonu obce. Tentokrát je přidána předpona *loc*, abychom se vyhnuli záznamům vazeb.

SK = la2529869 → *PK* = *loc.la2529869*

získaný záznam:

loc. la2529869	la2529869	JSON

Tato struktura se na první pohled jeví jako výhodnější, než ta uvedena v sekci 8.3, jelikož tyto „oboustranné“ vazby umožňují rychlejší a snadnější získání informací při jakémkoli dotazu. V principu, rozdíl mezi těmito strukturami je, že při dotazování v sekci 8.3 je nutno částečně se dotazovat využitím *SK*, zatímco v této struktuře je použit vždy *PK*. Tento způsob by akci měl vykonávat mnohem rychleji z principu, který je zmíněn v sekcích 8.2 a 8.3.

8.6 Shrnutí

V sekcích 8.1 - 8.5 byla uvedena veškerá navržená řešení struktury databáze DynamoDB, včetně principu dotazování na jednotlivé záznamy (obec, pokrytí). V tabulce 31 je uvedeno stručné shrnutí struktur z několika aspektů, po nahrání reálné sady dat, zmíněné v kapitole 5. Způsob, jakým import dat probíhal je uveden v sekci 8.7.

Sloupec *Vazebné záznamy* určuje, zda (popř. kolik) obsahuje struktura těchto typů záznamů. *Polygony* značí výskyt polygonů formátu *JSON* v záznamech. *Počet záznamů* určuje celkový počet vytvořených záznamů po nahrání reálných dat. *Velikost* značí, kolik paměti daná struktura zabírá a *Výhody/Nevýhody* stručně zvýrazňují hlavní pozitiva a negativa dané struktury.

U návrhu v sekci 8.4 není uveden počet záznamů a velikost, protože se jedná o pouhý prototyp, který se, ve srovnání s ostatními návrhy, jevil jako nevhodný, a tudíž do něj nebyla data ani nahrána.

Tabulka 31 - Shrnutí navržených řešení.

Č.	Vazebné záznamy	Polygony	Počet záznamů	Velikost	Výhody/Nevýhody
8.1	NE	U všech záznamů	454 247	9.17 GB	+Efektivní dotazování -Velikost
8.2	NE	U všech záznamů	273 226	2.56 GB	+Méně duplikátů -Jeden dotaz pomocí *SK*
8.3	1 typ	U obcí U pokrytí	370 867	949.6 MB	+Bez duplikátů, velikost -Jeden dotaz pomocí *SK*
8.4	1 typ	U obcí U pokrytí	—	—	+Bez duplikátů, vše v *PK* -Dotazy pomocí *CONTAINS*
8.5	2 typy	U obcí U pokrytí	676 013	961.24 MB	+Bez duplikátů, velikost -Počet záznamů, složitost

Z tabulky 31 je zřejmé, že struktura ze sekce 8.1 je vhodná k testování, jelikož se projeví, zda jsou dotazy vykonány rychle a zároveň lze zkoumat, zda hraje roli velikost databáze. Naopak struktura ze sekce 8.5 má výrazně menší velikost a na rozdíl od té v sekci 8.3, veškeré dotazy probíhají pomocí *PK*, tudíž se jedná o druhou vhodnou strukturu k testování. Testování a výsledky jsou uvedeny v kapitole 9.

8.7 Import dat do databáze

Uvedené struktury pouze definují vzhled a systém fungování uvnitř databáze. Vložení velkého množství dat musí být vykonáno externě. K tomu je využito nástrojů v rámci *Amazon Web Services* a to konkrétně kódovacího prostředí AWS Lambda, které umožňuje provozovat vlastní kódy bez nutnosti jakékoliv jiné aplikace a tzv. *batch insert*, což je zjednodušeně funkce, pomocí které lze větší množství dat importovat po „dávkách“.

9 Analýza

Ke zjištění efektivity navržených struktur je nutno provést empirické testování. Nejprve je třeba provést prostý test na prototypu struktury (ručně vytvořených několik záznamů, viz. sekce 9.1) a poté navázat testováním doby trvání vykonání dotazů. K testování byly zvoleny struktury uvedené v sekcích 8.1 a 8.5, jelikož se jedná o dvě odlišná pojetí problému. Jeden z návrhů zabírá více uložiště, zatímco druhý má výrazně více záznamů.

9.1 Test struktury

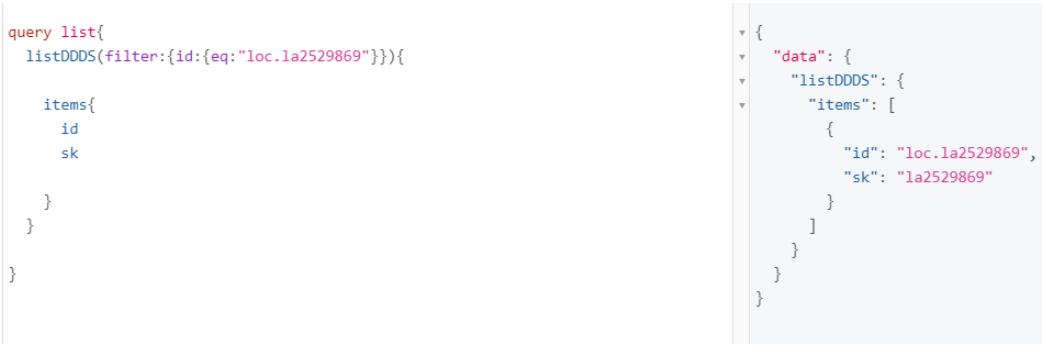
Dříve než je daná struktura empiricky testována je třeba ověřit funkcionalitu uvnitř struktury samotné. Tím je myšleno, že požadované dotazy jsou otestovány nejdříve v teorii, než se aplikují na praktické situace a reálná data. K tomu slouží aplikace v prostředí AWS zvaná AppSync. Tato aplikace umožňuje importovat existující DynamoDB do role zdroje dat. Tudíž poté dotazy, které jsou v tomto prostředí vykonávány, čerpají data z vybrané tabulky. Zároveň lze skrze toto prostředí tvořit nové záznamy či aktualizovat existující. Výhodou AppSync je možnost navrhnout si vlastní „schéma“, které určuje, jakým způsobem fungují již zmíněné dotazy na data, nové zápisy či aktualizace. Jednoduchý příklad testování dotazů je uveden na obrázku 10, kde je testována struktura uvedená v sekci 8.5. Levá strana představuje příkazovou řádku, ve které jsou zadány parametry dotazu, zatímco pravá strana představuje výsledky dotazu. Je předveden dotaz na obce, které jsou pokryty buňkou *PELabc52CA*. V tabulce 30 parametr *id* má stejný význam jako *PK* v navržené struktuře.

Obrázek 10 – Příklad dotazu v AppSync – 1.část.

Z výsledku dotazu vidíme, že vazebný záznam má v *SK* identifikátor hledané obce. Při pohledu na tabulku 30, jsme využili vazebný záznam za účelem získání záznamu:

loc.la2529869	la2529869	JSON

Ten je poté získán navazujícím dotazem, který je vyobrazen na obrázku 11.



Obrázek 11 - Příklad dotazu v AppSync – 2. část.

9.2 Empirické testování

Všechny navržené struktury se jeví jako funkční, ale každá má své určité nedostatky. Testovat ve všech ohledech každou strukturu je nepotřebné, proto byly k testování zvoleny struktury uvedeny v sekcích 8.5 a 8.1, protože se jedná o dvě základní myšlenky – návrh se záznamy vazby a bez nich. Testování probíhá způsobem, že z nahraných reálných dat je náhodně dotazováno určité množství záznamů.

Ideální by bylo testovat všechny dotazy ve stejném poměru záznamů (tj např 600 obcí a 7000 záznamů pokrytí), ale jelikož se každý dotaz účtuje (v prostředí AWS), postačí pro tuto agendu dotazů méně.

Pro strukturu s vazbami bylo vymezeno 5000 (1.typ, pokrytí) a 1000 (2.typ, obce) dotazů, zatímco pro duplikáty 1000 dotazů (oba typy), přičemž nebyly vždy pokládány dotazy na stejné záznamy. To znamená v praxi, že je provedeno 5000 nebo 1000 náhodných dotazů (na obce a pokrytí) a výstupem je tabulka, ve které je uveden počet nalezených záznamů, počet nalezených polygonů a čas, za který byl tento dotaz vykonán.

Náhodně se tedy vybere jedna entita (obec, buňka pokrytí) a zadá se na ní příslušný dotaz. V tabulkách jsou uvedeny průměrné časy pro výsledný počet nalezených záznamů.

Pro příklad – je zkoumána obec A. Zadá se dotaz na pokrytí, který „vrátí“ 5 záznamů s časy. Pro obce B, C, D se vykoná stejný dotaz a rovněž „vrátí“ 5 záznamů. Z obdržených časů se získá průměrný čas pro získaných 5 záznamů. Dotazování na pokrytí sítí probíhá stejným způsobem.

Je nutno podotknout, že při zkoumání obcí i pokrytí, se velikosti testovaných polygonů liší, tj nemají jednotný rozměr. To může způsobit určité nepravidelnosti ve výsledcích, například nízké časy pro velký počet záznamů (nebo naopak).

9.3 Testování struktury s vazebnými záznamy

9.3.1 Pokrytí obcí konkrétní buňkou

Nejprve je analyzován dotaz typu „Do kterých obcí zasahuje daná buňka?“. Při pohledu na sekci 8.5, jsou nejdříve obdrženy identifikátory obcí v *SK* prostřednictvím prvního dotazu (podle počtu získaných záznamů, víme do kolika obcí buňka zasahuje), a poté prostřednictvím druhého, jsou získány polygony zkoumaných obcí.

Příkladem průběhu testování je:

1. Položený dotaz na náhodně zkoumaný záznam pokrytí mobilní sítí
2. Získání počtu obcí, do kterých zkoumaná buňka pokrytí zasahuje (=počet vazebných záznamů)
3. Získání polygonů obcí (druhý dotaz)
4. Zaznamenání doby trvání kroků 1. - 3.
5. Opakování kroků 1. – 4. na jiných buňkách pokrytí

Pro různé buňky pokrytí obdržíme různý počet záznamu obcí. Pro moji práci není důležité, o jakou konkrétní buňku pokrytí se jedná, vyhodnocuje se pouze průměrný čas trvání dotazu v závislosti na počtu získaných záznamů obcí (tj nehraje roli velikost buňky, nebo kde se nachází). Je nutno podotknout, že zkoumaných 5000 dotazů je náhodných a neznamená to, že jsou zkoumány veškeré obce/pokrytí. Jinými slovy, hodnoty uvedené v tabulce 33 jsou získány pouze z této množiny 1000 dotazů (stejný fakt platí i v sekci 9.4.1, přičemž je generována jiná, náhodná množina dotazů).

Příklad:

Pro hodnotu *Počet záznamů = 2 , Čas [ms] = 322.83* z tabulky 33 platí, že nebyly zkoumány všechny záznamy pokrytí v ČR, které zasahují do přesně 2 obcí, ale pouze ty, které byly součástí 5000 zkoumaných dotazů.

Z tabulky 33 tedy vyplývají následující tvrzení, pro vymezený počet dotazů:

Dotazy na buňky pokrytí, které zasahují přesně do 2 obcí trval průměrně 322.83 ms

Dotazy na buňky pokrytí, které zasahují přesně do 4 obcí trval průměrně 414.65 ms

…

Tabulka 32 obsahuje globální průměrné hodnoty (v rámci datasetu) pro tuto strukturu a tento typ dotazu. Jinými slovy – v průměru zkoumaná buňka pokrytí zasahovala do 4.918 obcí a dotaz trval 457.683 ms.

Tabulka 32 - Průměrné hodnoty.

Průměrný počet obdržených záznamů	**průměrný čas[ms]**
4.918	457.683

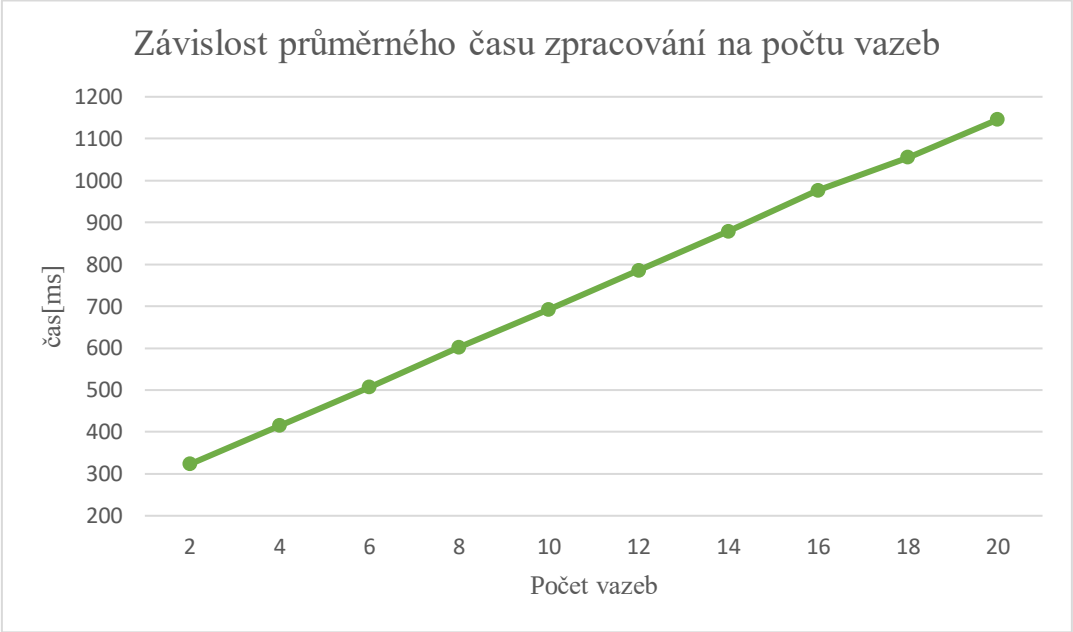
Tabulka 33 – Vazby (1.dotaz).

Počet záznamů	**Čas [ms]**
2	322.83
4	414.65
6	506.47
8	601.98
10	692.04
12	785.49
14	878.6
16	976.38
18	1055.47
20	1145.48

Obrázek 12 je pouhá vizualizace tabulky 33, tj je utvořen z dat této tabulky. Slouží ke zkoumání charakteru křivky závislosti času vykonání dotazu na počtu nalezených záznamů, jelikož ne vždy to může být z tabulkových hodnot zřejmé.

Z obrázku je zřejmé, že závislost je lineární. Pro zkoumaný dataset 5000 dotazů platí pro tento typ dotazu, že s rostoucím počtem pokrytých obcí, roste i čas vykonání dotazu.

Jelikož časy pro nalezených 20 záznamů zasahují do rozmezí 1-1.5s dá se tento přístup označit za efektivní. Mezi zkoumanými dotazy se nacházely i dotazy pro nalezených 30-60 záznamů, přičemž jejich čas vykonání byl v rozmezí 1.5 – 3s. S ohledem na fakt, že záznamů pokrývajících obec je ve většině případů méně než 100, jsou tyto časy odezvy adekvátní.



Obrázek 12 - Graf pro záznamy (1.dotaz).

9.3.2 Pokrytí konkrétní obce

Opačný dotaz, „Které buňky pokrytí zasahují do dané obce?“, probíhal takřka analogicky. Jak je uvedeno v sekci 8.5, nejprve jsou obdrženy identifikátory buněk pokrytí mobilní sítí v *SK* prostřednictvím prvního dotazu a poté pomocí druhého jsou získány polygony zkoumaných buněk. Průběh testování je v tomto případě:

1. Položený dotaz na náhodně zkoumanou obec
2. Získání počtu buněk (vazebných záznamů) pokrytí mobilní sítí, které do zkoumané obce zasahují
3. Získání polygonů buněk pokrytí, obdržených v kroku 2. (2. dotaz)
4. Zaznamenání doby trvání kroků 1.-3.
5. Opakování kroků 1.-4. na jiné obci

Pro většinu obcí se obdrží jiný počet záznamů o pokrytí. Pro mojí práci nehraje roli velikost obce, či její lokalita, zaznamenává se pouze průměrná doba trvání dotazů v závislosti na počtu nalezených záznamů pokrytí. I pro tento dotaz platí, že hodnoty

uvedené v tabulkách byly získány z 1000 náhodných dotazů (stejný fakt platí i v sekci 9.4.2, přičemž je generována jiná, náhodná množina dotazů). Význam tabulkových hodnot je vysvětlen na příkladu.

Příklad:

Hodnota *Počet záznamů = 6, Čas [ms] = 508.*5 z tabulky 35, platí pouze pro záznamy v rámci zkoumaných dotazů, tj spočítaný průměrný čas není utvořen ze všech obcí, do kterých zasahuje přesně 6 záznamů pokrytí.

Pro tento typ dotazu je závislost uvedena v tabulce 35. Zjistíme z ní:

Dotazy na obce, do kterých zasahuje přesně 6 záznamů o pokrytí trval průměrně 508.5 ms

Dotazy na obce, do kterých zasahuje přesně 8 záznamů o pokrytí trval průměrně 3804.5 ms

…

Je zřejmé, že sloupec *Počet záznamů* v tabulce 35 neobsahuje stejné hodnoty jako v ostatních případech. To je buď z toho důvodu, že tento test produkoval nepravidelné výsledky a pro některé z dříve použitých hodnot nebyly ani získány časy (chybné hlášení) nebo proto, že náhodná sada dotazů neobsahovala konkrétní hodnotu počtu záznamů (např. nebyl vygenerován výstup pro 2 nalezené záznamy). Uvedené hodnoty jsou ty, které se nejvíce blíží již použitým hodnotám.

Tabulka 34 obsahuje průměrný počet získaných záznamu a jejich dobu trvání, přičemž tyto hodnoty byly spočítány využitím pouze těch dotazů, které produkovaly časy. Velké množství dotazu skončilo chybným hlášením (dlouhá doba trvání dotazu). To znamená, že z dotazů na obce, které měly časy, byly obce průměrně pokryty 45.32 buňkami mobilní sítě a dotaz trval průměrně 8031.36 ms.

Je nutno podotknout, že tabulkové hodnoty od počtu 17 záznamů obsahovaly několik záznamů s hodnotou „*TimeOut*“ neboli administrátorem přerušený příkaz. Několik dotazů zabíralo příliš mnoho času a v této tabulce nejsou uvedeny (závislost je tvořena pouze z číselných hodnot).

Tabulka 34 - Průměrné hodnoty.

Průměrný počet nalezených záznamů	průměrný čas[ms]
45.32	8031.36

Tabulka 35 - Vazby (2.dotaz).

Počet záznamů	Čas [ms]
6	508.5
8	3804.5
11	1906
12	4123.14
13	4911.85
14	2251.5
15	6695.67
16	6204.36
17	6588.3
18	5371.67
19	10198.5
20	5127.46

Pro tento dotaz časy zpracování nevykazují takto efektivní a lineární závislost jako v sekci 9.3.1. Z testovaných 1000 dotazů (neboli obcí) přibližně 50-60 % vykazovalo buď chybné hlášení nebo bylo pozastaveno administrátorem, což je způsobeno příliš dlouhou dobou trvání dotazu. Do rozmezí 20 nalezených záznamů (a tedy i polygonů) byly vyprodukovány do jakési míry adekvátní časy. Veškeré dotazy, které vrátily více než 20 záznamů, neprodukovaly použitelné výsledky.

Na obrázku 13 je opět pouhá vizualizace tabulky 35, ze kterého lze pozorovat charakter křivky. Při pohledu na obrázek 11 z prvního dotazu (sekce 9.3.1) jsou časy obecně výrazně vyšší. Krom několika poklesů v časech (11,14,18,20 záznamů) křivka má rostoucí charakter. Tyto poklesy mohou být buď náhodné, nebo mohou být způsobeny analýzou z malého množství dat (pro 20 nalezených záznamů většina dotazů neměla použitelnou dobu trvání) nebo variabilní velikostí polygonů. Obecně důvod kolísavého charakteru je podrobněji rozebrán v sekci 9.5.

Obrázek 13 - Graf pro záznamy (2.dotaz).

Výsledek tohoto dotazu nabízí jedno možné vysvětlení, proč valná většina dotazů nebyla finalizována. Dotaz v sekci 9.3.1 proběhl bez obtíží a produkoval dobré výsledky. Rozdíl je však, že jedna buňka pokrytí většinou zasahuje do menšího počtu obcí, než je tomu naopak. Jinými slovy v reálném světě:

Zkoumaná buňka pokrytí zasahuje většinou do rozmezí 1-50 obcí, tj je nutno projít maximálně 50 záznamů.

Do zkoumané obce může zasahovat i 150+ buněk pokrytí, tj je nutno projít velké množství záznamů.

Při zkoumání jedné obce, do které zasahuje 150 buněk pokrytí je tedy nutno získat 150 záznamů vazeb a pomocí nich obdržet příslušné polygony. Jelikož stejný typ dotazu ve struktuře s duplikáty výrazné potíže nevykazoval, může být problém v principu vazebného záznamu.

9.4 Testování struktury s duplikáty

9.4.1 Pokrytí obcí konkrétní buňkou

Z důvodu jednoduchosti této struktury se testování mírně lišilo od toho v sekci 9.3.1. Průběh analýzy samotné je identický jako v sekci 9.3.1. Klíčový rozdíl mezi

zkoumanými strukturami jsou vazebné záznamy, které tato struktura nemá. Vazebné záznamy jsou z pohledu uložiště úspornější, avšak z hlediska rychlosti vykonání dotazů mohou být nevhodné. Dotazování probíhá následujícím způsobem:

1. Položení dotazu na konkrétní buňku pokrytí
2. Získání všech záznamů obcí, které tato buňka pokrývá, včetně polygonů
3. Zaznamenání doby trvání
4. Opakování kroků 1.-3. na jiných buňkách

Výsledky analýzy prvního dotazu pro strukturu s duplikáty polygonů jsou uvedeny v tabulce 37, přičemž platí tvrzení o zdroji hodnot ze sekce 9.3.1. Z tabulky 37 lze vyčíst, že:

Dotazy na buňky pokrytí, které zasahovaly přesně do 5 obcí trval průměrně 328.71 ms

Dotazy na buňky pokrytí, které zasahovaly přesně do 10 obcí trval průměrně 239 ms.

...

V tabulce 36 jsou uvedeny globální průměrné hodnoty pro tento typ dotazu, tedy – 1000 zkoumaných buněk pokrytí průměrně zasahovalo do 3.012 obcí a vykonání dotazu trvalo 245.802 ms.

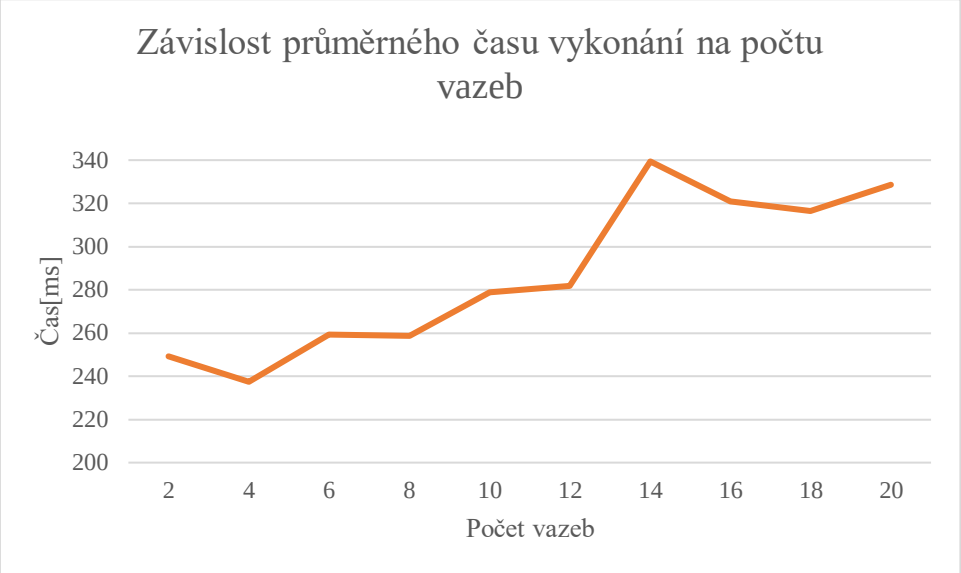
Tabulka 36 - Průměrné hodnoty.

Průměrný počet nalezených záznamů	**průměrný čas[ms]**
3.012	245.802

Tabulka 37 - Duplikáty (1.dotaz).

Počet záznamů	**Čas [ms]**
2	249.19
4	237.41
6	259.38
8	258.65
10	278.81
12	281.83
14	339.4
16	321
18	316.333
20	328.75

Obrázek 14 je opět pouhou vizualizací hodnot z tabulky 37. Vyjma drobného poklesu doby trvání pro 8 a 16 záznamů je tato závislost lineární. Maximální počet získaných záznamů v rámci datasetu byl cca 25 a při tomto počtu trval dotaz v rozmezí 380-420 ms.



Obrázek 14 - Graf pro duplikáty (1.dotaz).

V porovnání s obrázkem 12 v sekci 9.3.1 byly v této struktuře vykonány dotazy rychleji. Sice má zde křivka mírné kolísání (příčiny vysvětleny v sekci 9.5), avšak celkové časy jsou nižší. Zároveň dobrým výsledkem tohoto testování je fakt, že rozdíl mezi časy pro nalezených 0 a 16 záznamů je přibližně 0.1s, což je zanedbatelný růst doby trvání. Krom toho nebylo v množině 1000 zkoumaných dotazů vyprodukováno žádné chybné hlášení.

9.4.2 Pokrytí konkrétní obce

Testování druhého dotazu pro tuto strukturu probíhalo analogickým způsobem jako v sekci 9.4.1. Výsledek dotazu, zjišťující počet záznamů pokrytí v dané obci, byl v předchozím testování nepříznivý (sekce 9.3.2). Pro tuto strukturu lze tvrdit, že výsledky jsou pozitivní.

Průběh testování byl následující:

1. Položení dotazu na konkrétní obec
2. Získání všech záznamů buněk pokrytí, které do zkoumané obce zasahují, včetně příslušných polygonů
3. Zaznamenání doby trvání
4. Opakování kroků 1.-3. na jiných buňkách

V tabulce 39 jsou uvedeny výsledky pro analýzu druhého dotazu u struktury s duplikáty. Tento dataset vygeneroval vyšší hodnoty ve sloupci *Počet záznamů,* tj neobsahoval hodnoty 2,4 apod. To je z testovacího hlediska i vhodné, jelikož jsou poprvé testovány takto vyšší hodnoty záznamů. Důvodem, proč v sekci 9.3.2 nejsou uvedeny časy pro hodnoty 30–80 záznamů je, že takovéto množství záznamů vrátilo chybné hlášení či nebylo vůbec vygenerováno. Pro tento dotaz platí rovněž tvrzení o významu hodnot ze sekce 9.4.2. Z tabulky 39 lze tedy zjistit, že:

Dotazy na obce, do kterých zasahuje přesně 5 záznamů o pokrytí trval průměrně 328.71ms.

Dotazy na obce, do kterých zasahuje přesně 10 záznamů o pokrytí trval průměrně 239ms.

…

Tabulka 38 obsahuje průměrný počet získaných záznamu a jejich dobu trvání, přičemž tyto hodnoty nebyly spočítány využitím všech 1000 dotazů. I tento typ struktury produkoval několik chybných hlášení (v hodnotách kolem 140 nalezených záznamů), avšak i tak jich bylo výrazně méně než v sekci 9.3.2. Při pohledu na tabulku 38 je zřejmé, že z dotazů na 1000 obcí byly obce průměrně pokryty 34.544 buňkami mobilní sítě a dotaz trval průměrně 300.232 ms.

Na obrázku 15 je graf vytvořený z hodnot tabulky 39. Z něj lze vypozorovat, že vyjma vyššího skoku v hodnotě 40 záznamů je trend závislosti doby trvání na počtu obdržených záznamů také lineární. Při pohledu na časy v obrázku 15 či tabulce 39, jsou tyto časy výrazně lepší než ty uvedené pro stejný dotaz v předchozí struktuře (viz. sekce 9.3.2).

Otázkou je nepravidelný nárust/pokles časů ve vyšších počtech záznamů. Dá se říci, že poklesy časů ve větších hodnotách mohou být způsobeny tím, že byly dotazovány menší polygony. Naopak u menších hodnot, které mají vyšší časy (obrázek 15, 40 záznamů), je možné že nalezené polygony byly větší než v jiných případech. Tato problematika je podrobněji rozebrána v sekci 9.5.

Tabulka 38 - Průměrné hodnoty.

Průměrný počet nalezených záznamů	průměrný čas[ms]
34.544	300.232

Tabulka 39 - Duplikáty (2.dotaz).

Počet záznamů	Čas [ms]
5	328.71
10	239
15	286.27
20	241.65
25	258.22
30	261.5
40	386.25
50	311.5
60	322
70	339
80	379.5

Obrázek 15 - Graf pro duplikáty (2.dotaz).

Obecná nevýhoda tohoto řešení je velikost. V těchto dotazech jsou pokaždé zahrnuty i polygony pokrytí, které jsou například velmi obsáhlé, avšak do zkoumané obce zasahují jen malou částí, ale pro toto testování jsou načítané jako celek.

9.5 Shrnutí

Z navržených pěti struktur byly zvoleny dvě a ty byly empiricky otestovány v sekcích 9.3 a 9.4. Z testování vyplývá, že dotaz na buňku pokrytí, která zasahuje do určitých obcí probíhá u obou testovaných struktur bez větších obtíží, zatímco dotaz opačný je efektivní pouze u struktury v sekci 8.1.

Krom grafu uvedeného v sekci 9.3.1 (obrázek 12), vykazovaly grafy místy „nečekaný“ skok/pokles v hodnotách. Jak je zmíněno v závěru sekce 9.2 kolísavý charakter může být způsoben nepravidelností polygonů.

Jiné možné vysvětlení toho může být, že je v jedné instanci zkoumaná malá obec s 2 malými záznamy (polygony) pokrytí a poté je zkoumána jiná obec, rovněž s 2 polygony pokrytí, které jsou však velké. Tyto dvě hodnoty jsou poté zaznamenány a tvoří průměr, což může způsobit nepravidelnosti. V této práci se ale takové případy nerozlišují.

Další vliv na testování může mít zkoumaná oblast. Města a hustě obydlené lokality jsou více pokryté buňkovou sítí než místa jiná a množství záznamů nebo jejich velikost může mít vliv na výsledky testování.

Jak je zmíněno i v závěru sekce 9.4.2 může být problémem velikost dat. Možným řešením tohoto problému může být ukládání všech polygonů do jiného formátu, než je *JSON*, a to například do formátu WKB (viz. sekce 3.4.4), který je mnohem úspornější než *JSON*, co se paměti týče. NoSQL databáze mohou zpracovávat oba tyto formáty. Mohl by nastat problém v pozdějším převodu a vizualizaci, jelikož data v binární podobě nelze jednoduše zobrazit a nejsou tzv. *human readable* neboli čitelné člověkem, zatímco *JSON* tyto výhody zahrnuje.

9.6 Alternativy

Celé testování a veškeré návrhy byly prováděny v prostředí od Amazonu. DynamoDB není jediné možné NoSQL řešení na trhu. Je možné, že pro tento typ testování by se

jiné databáze jevily jako lepší možnost. Takové databáze jsou například Redis, CouchDB od Apache Software, Cassandra od stejné firmy či Elasticsearch. Redis se vyznačuje vysokou rychlostí zpracování dat (jsou ukládány do RAM), Cassandra, která byla vyvinuta v „laboratořích“ Facebook-u je hybridní databází (*key value/column oriented*), CouchDB je ideální při tvorbě webových aplikací a například Elasticsearch má výhodu, že je možno vykonávat komplexní dotazy. Všechny uvedené datábaze mají své výhody i nevýhody. Abychom došli k nějakému komplexnímu závěru, která z těchto databází je pro danou agendu nejefektivnější, bylo by nutno provést totožné testy na všech platformách [34].

10 Závěr

Tato práce se zaměřovala na možnosti realizace struktury databáze, která má být využita na ukládání záznamů o pokrytí České republiky mobilní sítí. Cílem práce bylo navrhnout vhodnou strukturu, která by vyhovovala stanoveným požadavkům při dotazování na jednotlivé záznamy v databázi.

V první části práce je teoretický úvod, popisující veškeré technologie a principy, které byly potřebné pro realizaci. K práci s daty (konkrétně k vizualizaci) byl využit program QGIS, jehož princip je rozebrán v kapitole 3. Poté je uveden stručný teoretický přehled na strukturu a funkci databází jako celku, přičemž jsou následně rozebrány navržené struktury databáze.

K realizaci byla využita databáze DynamoDB od Amazonu, která nabízí četné výhody, jako například možnost škálování, což je v problematice dat z mobilních sítí důležitá vlastnost. Pro moji práci byla ale klíčová vlastnost DynamoDB ta, že nabízí možnost, aby s daty manipulovalo velké množství uživatelů ve stejnou chvíli. Bylo navrženo několik možných struktur databáze a u každé z nich jsou názorně uvedeny způsoby, jakými probíhají konkrétní požadované dotazy na data.

K empirické analýze byly zvoleny dvě navržené struktury, jevící se jako nejadekvátnější a u nich byly otestovány rychlosti, jakými probíhají dané dotazy. Výsledky jsou zpracovány do podoby tabulek a grafů v sekcích 9.3 a 9.4.

Pokud bych měl zhodnotit výsledek, tak jeden typ požadovaného dotazu se jevil u obou zkoumaných struktur jako efektivní a dalo by se říct, že by mohl být v určité formě aplikován v reálném provozu. Dotaz druhý vykazoval relativně efektivní výsledky pouze u jedné struktury (sekce 9.4), přičemž u druhé vyprodukoval překvapivé množství chybných hlášení a zároveň i vyšší časy (sekce 9.3), tudíž je definitivně nutno tuto problematiku rozebrat podrobněji, případně využít alternativního přístupu či software.

Seznam zdrojů

[1] HEINDL, Eduard. *Mobile Network: E-Business Technology* [online]. , 5-18 [cit. 2020-04-24]. No.232493. Dostupné z: https://webuser.hs-furtwangen.de/~heindl/ebte-09ss/Mobile-network.pdf

[2] SAUTER, Martin. *From GSM to LTE-Advanced Pro and 5G: An Introduction to Mobile Networks and Mobile Broadband*. 3rd ed. Hoboken, NJ, USA: Wiley, 2017. ISBN 9781119346906.

[3] BEČVÁŘ, Zdeněk, Pavel MACH a Ivan PRAVDA. *Mobile networks*. Prague: Czech Technical University, 2013. ISBN 978-80-01-05306-5.

[4] RYŠÁNEK, F. PÁSMA LTE/UMTS/EDGE/GSM POUŽÍVANÁ V ČESKÉ REPUBLICE. Průmyslové počítače a komunikace: Průmyslové systémy [online]. 2016 [cit. 2020-05-12]. Dostupné z: http://www.fccps.cz/pasma-lteumtsedgegsm-pouzivana-v-ceske-republice-1379

[5] Územní struktura. Český statistický úřad [online]. 9.1.2020 [cit. 2020-05-12]. Dostupné z: https://www.czso.cz/csu/rso/uzemni_struktura

[6] SOOD, Roopali a Atul GARG. Digital Society From 1G to 5G: A Comparative Study. *International Journal of Application or Innovation in Engineering & Management (IJAIEM)* [online]. 2014, **2014**(3), 186-193 [cit. 2020-04-24]. ISSN 2319-4847. Dostupné z: https://www.researchgate.net/publication/263657708_Digital_Society_from_1G_to_5G_A_Comparative_Study

[7] HUISMAN, Otto a Rolf A. DE BY. *Principles of Geographic Information Systems: An introductory textbook*. 4. 7500 AA Enschede, The Netherlands: The International Institute for Geo-Information Science and Earth Observation (ITC), 2009. ISBN 978-90-6164-269-5.

[8] ČEPICKÝ, Jáchym, Martin LANDA a Ľudmila FURTKEVIČOVÁ. *Úvod do nástrojů Open Source GIS* [online]. 26.11.2019 [cit. 2020-04-28]. Dostupné z: https://training.gismentors.eu/open-source-gis/

[9] FAAS, Jessy. *GIS - Educatie: GIS IN HET VOORTGEZET ONDERWIJS – WAT IS GIS?* [online]. 1.7.2016 [cit. 2020-05-09]. Dostupné z: https://jessyfaas.com/2016/07/01/gis-in-het-voortgezet-onderwijs-wat-is-gis/

[10] *Prostorová Data* [online]. Přírodovědecká Fakulta Univerzity Karlovy, 2014, (1), 1-13 [cit. 2020-04-24]. Dostupné z: https://www.natur.cuni.cz/geografie/geoinformatika-kartografie/ke-

stazeni/projekty/moderni-geoinformacni-metody-ve-vyuce-gis-a-kartografie/prostorova-data/

[11] *Úvod do geografických informačních systémů (GIS): Jednotlivé činnosti v projektu GIS – zpracování a uchování dat* [online]. [cit. 2020-04-29]. Dostupné z: https://kgm.zcu.cz/studium/ugi/elearning/msgisu06s05cz/default.htm

[12] *Úvod do geografických informačních systémů (GIS): Jednotlivé činnosti v projektu GIS – zpracování a uchování dat* [online]. [cit. 2020-04-29]. Dostupné z: https://kgm.zcu.cz/studium/ugi/elearning/msgisu06s04cz/default.htm

[13] ESRI. *ESRI Shapefile Technical Description: An ESRI White Paper* [online]. Červenec 1998 [cit. 2020-04-29]. Dostupné z: https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf

[14] OGC. *Geography Markup Language* [online]. [cit. 2020-04-29]. Dostupné z: https://www.ogc.org/standards/gml

[15] BUTLER, H., M. DALY, A. DOYLE, S. HAGEN a T. SCHAUB. *The GeoJSON Format: RFC 7946* [online]. Srpen 2016 [cit. 2020-04-29]. DOI: 10.17487/RFC7946. Dostupné z: https://www.rfc-editor.org/info/rfc7946

[16] *OGC GeoPackage* [online]. [cit. 2020-04-28]. Dostupné z: https://www.geopackage.org/

[17] BALDRIDGE, John. *ArcGIS 10: What is a Geodatabase?* [online]. In: . The Evergreen State College, 2012, Únor [cit. 2020-04-24]. Dostupné z: https://helpwiki.evergreen.edu/wiki/index.php/ArcGIS_10:_What_is_a_Geodatabase%3F

[18] CHRISTENSSON, P. *JPEG Definition* [online]. [cit. 2020-04-29]. Dostupné z: https://techterms.com/definition/jpeg

[19] GIMOND, Manuel. *Intro to GIS and Spatial Analysis* [online]. 9.8.2019 [cit. 2020-05-09]. Dostupné z: https://mgimond.github.io/Spatial/feature-representation.html

[20] CHRISTENSSON, P. *GIF Definition* [online]. Srpen 2016 [cit. 2020-04-29]. Dostupné z: https://techterms.com/definition/gif

[21] SHICA, Sak a Dr. Kusum GUPTA. Various Raster and Vector Image File Formats. *IJARCCE* [online]. 2015, **4**(3), 268-271 [cit. 2020-04-24]. DOI: 10.17148/IJARCCE.2015.4364. ISSN 22781021. Dostupné z: http://ijarcce.com/upload/2015/march-15/IJARCCE%2064.pdf

[22] ORACLE. *MySQL 5.1 Reference Manual: Including MySQL Cluster NDB 6.X/7.X Reference Guide* [online]. [cit. 2020-04-29]. Dostupné z: https://docs.oracle.com/cd/E17952_01/mysql-5.1-en/gis-data-formats.html

[23] DERCLAYE, Estelle. What is a Database? *The Journal of World Intellectual Property* [online]. 2002, **5**(6), 981-1011 [cit. 2020-04-24]. DOI: 10.1111/j.1747-1796.2002.tb00189.x. ISSN 14222213. Dostupné z: http://doi.wiley.com/10.1111/j.1747-1796.2002.tb00189.x

[24] BHOJARAJU, G. a M.M KOGANURMATH. *Database System: Concepts and Design* [online]. 2003, , 1-19 [cit. 2020-04-24]. Dostupné z: https://www.researchgate.net/publication/257298522_Database_Management_Concepts_and_Design

[25] CORONEL, Carlos, Steven MORRIS a Peter ROB. *Database Systems: Design, Implemetation and Management* [online]. 9. USA: Cengage Learning, 2011 [cit. 2020-04-24]. ISBN 978-0-538-46968-5. Dostupné z: http://corpgov.crew.ee/Materjalid/Database%20Systems%20-%20Design,%20Implementation,%20and%20Management%20(9th%20Edition).pdf

[26] MICROSOFT. *Structured Query Language (SQL)* [online]. 19.1.2017 [cit. 2020-04-29]. Dostupné z: https://docs.microsoft.com/en-us/sql/odbc/reference/structured-query-language-sql?redirectedfrom=MSDN&view=sql-server-ver15

[27] GROLINGER, Katarina, Wilson A HIGASHINO, Abhinav TIWARI a Miriam AM CAPRETZ. Data management in cloud environments: NoSQL and NewSQL data stores. *Journal of Cloud Computing: Advances, Systems and Applications* [online]. 2013, **2**(1) [cit. 2020-04-29]. DOI: 10.1186/2192-113X-2-22. ISSN 2192-113X. Dostupné z: http://journalofcloudcomputing.springeropen.com/articles/10.1186/2192-113X-2-22

[28] BALASUBRAMANIAN, Gowri. *AWS Database blog: Choosing the Right DynamoDB Partition Key* [online]. 20.2.2017 [cit. 2020-05-09]. Dostupné z: https://aws.amazon.com/blogs/database/choosing-the-right-dynamodb-partition-key/

[29] Amazon DynamoDB: Fast and flexible NoSQL database service for any scale. *AWS* [online]. [cit. 2020-04-24]. Dostupné z: https://aws.amazon.com/dynamodb/

[30] DEBRIE, Alex. *DynamoDB, Explained: A Primer on the DynamoDB NoSQL database* [online]. [cit. 2020-04-24]. Dostupné z: https://www.dynamodbguide.com/

[31] *PostGIS: Spatial and Geographic object for PostgreSQL* [online]. [cit. 2020-04-29]. Dostupné z: https://postgis.net/

[32] *Okres dle statistické klasifikace NUTS, LAU* [online]. 21.11.2016 [cit. 2020-05-09]. Dostupné z: https://www.czso.cz/csu/rso/okres-dle-statisticke-klasifikace-nuts-lau

[33] Technical Overview of Virtual Private Networks(VPNs). *International Journal of Scientific Research* [online]. 2013, **2013**(2), 93-96 [cit. 2020-04-24]. Dostupné z: https://www.researchgate.net/publication/274929918_Technical_Overview_of_Virtual_Private_NetworksVPNs/citation/download

[34] Top Five NoSQL Databases and When to Use Them. *IT Business Edge* [online]. [cit. 2020-05-15]. Dostupné z: https://www.itbusinessedge.com/slideshows/top-five-nosql-databases-and-when-to-use-them-07.html