VYSOKÁ ŠKOLA BÁŇSKÁ -
TECHNICKÁ UNIVERZITA OSTRAVA

Hornicko-geologická fakulta
Institut geoinformatiky

ZPRACOVÁNÍ DOPRAVNÍCH DAT V PROSTŘEDÍ PROSTOROVÉ DATABÁZE

bakalářská práce

Autor: Vojtěch Zlý
Vedoucí bakalářské práce: RNDr. Daniela Szturcová, Ph.D.

Ostrava 2014

ZADÁVACÍ PROTOKOL

VŠB - Technická univerzita Ostrava
Hornicko-geologická fakulta
Institut geoinformatiky

Zadání bakalářské práce

Student: **Vojtěch Zlý**

Studijní program: B3646 Geodézie a kartografie

Studijní obor: 3646R006 Geoinformatika

Téma: Zpracování dopravních dat v prostředí prostorové databáze
Processing of Traffic Data in a Spatial Database

Zásady pro vypracování:

Nad prostorovými daty, která představují polohu vozidla při jízdě, proveďte analýzu pohybu vozidla vůči podkladovým mapám.

1. Seznamte se s dopravními daty a s prostředím PostgreSQL/PostGIS.
2. Navrhněte postup předzpracování dat tak, aby bylo možné data ve vhodném formátu ukládat do databáze.
3. Navrhněte datovou strukturu vhodnou pro následné zpracování dat.
4. Poskytnutá data uložte do prostředí PostgreSQL/PostGIS.
5. Navrhněte a implementujte funkce, které vyhodnotí dopravní situaci v zájmové oblasti.
6. Proveďte vyhodnocení dosažených výsledků.

Rozsah grafických prací:
dle potřeby

Rozsah původní zprávy:
30 - 40 stran textu

Seznam doporučené odborné literatury:

http://postgis.refractions.net/
Girardin, F.: Visualising the Pulse of the City, http://www.infovis.net/printMag.php?lang=2&num=190
http://senseable.mit.edu/realtimerome/
http://www.itoworld.com/static/mapping_and_spatial_analysis/ito_map.html

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **RNDr. Daniela Szturcová, Ph.D.**

Datum zadání: 31.10.2013

Datum odevzdání: 30.04.2014

prof. Ing. Zdeněk Diviš, CSc.
vedoucí institutu

prof. Ing. Vladimír Slivka, CSc., dr.h.c.
děkan fakulty

Prohlášení

- ***Celou bakalářskou práci včetně příloh, jsem vypracoval samostatně a uvedl jsem všechny použité podklady a literaturu.***
- ***Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č.121/2000 Sb. - autorský zákon, zejména § 35 – využití díla v rámci občanských a náboženských obřadů, v rámci školních představení a využití díla školního a § 60 – školní dílo.***
- ***Beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně, ke své vnitřní potřebě, bakalářskou práci užít (§ 35 odst. 3).***
- ***Souhlasím s tím, že jeden výtisk bakalářské práce bude uložen v Ústřední knihovně VŠB-TUO k prezenčnímu nahlédnutí a jeden výtisk bude uložen u vedoucího bakalářské práce. Souhlasím s tím, že údaje o bakalářské práci, obsažené v Záznamu o závěrečné práci, umístěném v příloze mé bakalářské práce, budou zveřejněny v informačním systému VŠB-TUO.***
- ***Bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona.***
- ***Bylo sjednáno, že užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).***

V Ostravě dne 29. 4. 2014 ***Vojtěch Zlý***

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat RNDr. Daniele Szturcové, Ph.D. za odborné vedení, trpělivost, ochotu a připomínky při zpracování bakalářské práce.

Anotace

V předložené práci je sepsán návrh a tvorba databáze, import do připravené databáze a zpracování časoprostorových dat. V úvodní části se zabývám základní problematikou, v dalších částech se zabývám programovými aplikacemi a nástroji, které byly využívané při zpracování dané práce. Následně je představen postup práce, který vyústil k potřebnému výsledku. Při implementaci byla použita PostgreSQL databáze s nadstavbou PostGIS. Závěrem je vše shrnuto a jsou sepsány přínosy, které mi to dalo.

Klíčová slova: PostgreSQL, PostGIS, SQL, GIS, dopravní data, prostorová databáze

Summary

The thesis is aimed at database creation and import and processing space-timed data into the prepared database. There are basic issues presented in a first part of the paper, others parts deal with software applications and tools that were used during processing of the paper. Then there are a workflow and work-results introduced. There was used PostgreSQL database with PostGIS extension during the implementation. Conclusion presents final benefits and impact of the paper.

Keywords: PostgreSQL, PostGIS, SQL, GIS, traffic data, spatial database

OBSAH

1 ÚVOD 1
2 APLIKACE A DATA 1
2.1 Aplikace 1
2.2 Data 3
3 PŘÍPRAVA DAT 6
3.1 Instalace PostgreSQL/PostGIS 6
3.2 Seznámení se s datovými typy 7
3.3 Vkládaní dat do databáze 7
3.4 Nastavení jedinečného identifikátoru 9
3.5 Preprocesing 10
3.6 Mazaní duplicit 10
3.7 Partitioning 12
3.8 Nastavení geometrie 14
3.9 Vymezení zájmové oblasti 15
3.10 Propojení tabulek jízdy a polohy 19
3.11 Selekce bodů 20
4 ANALÝZA 24
4.1 Úvod do problematiky 25
4.2 Tvorba analýz 25
4.3 Interpretace analýz 28
5 ZÁVĚR 45
SEZNAM POUŽITÉ LITERATURY 46
SEZNAM OBRAZKŮ 47
SEZNAM GRAFŮ 48
SEZNAM TABULEK 49
PŘÍLOHY 50

SEZNAM ZKRATEK

AJAX - Asynchronous JavaScript and XML
BSD - Berkeley Software Distribution
CPU - Central Processing Unit
DFD - Data flow diagram
EPSG - European Petroleum Survey Group
ERD - Entity-relationship diagram
GB - Gigabyte
GIS - Geographic information systém
GPL - General Public License
HGF - Hornicko-geologická fakulta
JDBC - Java Database Connectivity
NASA - National Aeronautics and Space Administration
ODBC - Open Database Connectivity
OGC - Open Geospatial Consortium
OSM - OpenStreetMap
PC - Personal computer
PHP - PHP: Hypertext Preprocessor
QGIS - Quantum GIS
SED - Stream editor
SQL - Structured Query Language
SRID - Spatial Reference System Identifier
TCP/IP - Transmission Control Protocol/Internet Protocol
VŠB-TUO - Vysoká škola báňská – Technická univerzita Ostrava
WCS - Web Coverage Service
WFS - Web Feature Service
WGS - World Geodetic System
WMS - Web Map Service

1 ÚVOD

Motivem bakalářské práce je zpracování dopravních dat v prostředí prostorové databáze. Veškerá zpracování dat patří k nejdůležitějším úkolům geoinformatika. Podle toho jak jsou data zpracována a strukturovaná, tak jsou i následně hodnocena. Když jsou data správně strukturována a vyhodnocena, tak dané výstupy vedou ke znalostem, díky kterým pak můžeme ovlivňovat řízení dopravy.

Sběrem dopravních dat se zabývá celá řada subjektů. Tyto subjekty data sbírají, ukládají, případně je i zpracovávají. Pokud neprovádějí vlastní zpracování, předávají data dalším subjektů, které s nimi dále pracují.

Tato dopravní data mi byla poskytnuta ke zpracování bakalářské práce. Dopravní data je potřeba v počátku správně strukturovat a to proto, aby bylo možno s nimi dále manipulovat. Když jsou data strukturovaná, bude možno provádět statistické výpočty, prostorové analýzy nebo je publikovat.

2 APLIKACE A DATA

K uskutečnění práce jsem využil řadu aplikací. Když jsem tyto aplikace vybíral, kladl jsem důraz na tzv. open-source software. Když nebylo možno dosáhnout tohoto cíle, tak jsem vybíral aplikace, které byly poskytovány pod freeware, případně pod shareware licencí.

2.1 Aplikace

SED Strem editor je jednoduchý, ale výkonný počítačový program, sloužící k aplikaci rozličných předdefinovaných textových transformací na sekvenční proud textových dat. Po jednotlivých řádcích prochází vstupní soubor, dle pravidel určených v jednoduchém jazyku sedovském skriptu. Každý řádek upraví a poté vypíše. [1]

Notepad ++ je volně šiřitelný editor zdrojových kódu a nahrazení poznámkového bloku. Tento editor je možno použít i v prostředí Windows a jeho použití se řídí GPL licencí. Notepad ++ se dokonce snaží snížit světové emise oxidu uhličitého. Při použití méně energie CPU, může PC přiškrtit a snížit spotřebu energie, což má za následek zdravější prostředí. [2]

CASE Studio 2 je projekční databázový nástroj, který umožňuje vytvářet entity ERD diagramu pro různé databázové systémy. Zahrnoval plnou podporu následujících databází: Oracle, DB2, MS SQL, InterBase, MySQL, PostgeSQL, Sybase ASE, MS Acces, Firebird, Ingres, Informix, V8, DBISAM a několik dalších. Při tvorbě ERD, program považuje za jednotlivé položky databáze například referenční integrity, omezení, domény, triggery a oprávnění uživatelů. Poskytuje galerii pro uložení nejčastěji používané části modelů nebo slovník s předdefinovaných uživatelských datových typů. Kromě toho,

datové toky mezi tabulkami lze také snadno popsal vytvořením vhodných dat diagramů DFD. [3]

PostGIS je rozšíření objektově-relačního databázového systému PostgreSQL pro podporu geografických objektů. PostGIS implementuje specifikaci Simple Features konsorcia OGC. [4]

PostgreSQL je plnohodnotným relačním databázovým systémem s otevřeným zdrojovým kódem. Má za sebou více než sedmnáct let aktivního vývoje a má vynikající pověst pro svou spolehlivost a bezpečnost. Běží nativně na všech rozšířených operačních systémech včetně Linuxu, UNIXů (AIX, BSD, HP-UX, SGI-IRIX, Mac OS X, Solaris, Tru64) a Windows. Stoprocentně splňuje podmínky ACID, plně podporuje cizí klíče, operace JOIN, pohledy, spouště a uložené procedury. Obsahuje většinu SQL92 a SQL99 datových typů, např. INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL a TIMESTAMP. Nechybí ani podpora moderních datových typů jako je JSON nebo XML. [5]

PostgreSQL je šířen pod BSD licencí, která je nejliberálnější ze všech open source licencí. Tato licence umožňuje neomezené bezplatné používání, modifikaci a distribuci PostgreSQL a to ať pro komerční nebo nekomerční využití. PostgreSQL můžete šířit se zdrojovými kódy nebo bez nich, zdarma nebo komerčně. [5]

PostgreSQL umožňuje běh uložených procedur napsaných v několika programovacích jazycích, v Perlu, v Pyhon, v jazyku C nebo v speciálním PL/pgSQL, jazyku vycházejícím z PL/SQL fy. Oracle. Existují PostgreSQL varianty JDBC, ODBC, dbExpress, Open Office, PHP, .NET, Perl nativních rozhraní. K PostgreSQL existuje překladač Embedded SQL pro C a C++. Dále existuje experimentální podpora SQL/PSM - standardizovaného jazyka pro návrh uložených procedur v ANSI SQL. [5]

Předností systému PostgreSQL je rozšiřitelnost. Systém může být bezproblémově rozšiřován o nové datové typy, funkce operátory, agregační funkce, procedurální jazyky. Díky tomu mohly vzniknout následující rozšíření: PostGIS - podpora pro geografické informační systémy, Slony-I - master to multiple slaves replikace. Na serverech pgfoundry nebo PGXN je k dispozici několik desítek doplňků včetně doplňků rozšiřující o funkcionalitu MySQL, SQL Serveru a Oraclu. Uživatelé také oceňují vynikající uživatelskou podporu. [5]

pgAdmin je populární a nejlépe vybavená administrační a vývojářská Open Source platforma pro PostgreSQL, nejpokročilejší Open Source databázi na světě. Aplikaci lze používat pod Linuxem, FreeBSD, Solaris, Mac OSX a Windows systémy ke správě PostgreSQL 7.3 a novějšího běžícího na libovolné platformě, stejně jako komerčních a odvozených verzí PostgreSQL, jako je Postgres Plus Advanced Server a Greenplum database. [6]

pgAdmin je navržený jako odpověď na potřeby všech uživatelů, od psaní jednoduchých dotazů SQL po vývoj komplexních databází. Grafické rozhraní podporuje všechny funkce PostgreSQL a činní administraci snadnou. Aplikace rovněž obsahuje editor SQL se zvýrazňováním syntaxe, editor kódu serverové strany, agenta pro plánování úkolů, podporu pro replikační systém Slony-I a další. Připojení k serveru může probíhat přes TCP/IP nebo Unixové sokety (na nixových platformách) a může být kvůli bezpečnosti šifrované SSL. Pro komunikaci s databázovým serverem nejsou potřeba žádné přídavné ovladače. [6]

pgAdmin je vyvíjen komunitou expertů na PostgreSQL z celého světa a je dostupný ve více jak tuctu jazyků. Jedná se o svobodný software vydaný pod licencí PostgreSQL License. [6]

Quantum GIS je svobodný a multiplatformní geografický informační systém. QGIS je psán v jazyku C++, grafické uživatelské rozhraní je postaveno na knihovně Qt. Zásuvné moduly je možno vytvářet v C++ nebo Pythonu. QGIS umožňuje zejména prohlížení, tvorbu a editaci rastrových a vektorových vrstev, zpracování GPS dat a tvorbu map. Funkčnost rozšiřují zásuvné moduly, významný je modul zpřístupňující funkce GRASS GISu – QGIS tak může sloužit jako jeho nadstavba. [7]

GeoServer je open source webový server napsaný v programovacím jazyce Java umožňující sdílet ale i upravovat geografická data. Projekt klade důraz na interoperabilitu, publikuje data s využitím otevřených standardů v oblasti. GeoServer umožňuje propojení informací poskytovaných z různých zdrojů, jako jsou např. virtuální glóby (Google Earth, NASA World Wind) či webové aplikace postavené na OpenLayers, Google Maps a dalších. GeoServer implementuje standardy jako WMS, WCS či WFS. GeoServer je komunitní projekt. Projekt založila společnost OpenGeo, která se v současnosti zcela zásadním způsobem na vývoji projektu stále podílí. Mezi další společnosti, které do vývoje projektu přispívají, patří např. Refractions či italská společnost GeoSolutions. [8]

2.2 Data

Dopravní data se pořizují pro obsahové hodnoty jejich informací. Díky těmto informacím jsme schopni dosáhnout znalostí, jejímž výsledkem je zachycení dopravní zátěže. Tyto výsledky mohou být následně použity pro řízení dopravy nebo pro koordinaci krizových či mimořádných situací. Tato data přímo pořizují flotily vozidel, která zachycují svou polohy do logu pomocí navigačních zařízení. Tyto logy většinou mají pevně danou strukturu. Tuto strukturu tvoří nějaké id, souřadnice, a časové značky změřené polohy vozidla. Dopravní data mohou být získávána nejen prostřednictví navigačních zařízení, ale také pomocí mýtných brán, nebo indukčních smyček umístěné pod vozovkou.

Data, která mi byla poskytnuta k tvorbě bakalářské práce, obsahovala informace o pohybu flotily vozidel ve střední Evropě. Informace o pohybu této flotily byly zaznamenány v časovém období od roku 2007 do roku 2009. Obdržená data byla přesně

strukturovaná a uložena v textovém formátu *.txt. Jednotlivé hodnoty záznamu byly striktně odděleny oddělovačem tabulátorového typu, což velice ulehčilo další zpracování. Data byla bez jakékoliv dokumentace, což v prvopočátku velice ztěžovalo jejich zpracování. Nebyl jsem totiž schopen určit, jaké informace nesou sloupce, které definují časové hodnoty. U těchto časových záznamů jsem nevěděl, jestli je mám ukládat jako dělený prvek datum a čas s datovými typy date a time. Nebo jako celek s datovým typem timestamp. Avšak po několika neúspěšných importech do databáze a následné nalezení funkce v textovém editoru, která zobrazuje, kde jsou jednotlivé hodnoty záznamu děleny, mi došlo, jaký datový typ mám zvolit.

2007-06-08 08:39:18→	2007-06-08 08:45:24→	50.46304→	15.16934→	50.46751→	15.17661→	2.6↓
2007-06-08 08:36:43→	2007-06-08 08:45:29→	49.02713→	17.66601→	49.03268→	17.64608→	3.4↓
2007-06-08 08:44:11→	2007-06-08 08:45:36→	49.86492→	18.51627→	49.86506→	18.51604→	0↓
2007-06-08 08:44:11→	2007-06-08 08:45:36→	49.86492→	18.51627→	49.86506→	18.51604→	0↓
2007-06-08 08:45:00→	2007-06-08 08:45:09→	49.19695→	16.60569→	49.19695→	16.60569→	0↓
2007-06-08 08:45:00→	2007-06-08 08:45:09→	49.19695→	16.60569→	49.19695→	16.60569→	0↓

Obrázek 1: Screenshot surových dat jízd

2007-06-01 04:14:15→	49.04098→	17.58982↓
2007-06-01 04:14:25→	49.04069→	17.58684↓
2007-06-01 04:14:35→	49.04038→	17.58386↓
2007-06-01 04:14:45→	49.04008→	17.58087↓
2007-06-01 04:14:55→	49.03978→	17.57787↓
2007-06-01 04:15:05→	49.03961→	17.57487↓

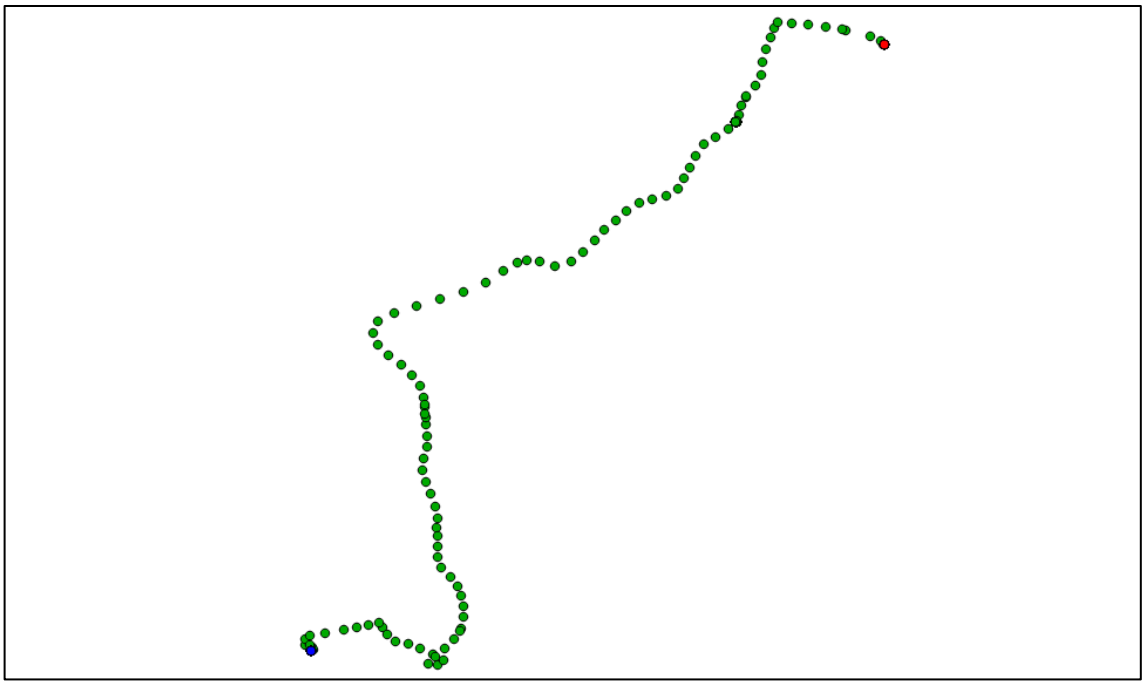
Obrázek 2: Screenshot surových dat poloh

Řádkové záznamy v textovém dokumentu jízd byly děleny tak, že časové značky znázorňovaly, kdy vozidlo vyjelo ze své počáteční pozice a kdy dojelo do požadované lokace. Dvojice souřadnic mi definovaly polohy, odkud vozidlo vyjelo a kde dorazilo. V surové podobě měl textový dokument velikost 1,8 GB.

Podobně tomu bylo i v textových dokumentech s polohami vozidel. Řádkové záznamy byly velice podobné, jen s tím rozdílem, že časová značka a dvojice souřadnic definovaly, kde a kdy se dané vozidlo nacházelo. V surové podobě se velikosti jednotlivých textových dokumentů lišily. Textový dokument, který obsahoval data za rok 2007, měl velikost 7 GB. 21,9 GB měl velikost textový dokument, který nesl data za rok 2008. A poslední z nich, který obsahoval údaje za rok 2009, měl velikost 42,5 GB.

Jejich výsledný SRID jsem nakonec určil jako 4326, což odpovídá souřadnicovému systému WGS 84.

Zde je screenshot z QGISu, který ukazuje průběh jedné náhodně vybrané trasy. Červený bod znázorňuje počátek dané jízdy, modrý konec a zelené body vyznačují průběh samotné trasy, kudy se vozidlo pohybovalo.



Obrázek 3: Screenshot z QGISu náhodně zvolené jízda propojená s polohami

Počet vozidel sbírajících data ve flotile nemohu přesně určit pro opakované poruchy na serveru. Avšak pro malou ukázku, mohu poskytnout údaj o počtu vozidel ve flotile za měsíc červen roku 2008. Flotila činila cirka 1 749 vozidel.

Velikosti textových souboru, které se následně importovaly do databáze, značně rostly při jakékoliv úpravě, např. při nastavení OID, definování geometrie, nebo když jsem prováděl indexaci.

Dalším vstupem, který vstupoval do mé práce, byla OSM data, která poskytující OpenStreetMap. Tato data jsem stahoval pomocí pluginu, který je implementován v QGISu. Stahovaná data se vázala k předem určeným zájmovým oblastem. Dané podkladové informace se uloží do formátu *.osm, ve kterém jsou uložené jako bodové, liniové a polygonové. Jediný limit při stahování je, že mohu najednou stáhnout 50 000 uzlů. Následně jsem si vyexportoval z daného souboru potřebnou liniovou vrstvu představující silnice do shapefilu, který jsem potom mohl nahrát do databáze.

3 PŘÍPRAVA DAT

K hlavnímu řešení práce jsem využíval PostgreSQL/PostGIS a QGIS. PostgreSQL/PostGIS jsem vytvářel prostorové databáze a jednotlivé analýzy. A pomocí QGISu, jsem si zobrazoval jednotlivé výsledky svých postupů.

Prostorová databáze je schopná pracovat s prostorovými daty. Mezi tato prostorová data může patřit například bod, úsečka nebo polygon. Prostor proto znamená, že obsahuje 2 a více dimenzí. Abych mohl pracovat s prostorovou databází, tak musí obsahovat prostorové datové typy a operace.

3.1 Instalace PostgreSQL/PostGIS

a) Instalace PostgreSQL/PostGIS na local

Před samotným zpracováním bylo nejdříve nutné nainstalovat potřebný software. Ke své práci jsem si vybral open-source program PostgreSQL s nadstavbou PostGIS. Daný software jsem instaloval na svůj notebook Lenovo G560, který měl tyto parametry:

Procesor: Intel® Core™ i3 CPU M 370 2,40 GHz
Nainstalovaná paměť: 4,00 GB
Typ grafického adaptéru: NVIDIA GeForce 310M
Typ systému: 64bitový operační systém
Operační systém: Windows 7 Home Premium

PostgreSQL jsem instaloval z balíčku poskytujících na stránkách PostGISu. Samotná instalace aplikace je pro nezkušeného uživatele poněkud složitá. Je nutné, aby si uživatel dával obzvlášť pozor, co a kde instaluje. Já jsem instaloval ke své práci PostgreSQL ve verzi 9.1 a PostGIS ve verzi 2.0.1.

Problémy, které mi vznikly při instalaci, byly zapříčiněny mou nedostatečnou zkušeností s instalací takto rozsáhlých aplikací. Hlavním problémem bylo špatné definování portu pro komunikaci. Při instalaci jsem nechal defaultní hodnoty portu, což mělo za následek nefunkčnost PostgreSQL s dalšími aplikacemi, které už byly nainstalovány před instalací samotného PostgreSQL.

b) Instalace PostgreSQL/PostGIS na server

Při svých pracích na locale jsem po krátké době byl nucen převézt svůj projekt na server v důsledku množství dat, které jsem se pokoušel zpracovat. Proto mi byl poskytnut virtuální školní server, díky kterému jsem mohl dále pokračovat na své práci.

Instalace PostgreSQL/PostGIS na server bylo pro mě velice obtížné. Kompletní instalace společně s Tomcatem, Geoserverem, Apachem a dalšími aplikacemi, včetně nastavení uživatelských profilů, zabrala přes 10 hodin čistého času.

Největší problémem při instalaci byla nedostatečná znalost Linuxu a práce v něm. Během instalace bylo provedeno mnoho chyb, jednou z nich byla např. instalace PostgreSQL/PostGIS z neaktuální verze balíčku.

3.2 Seznámení se s datovými typy

PostgreSQL/PostGIS obsahuje velice mnoho datových typů. Díky tomuto množství je uživatel schopně ideálně popsat všechny objekty nacházející se v databázi. Kompletní výčet těchto datových typů můžeme nalézt v dokumentacích.

Ke své práci jsem využíval tyto datové typy: Integer, Numeric, Character, Timestamp without time zone, Point, LineString, MultiLineString, Polygon, Raster, OID, Serial.

Datový typ	Velikost	Popis	Rozsah
Integer	4 bajty	Typický pro celé číslo	-2147483648 až +2147483647
Numeric	proměnlivé	Uživatelem zadaná přesnost	až 131 072 číslic před desetinnou čárkou až 16 383 číslic za desetinnou čárkou
Serial	4 bajty	Přírůstkové celé číslo v rámci tabulky	1 až 2147483647
OID	-	Přírůstkové celé číslo v rámci databáze	-
Character(n), char(n)	-	Pomocí n definujeme pevnou délku, zapisujeme řetězce	-
Timestamp without time zone	8 bajtu	Ukládá datum a čas bez časových zón	-

Tabulka 1: Požívaných datových typů [9]

Datový typ	Zápis
Point	(X Y)
LineString	$(X_1\ Y_1, X_2\ Y_2, X_2\ Y_3)$
MultiLineString	$((X_1\ Y_1, X_2\ Y_2, X_2\ Y_3),(X_3\ Y_4, X_4\ Y_3, X_5\ Y_6))$
Polygon	$((X_1\ Y_1, X_5\ Y_1, X_5\ Y_5, X_1\ Y_5, X_1\ Y_1),(X_2\ Y_2, X_3\ Y_2, X_3\ Y_3, X_2\ Y_3, X_2\ Y_2))$
Raster	Mnoho hodnot k vyplnění – více v dokumentaci k PostGISu

Tabulka 2: Používaných datových typů pro nastavení geometrie [10]

3.3 Vkládaní dat do databáze

Data jsem se pokoušel do databáze vložit čtyřmi způsoby: pomocí QGIS Importu, PostGIS Import Manageru a dvou SQL příkazu INSERT INTO a COPY.

První dva zmiňované postupy - GIS Import a PostGIS Import se ukázaly vhodné pro vkládání malého množství dat, např. pro vkládání shapefilu obsahující administrativní členění.

3.3.1 QGIS Import

V QGISu je možno data importovat pomocí modulu Správce databází. Tento modul je schopen importovat velkou škálu souboru. Po dokončení importu vnikne v databázi nová tabulka.

Tento způsob jsem využíval, když jsem potřeboval importovat OSM data, nebo jiné prostorové objekty.

3.3.2 PostGIS Import Manager

Přes Import Manager se dají data vkládat pouze ve formátech *.dbf a *.shp, kde po dokončení importu vznikne nová tabulka.

Tento způsob se stal pro vkládání dat nepoužitelný, protože data uložená v textovém formátu přesahovala limit 2 GB, což znamená, že bylo třeba nejdříve dané textové soubory rozdělit na menší soubory a potom z nich vytvořit *.dbf, či *.shp, které jsem mohl následně importovat do databáze.

Dané *.dbf, nebo *.shp jsem vytvářel pomocí programu QGIS. Do něhož jsem importoval textové dokumenty, nad kterými jsem následně definoval geometrii. Když byla geometrie se souřadnicovým systémem nastavena, mohl jsem výsledek exportovat do *.dbf, nebo do *.shp.

3.3.3 INSERT INTO

Dalšími způsobem vkládání dat bylo pomocí SQL dotazu INSERT INTO. Tímto způsobem jsem se snažil vložit jednotlivé záznamy do databáze. Avšak tento způsob se ukázal jako velice nevhodný, protože bylo nutné vytvořit pro každý záznam samostatný dotaz, který by vkládal jednotlivé záznamy.

3.3.4 COPY

Jako poslední přišel na řadu SQL příkaz COPY, který je svou konstrukcí velice jednoduchý. Data se vkládají ze souboru *.csv. Proto bylo třeba pouze rozdělit původní textová data na 2 GB a změnit jejich koncovku z TXT na CSV.

Když byly *.csv soubory připravené pro vkládání, sestavil jsem SQL příkaz, který výsledně vypadal takto:

```
COPY název_tabulky(atributy)
     FROM 'cesta_k_adresáři_na_disku'
          WITH DELIMITER AS 'typ_oddělovače' CSV;
```

3.4 Nastavení jedinečného identifikátoru

Jedinečný identifikátor jsem nastavoval k rozlišení každého záznamu v tabulce. Díky tomuto rozlišení jsem schopen přesně identifikovat jednotlivé záznamy na úrovních jednotlivých tabulek nebo úrovní celé databáze. Tyto identifikátory jsem používal k identifikování záznamů při mazání duplicit.

Jedinečné identifikátory můžu dělit na dva typy a to na SERIAL a OID. Tyto identifikátory definuji v tabulce jako jeden z atributů tabulky, který má právě daný datový typ SERIAL nebo OID.

SERIAL je jedinečný identifikátor pro danou tabulku, což znamená, že se je do tabulky přidá nový atribut s názvem například *gid*, který bude mít datový typ SERIAL. Tím docílíme toho, že každému novému záznamu je přiřazeno jedinečné číslo. Z toho vyplývá, že datový typ SERIAL vytvoří pro daný atribut jedinečné přírůstkové číslo. V praxi to znamená, že první záznam nabývá hodnoty gid = 1 a další je následován záznamem s hodnotou gid = 2.

OID je jedinečný identifikátor v databázi. A to znamená, že určité tabulce se přiřadí atribut s datovým typem OID, který bude obsahovat přírůstkové číslo jedinečné v celé databázi. Následný postup je obdobný jako v příkladu se SERIALem, jen s tím rozdílem, že datový typ je OID. A hodnota oid = 1 je roven prvnímu záznamu v databázi.

Ve svých tabulkách jsem používal datový typ OID a to proto, že bych se mohl dostat do situace, kdy dvě tabulky se stejnou tematickou informací se dostanou do rozepře. Například budu nucen rozdělit tabulku polohy09, kvůli její velikosti a následně nad vzniklými vymazat duplicity. A nastane zmiňovaný problém s dvěma tabulkami se shodnými jedinečnými identifikátory. Když dochází k mazání duplicit, musí být vždy nějaký atribut, který bude jedinečný.

I když datový typ SERIAL jedinečný pro danou tabulku, je možno řešit problém se dvěma tabulkami se stejnou tematickou informací tak, že při volání vypíši celou cestu od databáze přes schéma až po tabulku a tím rozliším jednotlivé tabulky, které mají nastavené své jedinečné identifikátory s datovým typem SERIAL.

Nastavení datového typu OID lze provézt dvojím způsobem. První způsob je tokový, že ho nadefinujeme hned při tvorbě tabulky, nebo ho můžu definovat, až když je tabulka vytvořena. Já jsem OID vytvářel až tehdy, když jsem měl tabulku zhotovená.

Příkaz k přiřazení jedinečného identifikátoru do tabulky vypadá takto:

```
ALTER TABLE název_tabulky SET WITH OIDS;
```

3.5 Preprocesing

Jedná se o přípravu dat k dalším činnostem. Díky preprocesingu se zabývám: vhodným zobrazením dat, mazáním přítomností redundantních informací a filtrací chybných informací. Pokud by preprocesing nebyl řádně proveden, došlo by k chybným výsledkům analýz. Z toho mi vyplývá, že po provedení preprocesingu mi kvalita dat výrazně stoupne a já mohu provádět přesnější a lépe hodnocené analýzy.

3.6 Mazaní duplicit

Mezi prvními problémy, s kterými jsem se musel vyrovnat, bylo omezení programu Notepad ++. Program Notepad ++ není totiž schopen otevřít velké textové dokumenty, díky čemuž není možno využít makra pro mazání duplicit. Musel jsem proto sáhnout po jiných řešeních.

Pro následnou vizualizaci a proces mazání duplicit jsem vybral několik způsobů. A to především pomocí textového editoru EmEditor, streamovacího editoru SED a SQL dotazů.

3.6.1 Mazání duplicit EmEditorem

Jedná se klasický textový editor s velkým množství funkcí pro uživatele. Postup mazání duplicit byl následující: Textový soubor s duplicitními záznamy jsem otevřel v daném editoru, který se načetl do paměti počítače. Po načtení souboru jsem si našel funkci, která maže duplicity a spustil. Jakmile byla operace dokončena, stačilo daný souboru uložit. Nevýhodou tohoto postupu byla délka celého postupu a náročnost na hardware počítače.

3.6.2 Mazání duplicit Streamovacím editorem SED

Jedná se o program, který je součástí Linuxového operačního systému, který je nainstalován na propůjčeném serveru. SED se převáženě požívá pro práci se streamy v shellu. Já jsem ho využíval k zmiňovanému mazání duplicit. Samotná operace se spouští z terminálu.

Mnou spouštěný příkaz vypadal takto:

```
sed -n 'G; s/\n/&&/; /^\([ -~]*\n\).*\n\1/d; s/\n//; h; P'
vybraný_soubor.txt > výsledný_soubor.txt
```

Daný příkaz je složen z několika části usazených mezi jednoduchými uvozovkami. V těchto uvozovkách nalezneme určité sekce kódu oddělené středníky a mezerou, které specifikují určitou operací.

Za jednoduchými uvozovkami nalezneme část kódu, které specifikují, z jakého souboru se budou načítat data a do jakého souboru se bude ukládat výsledek.

3.6.3 Mazání duplicit SQL dotazem

Sestavení dotazu pro mazání jsem prováděl v prostředí programu pgAdmin. Pro mazání duplicit se mi podařilo sestavit dva druhy SQL dotazu.

První vypadá takto:

```
DELETE FROM zdrojová_tabulka
     WHERE jedinečný_identifikátor
          IN (SELECT max (jedinečný_identifikátor)
     FROM název_tabulky
          GROUP BY atributy HAVING count(*) > 1);
```

a smaže pouze jednu duplicitu, kterou nalezne. Z toho vyplývá, že se nejedná o cyklický dotaz, ale má tu výhodu, že je velice rychlý.

Druhý dotaz pro mazání duplicit vypadá takhle:

```
DELETE FROM zdrojová_tabulka
       WHERE OID = ANY (ARRAY (SELECT jedinečný_identifikátor
              FROM (SELECT row_number()
       OVER (PARTITION BY atributy), jedinečný_identifikátor
              FROM zdrojová_tabulka) x WHERE x.row_number > 1));
```

díky tomuto dotazu vymažu všechny duplicity v tabulce.

Pro svou práci jsem zvolil dotaz, který užíval mazání duplicit pomocí cyklu. Kdybych ho nevyužíval, byl bych nucen pouštět dotaz stále dokola, například 30 krát. Redundantní záznamy se v tabulkách často vyskytovaly v řádech desítek a ojediněle i stovek.

Pro představu jak byly operace náročné, jsem provedl operaci pro mazání redundantních záznamu na serveru i na locale. K tomuto porovnání jsem vybral dvě stejně velké tabulky, které jsem měl nahrané v databázi. Mazání duplicit na locale trvalo přibližně 8,5 minuty a na serveru stejné množství dat trvalo pouhé 3 minuty. Ovšem tento časový rozdíl narůstá s množstvím dat uložených v tabulce.

3.7 Partitioning

Po mnoho komplikacích, jsem se rozhodl, že využijí ve své databázové struktuře partitioning. Pomocí partitioningu jsem schopen docílit většího zrychlení zpracování a tím dosáhnout cílených výsledků.

Partitioning dělí tabulky pomocí dědičnosti. Každá tabulka musí být vytvořena jako podřízená tabulka jedné z nadřazené tabulky. Hlavní nadřazené tabulky bývají samy o sobě obvykle prázdné, existují jen proto, aby reprezentovali celou sadu dat. Já jsem dělil jednotlivé tabulky pomocí časových intervalů.

Ve svém příkladu jsem využíval kořenové tabulky, která obsahovala dané atributy a jedinečný identifikátor OID. Z této kořenové tabulky dědily informace další tabulky, které byly rozděleny do měsíčních intervalů. A nakonec z tabulek, které byly rozděleny dle měsíčních intervalů, dědily informace tabulky, které byly rozděleny do denního intervalu. Takhle vypadaly příkazy, které vytvořily danou strukturou:

```
CREATE TABLE název_tabulky (
      atribut datový_typ,
      atribut datový_typ,
      .,
      .
);

CREATE TABLE název_tabulky (
     CHECK (atribut_s_časovou_hodnotou >= časové schéma 'časová_jednotka'
     AND atribut_s_časovou_hodnotou < časové schéma 'časová_jednotka')
) INHERITS (název_kořenové_tabulky);

CREATE TABLE název_tabulky (
     CHECK (atribut_s_časovou_hodnotou >= časové schéma
     ''časová_jednotka' AND atribut_s_časovou_hodnotou < časové schéma
     'časová_jednotka')
) INHERITS (název_kořenové_tabulky);
.
.
.
```

Když byly tabulky rozděleny, bylo třeba je ještě indexovat. K indexaci jsem využil struktury B – tree.

```
CREATE INDEX název_indexu
      ON indexovaná tabulka (atribut_s_časovou_hodnotou);
```

Po vytvoření struktury, bylo třeba už jen tabulky naplnit daty. K tomu posloužil následující SQL příkaz:

```
INSERT INTO tabulka_do_které_se_vkládají_data
      SELECT atributy
            FROM zdrojová tabulka
```

```
WHERE atribut_s_časovou_hodnotou >= časové schéma 'časová_jednotka'
AND atribut_s_časovou_hodnotou < časové schéma 'časová_jednotka';
```

Výsledná databáze, kterou jsem finálně zpracovával, vypadala tak, že v kořenové části byla tabulka doprava2008, která obsahovala veškeré atributy. Z této kořenové tabulky doprava2008 vycházely další tabulky dělené podle měsíce doprava_y2008m01, doprava_y2008m02, doprava_y2008m03 atd. A na závěr z těchto tabulek, dělených dle měsíčních intervalů dědily další tabulky, které byly tříděny podle denních intervalů doprava_y2008m06d01, doprava_y2008m06d02, doprava_y2008m06d03 atd.

3.8 Nastavení geometrie

Když mám naplněné tabulky daty a nastaveny primární klíče s indexy, mohu nastavit geometrií ze sloupců, v kterých se nachází souřadnice. Tento krok nebylo vhodné provádět dříve, než bylo provedeno mazání duplicitních záznamu. S nastavenou geometrií trvá mazání duplicit mnohem déle.

Pro nastavení geometrie jsem vybral atributy, které obsahují souřadnice, ze kterých se bude geometrie vytvářet. A hned při jednom jsem nastavil souřadnicový systém pomocí kódu SRIDu: SRID = 4326. Čísleníky SRIDu jsou definovány pomocí standardů EPSG, což je sada geodetických a kartografických standardů.

SRID 4326 jsem určil pomocí řady testů v QGISu. Daný SRID jsem testoval tak, že jsem nahrál nějakou vrstvu, která byla definována v daném SRIDu a s ní porovnával dopravní data.

Nakonec jsem výsledek uložil do nové tabulky a to proto, abych si nechal původní data jako zálohu.

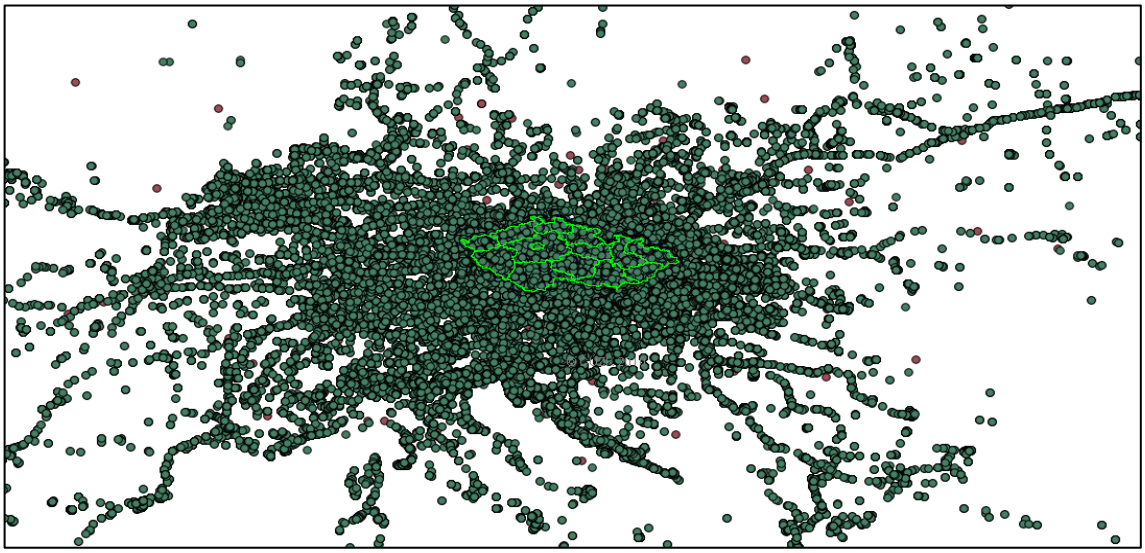
Výsledný dotaz tedy vypadal takto:

```
CREATE TABLE nová_tabulka
      AS (SELECT atributy
            ST_SetSRID   (ST_MakePoint   (atributy_souřadnic),   4326)   AS
            atribut
FROM zdrojová_tabulka);
```

Po tomto dotazu vzniklá nová tabulka s geometrií. To znamená, že klasické souřadnice x, y se převedly na binárního řetězce, který vypadá takto:
0101000020E6100000CC6262F3713D2E405… .

3.9 Vymezení zájmové oblasti

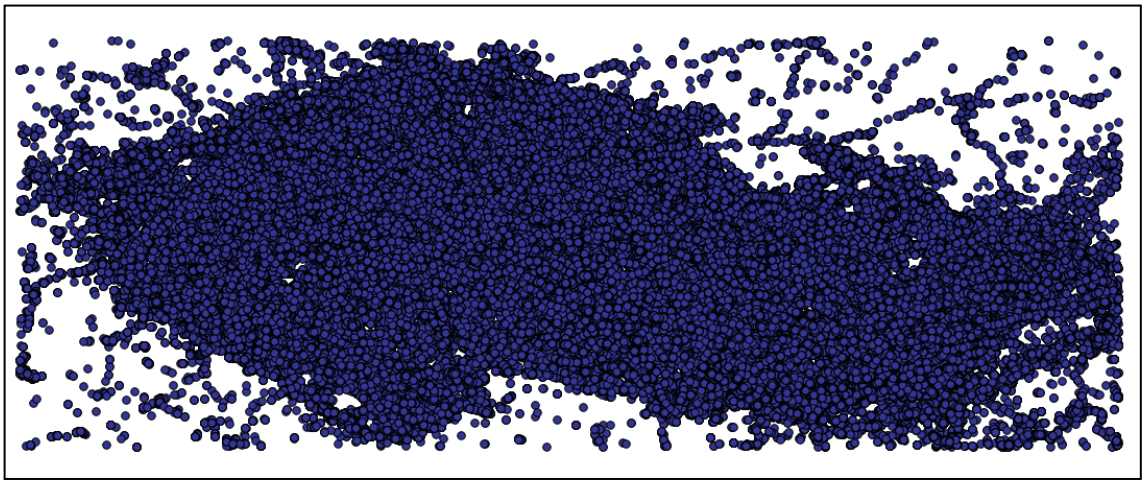
Mně poskytnutá data zahrnují území celé střední Evropy nikoliv data vymezená pro území České reptubliky. Proto bylo třeba tato nadbytečná data odstranit.



Obrázek 4: Screenshot z QGISu, zobrazující mračno bodů tvořící tabulku jízdy

K přesnému vyfiltrování nadbytečných dat bylo třeba sehnat vrstvu s územím České republiky, která byla následně pomocí programu QGIS exportována do databáze. Díky této vrstvy České republiky jsem mohl začít filtrovat nadbytečná data.

Prvním pokusem jsem použil operaci BOUNDERY, díky které jsem doufal, že mi přesně ořízne nadbytečné body podle hranice České republiky. Jenomže má úvaha byla špatná. Po vizualizaci dat jsem zjistil, že body byly oříznuty podle maximálních souřadnic naší republiky, což znamená, že byly sebrány maximální souřadnice severní, jižní, východní a západní hranice a podle nich se ořezávalo. Tímto vznikl ořezaný obdélník, který byl nežádoucí a chybný.



Obrázek 5: Screenshot z QGISu zobrazující vizualizaci metody BOUNDARY

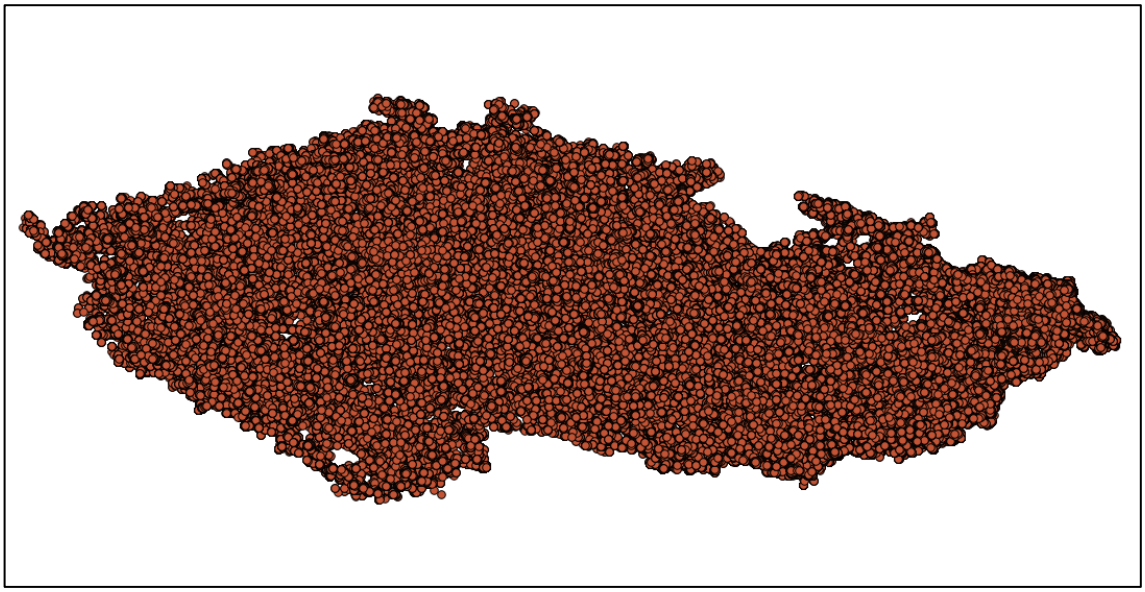
SQL dotaz vypadal takto:

```
CREATE TABLE nová_tabulka
     AS (SELECT atributy
          FROM zdrojová_tabulka, geometrie_hranice_ČR
     WHERE počáteční_geometrie && geometrie_hranice_ČR
          AND koncová_geometrie && geometrie_hranice_ČR);
```

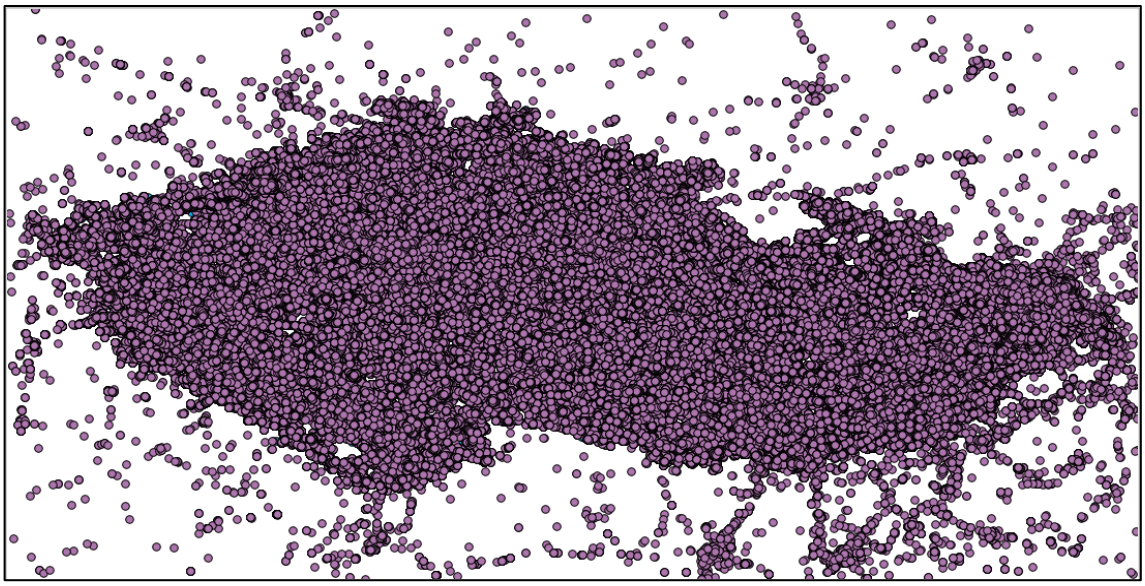
Druhý pokus však už byl úspěšný, protože jsem skombinoval příkaz ST_Intersects s příkazem BOUNDERY. Postup u tabulky jízdy byl takový, že jsem musel postupně ořezávat počáteční body, potom koncové body a na závěr vše sloučit. Tímto jsem vystihl jízdy, které probíhaly mezi sousedními státy. Například vozidlo započalo svou jízdu v zahraničí a končilo v České republice, nebo naopak. Vozidlo započalo svou jízdu v České republice a končilo v zahraničí.

Výsledek jsem vždy ukládal do nové tabulky a to proto, abych měl původní tabulku jako zálohu.

Když jsem ořezával pomocí počátečních bodů, následný grafický výstup vypadal takto:



Obrázek 6: Screenshot z QGISu zobrazující mračno bodů zpracované metodami BOUNDARY a ST_Intersect nad geometrii s počátečními body

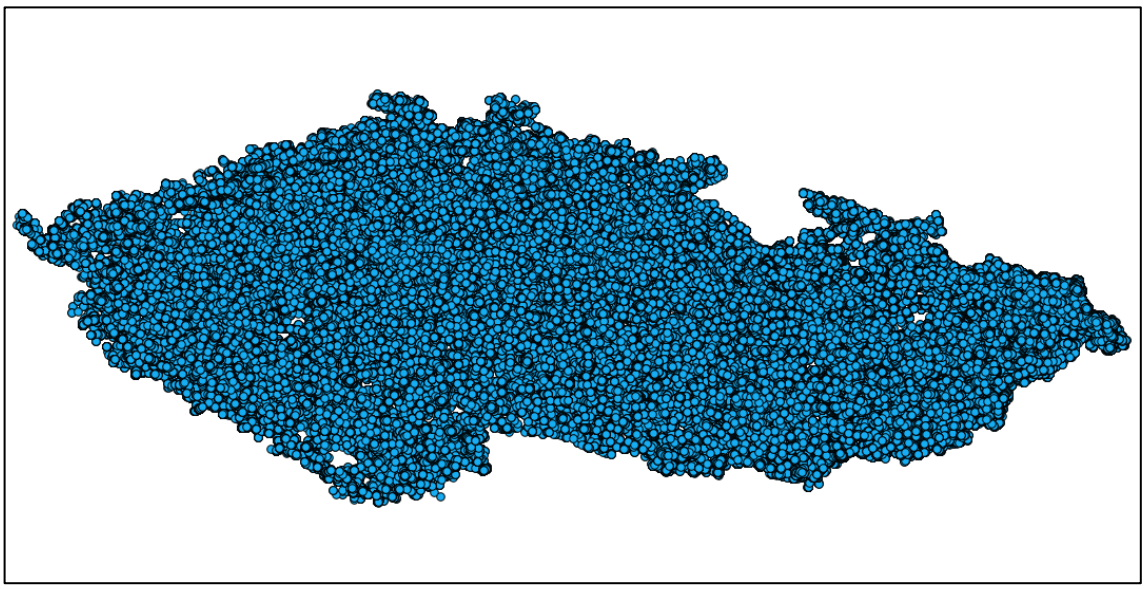


Obrázek 7: Screenshot z QGISu zobrazující mračno koncových bodů vázané na počáteční body

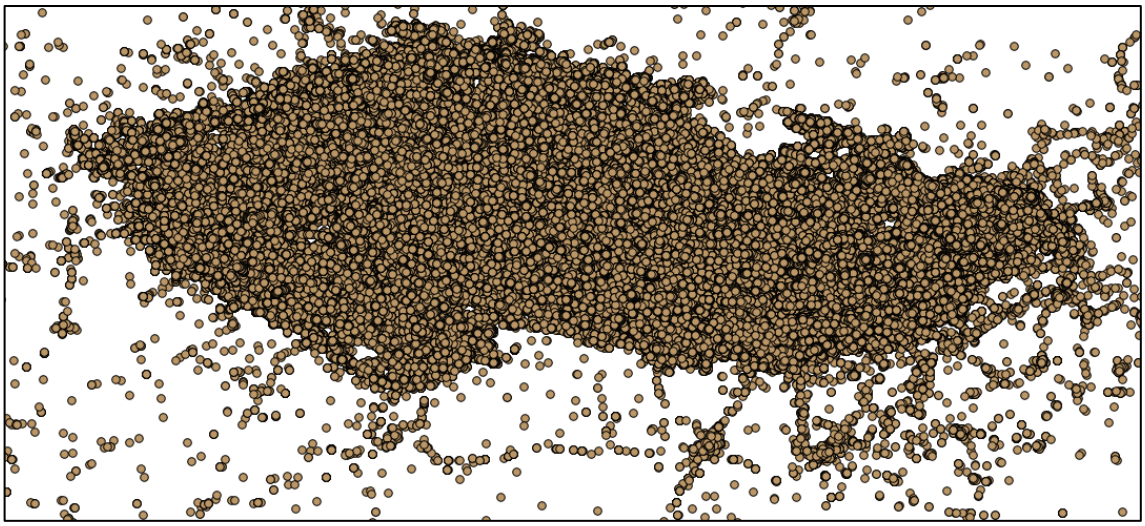
SQL příkaz pro ořezání počátečních bodů vypadal takto:

```
CREATE TABLE nová_tabulka
     AS (SELECT atributy
        FROM zdrojová_tabulka INNER JOIN tabulka_hranice_ČR
     ON (počáteční_geometrie && geometrie_hranice_ČR
        AND ST_Intersects(počáteční_geometrie, geometrie_hranice_ČR)));
```

Grafický výsledek ořezáván pomocí koncových bodů vypadal následovně:



Obrázek 8: Screenshot z QGISu zobrazující mračno bodů zpracované metodami BOUNDARY a ST_Intersect nad geometrii s koncovými body



Obrázek 9: Screenshot z QGISu zobrazující mračno počátečních bodů vázané na koncové body

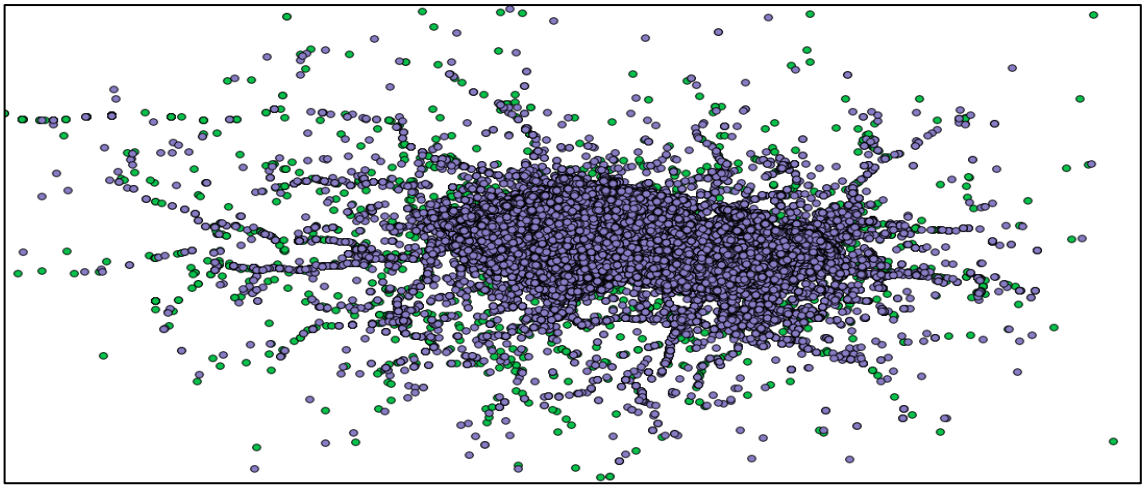
SQL příkaz pro ořezání koncových bodů vypadal takto:

```
CREATE TABLE nová_tabulka
     AS (SELECT atributy
        FROM zdrojová_tabulka INNER JOIN tabulka_hranice_ČR
     ON (koncová_geometrie && geometrie_hranice_ČR
        AND ST_Intersects(koncová_geometrie, geometrie_hranice_ČR)));
```

Na závěr SQL příkaz pro sloučení ořezaných počátečních a koncových bodů:

```
CREATE TABLE nová_tabulka
        AS(SELECT atributy FROM tabulka_ořezaných_počátečních_bodů
                UNION
        SELECT atributy FROM tabulka_ořezaných_koncových_bodů);
```

Díky tomuto poslednímu kroku jsem dostal jednu velikou tabulku jízd, v které se nachází geometrie počátečních a koncových bodů. Grafická vizualizace potvrzuje vystižení jízd i mezi sousedními státy a Českou republikou.



Obrázek 10:Screenshot z QGISu zobrazující mračno bodů propojením tabulek

Pro porovnání jsem tuto operaci provedl na serveru a na locale. Výsledný časový rozdíl byl ohromný. Daná operace na lokále trvala přibližně 10 hodin a na serveru pouhých 29 minut. Zde je zde viditelné, o kolik bylo zpracování na serveru rychlejší.

3.10 Propojení tabulek jízdy a polohy

Pro další práci je potřeba data taky propojit, tím mám na mysli propojení tabulek jízd a tabulek poloh bodů. Tato operace se dělá proto, abych mohl v následných pracích využívat složitějších postupů.

Záznamy v tabulkách obsahovaly vždy ID jednotlivých jízd a časové hodnoty poloh vozidel, díky kterým jsem byl schopen tato data přesně a úspěšně propojit. SQL dotaz propojení vypadal následovně:

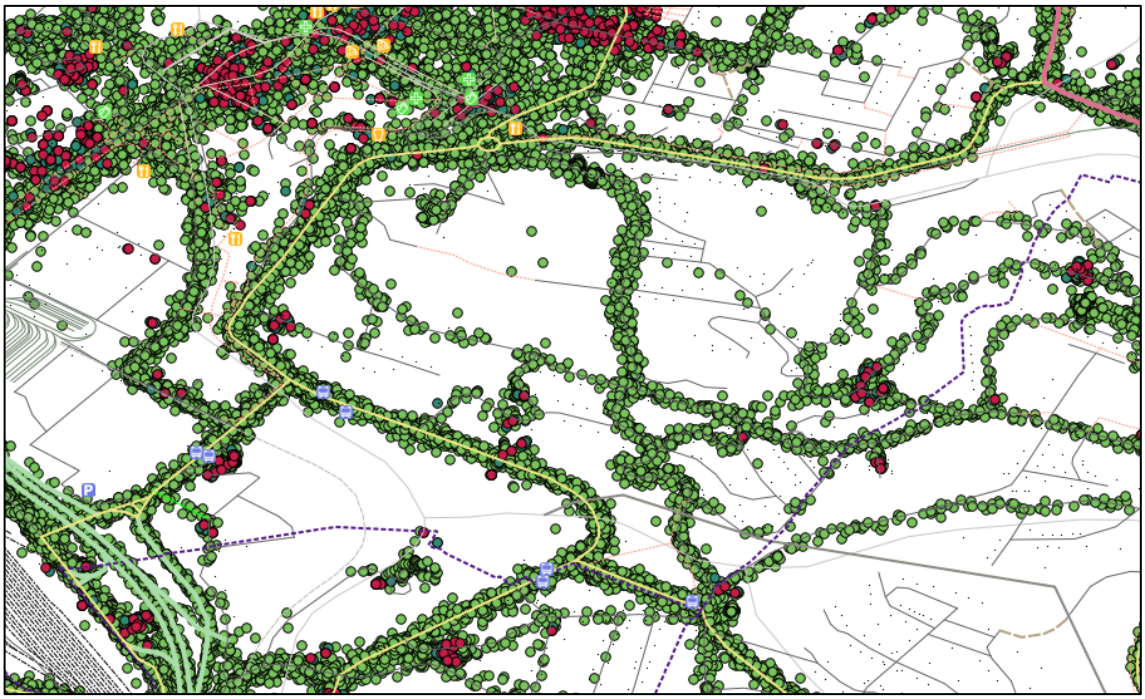
```
SELECT atributy FROM tabulka_jízd, tabulka_poloh
      WHERE ID_jízd = ID_poloh
            AND polohy_časový_atribut
                  BETWEEN jízdy_časový_atribut1
                        AND jízdy.časový_atribut2;
```

jid integ	datumcas1 timestamp without time	datumcas2 timestamp without time	pocatek geometry	konec geometry	delk num	pid integ	datumcas timestamp without time	poloha geometry
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:48:21	0101000020E6100000D47D0052
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:48:39	0101000020E610000062D68BA1
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:48:30	0101000020E6100000D47D0052
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:48:41	0101000020E61000001B2AC6F9
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:52:11	0101000020E610000039D6C56D
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:52:48	0101000020E61000000E677E35
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:50:50	0101000020E6100000713D0AD7
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:50:40	0101000020E6100000713D0AD7
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:51:39	0101000020E61000001CD31396
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:51:30	0101000020E61000000DC347C4
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:51:00	0101000020E6100000548CF337
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:51:10	0101000020E6100000548CF337
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:51:20	0101000020E6100000AA825149
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:52:01	0101000020E610000038109205
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:50:30	0101000020E6100000548CF337
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:50:39	0101000020E61000008DEE2076
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:49:40	0101000020E61000007F6ABC74
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:50:20	0101000020E6100000AA825149
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:51:50	0101000020E61000008E40BCAE
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:50:10	0101000020E6100000046257502
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:51:40	0101000020E6100000D5264EEE
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:49:50	0101000020E61000001B0DE02D
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:52:40	0101000020E6100000553002AA9
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:50:00	0101000020E61000002A745E63
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:52:41	0101000020E61000000E846401
1457	2008-06-01 19:48:21	2008-06-01 19:53:33	0101000020E6100000D47D00	0101000020E61000000E677E3507	0.4	1457	2008-06-01 19:52:31	0101000020E6100000B9C7D287

Obrázek 11: Screenshot z databáze, zobrazující datový výstup

3.11 Selekce bodů

Pro další pokročení práce bylo třeba vymyslet způsob jak selektovat body okolo linií, protože body se nenacházely přesně nad vybranými silnicemi, nýbrž jsem body nacházel i uprostřed obytných budov, průmyslových objektů, či veřejných prostranství. Proto bylo nutno selektovat a přiřazovat jednotlivé body k daným silnicím. Tuto operaci lze provézt mnoha způsoby. Já jsem vyzkoušel selekci pomocí vektorových bufferu, rastrových bufferu a pomocí metody ST_DWithin.



Obrázek 12: Screenshot z QGISu, zobrazující mračno bodů na nežádoucích místech

3.11.1 Selekce pomocí vektorového bufferu

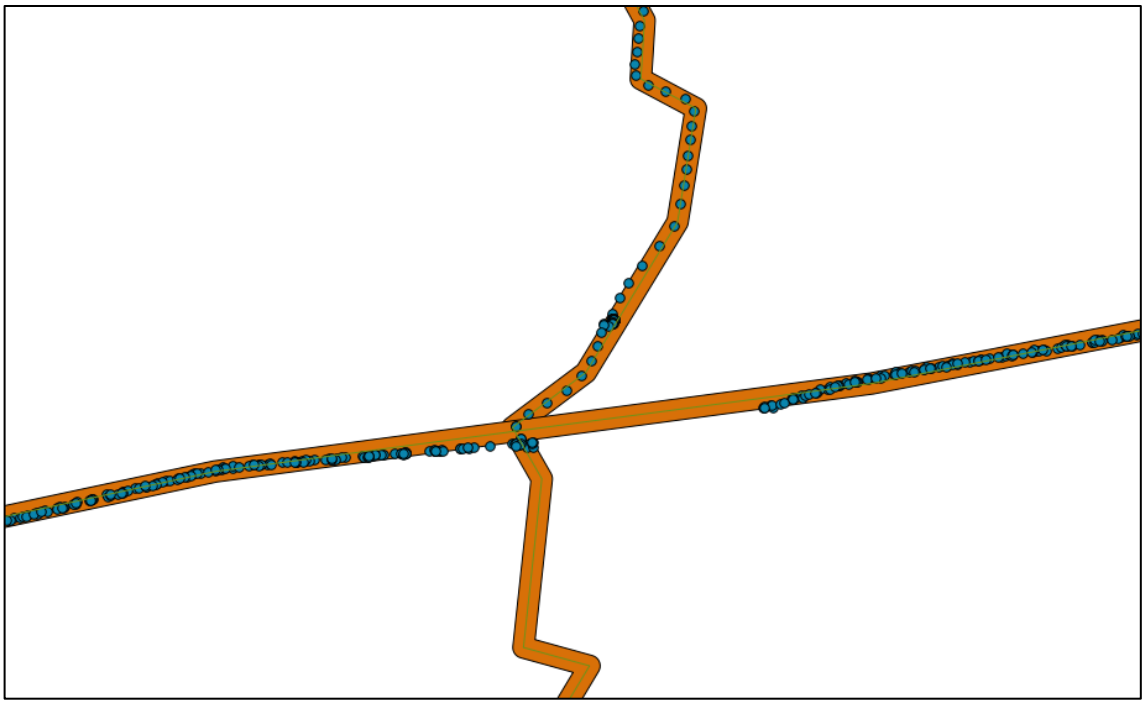
Mezi mé první pokusy patřily operaci s vektorovými buffery. Proto jsem se snažil vytvořit kolem vybraných linií buffer o jisté šířce. SQL příkaz k tvorbě bufferu vypadal následovně:

```
SELECT atributy, ST_Buffer(atribut_s_geometrii, šířka_bufferu,
'nastavení_bufferu') AS alias
      FROM zdrojová_tabulka;
```

a následná selekce bodů, které spadly do daného bufferu:

```
SELECT atributy
      FROM tabulka_s_body INNER JOIN tabulka_s_bufferem
            ON (atribut_geometrie_s_body && atribut_geometrie_s_bufferem
            AND ST_Intersects(atribut_geometrie_s_body,
            atribut_geometrie_s_bufferem));
```

Grafická vizualizace vektorových bufferu vypadala následovně:



Obrázek 13: Screenshot z QGISu, zobrazující vektorový buffer

Oranžovou barvou je vyobrazen buffer kolem zelené silniční komunikace. Modrou barvou jsou vyobrazené body polohy vozidel, které splnily podmínku, jež definovala šířku bufferu dle typu silnice.

Postup pomocí vektorového bufferu byl dobrý do doby, dokud jsem nenarazil na situaci, že dvě silnice vedly relativně blízko sebe. V tomto případě docházelo ke sloučení bufferu do jednoho velikého polygonu a tím následovaly problémy s přiřazováním bodů ke správnému polygonu.

Proto jsem se pokoušel nalézt jiné vhodné řešení.

3.11.2 Selekce pomocí rastrového bufferu

Jako druhý pokus byla práce s rastrovými buffery. Tvorba rastrových bufferu je mnohem složitější oproti jeho kolegovi vektorového bufferu. Proto jsem jeho tvorbu přijal jako výzvu, i když po radě testů jsem usoudil, že jeho využití prozatím nebude vhodné. A tady je SQL příkaz k tvorbě rastrového bufferu:

```
SELECT ST_AsRaster(
            ST_Buffer(
                  (SELECT atribut_geometrie
                        FROM zdrojová_tabulka), šířka_bufferu,
                  'nastavení_bufferu'), šířka_bufferu,délka_bufferu,
                  ARRAY[typ_pixelu],
                  ARRAY[RGB_barva_bufferu], ARRAY[RGB_barva_pozadí]));
```

selekce bodů pak vypadá takto:

```
SELECT atributy
      FROM tabulka_s_body INNER JOIN tabulka_s_rastrem
            ON (atribut_geometrie_s_body && (
            SELECT ST_Polygon(atribut_geometrie_s_bufferem)
                  FROM tabulka_s_rastrem) AND
            ST_Intersects(atribut_geometrie_s_body, (
            SELECT ST_Polygon(atribut_geometrie_s_bufferem)
                  FROM tabulka_s_rastrem)));
```

S vizualizací rastrových bufferu jsem měl poměrně veliké problémy, protože QGIS ve verzi, kterou jsem používal, je neuměl vizualizovat. Proto jsem byl nucen si napsat vlastní skript, který mi ve webovém prohlížeči zobrazil můj výsledný rastr.

Při tvorbě rastrových bufferu jsem se setkal s úskalím ohledně jeho složitosti. Ve verzi PostGISu, kterou jsem využíval, byla tvorba rastrových bufferu novinkou a nebylo v ní možno jednoduše vytvářet složitější buffery. Tím jsem pochopil, že když jsem chtěl vytvořit buffer nad křižovatkou, tak jsem měl smůlu, protože si nedokázal poradit s uzlem na křižovatce.

3.11.3 Selekce pomocí speciálních metod

Posledním pokusem jsem vybíral body pomocí metod ST_DWithin a ST_Distance. Dotaz této selekce vypadá takto:

```
SELECT alias.atributy, alias.atributy,
      ST_Distance(alias.atribut_geometrie_s_linii,
            alias.atribut_geometrie_s_body) AS alias
      FROM tabulka_s_linií AS alias, tabulka_s_body AS alias
WHERE ST_DWithin(alias.atribut_geometrie_s_linii,
      alias.atribut_geometrie_s_body, šířka_výběru)
      ORDER BY ST_Line_Locate_Point(alias.atribut_geometrie_s_linii,
            alias.atribut_geometrie_s_body),
            ST_Distance(alias.atribut_geometrie_s_linii,
                  alias.atribut_geometrie_s_body);
```

Při tvorbě tohoto dotazu jsem narážel na řadu úskalí. Mezi největší z nich bylo rozeznání směru, kudy se vozidlo pohybovalo. Pro rozeznání směru jízd vozidel bylo třeba vytvořit vnořený dotaz, který mi sloučil jízdy se stejným id a časem zahájení jízd, které jsem následně řadil podle poloh vozidel. Poté jsem zjišťoval, ke které silnici budou přiřázeny.

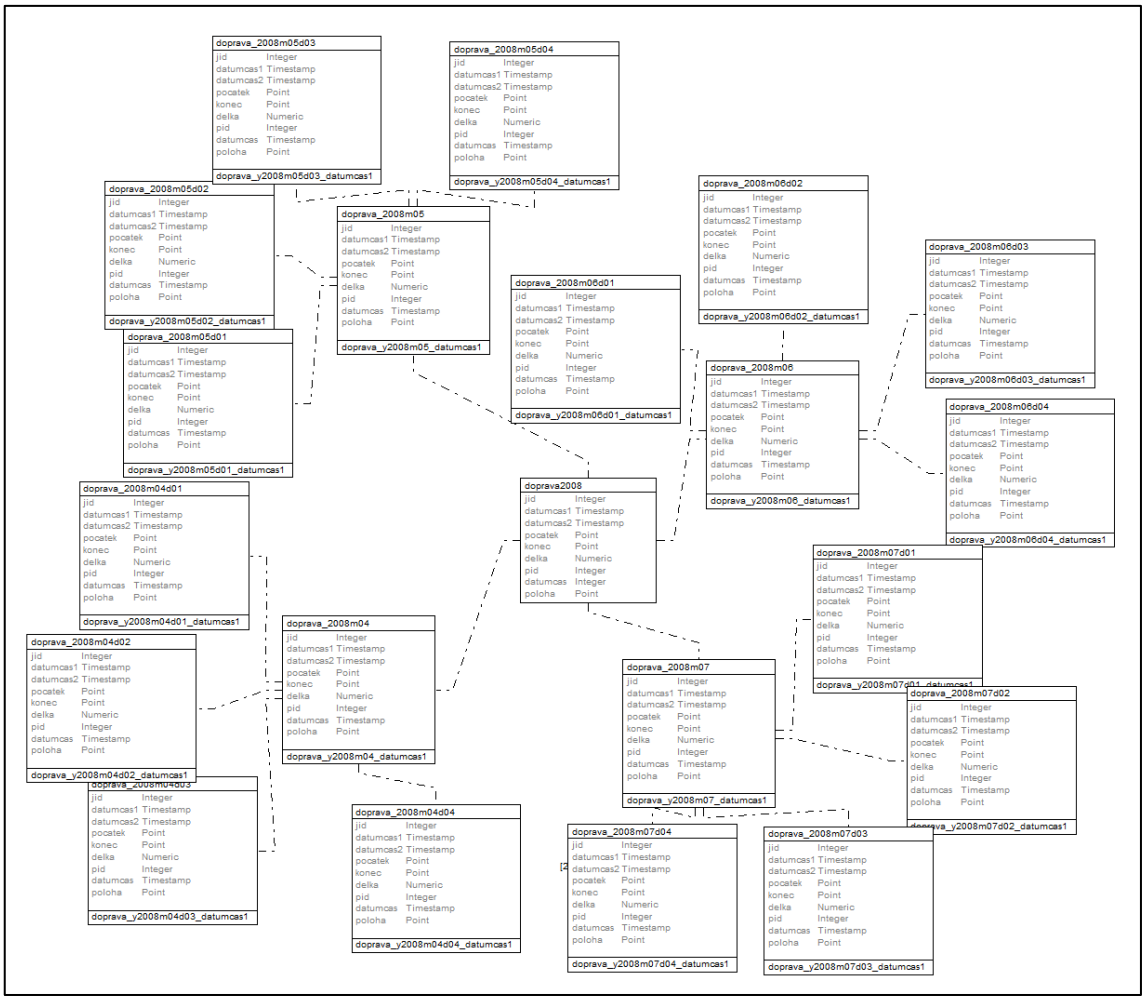
Obrázek 14: Screenshot z QGISu, zobrazující body nad jednotlivými směry silnice

Z těchto pokusů jsem použil ke své další práci poslední zmiňovaný postup s metodami ST_DWithin a ST_Distance, poněvač se mi zdál nejjednodušší a hlavně byl velice rychlý.

4 ANALÝZA

V následné části se budu zabývat zpracováním a vyhodnocením statistických analýz dopravních dat, kterou mám uloženou v prostředí prostorové databáze. Tato data prošla řadou operací, které je strukturovaly a upravovaly.

Samotná data před finálním analyzováním jsou strukturovaná do tabulek pomocí partitioningu. Pro lepší představu jsem vytvořil ukázku pomocí datového modelu, který zachycuje část celé struktury. Kompletní struktura zachycuje veškeré měsíce a dny v roce 2008.



Obrázek 15: Screenshot datového modelu

Nyní se budu zabývat tvorbou a implementací funkcí v prostředí databáze, které budou počítat následné analýzy. Veškeré funkce byly implementovány pomocí SQL jazyka.

4.1 Úvod do problematiky

Pro své analýzy jsem zpočátku měl na serveru připravené a zčásti už i rozpracované analýzy s daty pro území celé České republiky za roky 2007, 2008, 2009. Osud mi však nepřál. Stala se nehoda v podobě kolapsu serveru. Došlo k poškození diskového pole, a aby to nebylo málo, taky k podezření na poškození základní desky, nebo sběrnice. Následkem byla nejen ztráta dat, ale i mnoho odpracovaného času. Proto jsem byl nucen převézt veškerou práci zpět na svůj notebook. Bohužel zálohy ze serveru jsem měl pouze v podobě SQL dotazu, nikoliv dat, kvůli jejich velkému objemu, které jsem zpracovával. Pro tvorbu záloh by bylo totiž potřeba odděleného serveru s dostatečnou kapacitou, kterou jsem neměl k dispozici. K tvorbě záloh mi byl nabídnut pouze externí disk, ale jeho zprovoznění se nezdařilo. Po migraci práce ze serveru na notebook bylo třeba analyzovat, kolik dat bude schopen můj notebook zpracovávat. Po řadě testů jsem vyhodnotil situaci tak, že budu zpracovávat pouze malý vzorek dat oproti původnímu plánu. Pro následné analýzy jsem si vyčlenil území Moravskoslezského kraje, v kterém jsem si vybral několik silnic různých kategorií. Těmito rozdílnými kategoriemi jsem se snažil o pokrytí silnic různých tříd. Mezi vybrané vozovky patřily: silnice 648 – na trase Frýdek-Místek, Nový Jičín a zpět, znova silnici 648 – ale ve směru Frýdek-Místek, Český Těšín a zpět, R56 – na trase Frýdek-Místek, Ostrava a zpět, 11 – Rudná na trase Ostrava – Jih, Svinov a zpět a 11 – Opavská na trase Ostrava, Opava a zpět.

4.2 Tvorba analýz

Pro samotnou tvorbu analýzy bylo třeba si všechno pečlivě promyslet a rozvrhnout, co a jak bude vstupovat do výpočtu.

Na začátku bylo třeba si udělat v databázi pořádek, připravit data a pustit se do práce. Samotná analýza se vytvářela z několika kroků:

1) Vytvoření tabulky se zájmovým územím.
2) Vytvoření tabulky s daty přiřazenými k dané silnici.
3) Přiřazení OID tabulkám.
4) Kontrolnímu vymazání duplicit.
5) Tvorba tabulky, kde se budou zapisovat výsledky statistických analýz.
6) Výpočet analýzy.
7) Vložení výsledků do tabulky.
8) Kontrolní zobrazení.

ad 1) Tabulku se zájmovým územím jsem vytvářel kvůli tomu, abych omezil množství nepotřebných dat, která by vstupovala do výpočtu.

Zájmovou oblast jsem vytvořil tak, že nad vybranou silnici jsem nakreslil polygon. Tento polygon jsem zkonstruoval v QGISu a následně jsem ho importoval do databáze. Když byl polygon v databázi, mohl jsem selektovat body v daném území nad zvolenou

linií. Díky této selekci jsem snížil počet nepotřebných bodů vstupujících do výpočtu. Následný příkaz pak vypadal takto:

```
CREATE TABLE vyber_frydek_t_t26 AS (
      SELECT dop.jid, dop.datumcas1, dop.datumcas2, dop.pocatek,
      dop.konec, dop.delka, dop.pid, dop.datumcas, dop.poloha
            FROM doprava_y2008m06 AS dop
                  INNER JOIN box_frydek AS box
            ON (dop.poloha && box.geom
                  AND ST_Intersects(dop.poloha, box.geom))
);
```

Tvorbu této zájmové oblasti lze provézt také jinými způsoby, například pomocí bounding boxu v PostGISu metodou `ST_MakeEnvelope`, do které se zapisují parametry se souřadnicemi. Souřadnice symbolizují vrcholy čtvercového nebo obdélníkového polygonu zájmové oblasti. Nebo pomocí bufferu, který by byl vytvořen kolem zkoumaného úseku silnice.

Postup pomocí tvorby vlastního polygonu a následného importu do databáze jsem si zvolil díky jeho jednoduchosti. Takhle jsem si, dle vlastního uvážení, vytvořil polygon a importoval. Kdybych pracoval s bounding boxem musel bych pracně přepisovat souřadnice do SQL dotazu, kde při přepisu by mohla nastat chyba, v zaměně číselné hodnoty v souřadnicích. Při tvorbě bufferu bych musel zase zkoumat jeho ideálně vhodnou šířku.

ad 2) Další tabulku, kterou jsem vytvářel, byla tabulka, v které jsem přiřazoval jízdy k určitým silnicím. Více o této selekci jsem popisoval výše v kapitole 4.10.3 Selekce pomocí speciálních metod.

```
CREATE TABLE vybrane_body_frydek_t_y2008m06t26 AS (
      SELECT kon.datumcas1, jiz.jid, jiz.datumcas, jiz.datumcas2,
      jiz.poloha, jiz.pocatek, jiz.konec, jiz.delka, jiz.pid, trasa.id,
      ST_Distance(trasa.geom, jiz.poloha) AS vzdalenost
            FROM cesta_frydek AS trasa, (
                  SELECT vyber_frydek_t_t26.datumcas1, cesta_frydek.geom
                        FROM vyber_frydek_t_t26, cesta_frydek
                  GROUP BY vyber_frydek_t_t26.datumcas1,
                  cesta_frydek.geom) AS kon, vyber_frydek_t_t26 AS jiz
            WHERE ST_DWithin(trasa.geom, jiz.poloha, 0.00005)
                  AND kon.datumcas1 = jiz.datumcas1
            ORDER BY datumcas,
                  ST_Line_Locate_Point(trasa.geom, jiz.poloha),
                  ST_Distance(trasa.geom, jiz.poloha));
```

ad 3) Následně jsem tabulkám přiřazoval jedinečné identifikátory OID. Popis a smysl OIDček je zase popsán výše v předešlé kapitole 3.4 Nastavení jedinečného identifikátoru.

```
ALTER TABLE vybrane_body_frydek_t_y2008m06t26 SET WITH OIDS;
```

ad 4) Pro jistotu, když byly tabulky připravené pro výpočet analýz, provedl jsem kontrolní vymazání duplicit. Popis mazání duplicit je popsán v předešlé kapitole 3.5.3 Pomocí SQL dotazu

.

```
DELETE FROM vybrane_body_frydek_t_y2008m06t26
      WHERE OID = ANY(ARRAY(SELECT OID FROM (
            SELECT row_number() OVER (
                  PARTITION BY jid, datumcas1, datumcas2, delka), OID
                  FROM vybrane_body_frydek_t_y2008m06t26) x
            WHERE x.row_number > 1));
```

ad 5) Následně jsem vytvořil velice důležitou tabulku, do které se bude zapisovat výsledná analýza.

```
CREATE TABLE analyzy_frydek_t_y2008m06t26 (
      id1 integer,
      id2 integer
);
```

ad 6) Když bylo vše připravené, mohl jsem se konečně pustit do výpočtu analýz. Zabýval jsem se analýzami počtu průjezdů flotily vozidel v denní a týdenní horizontu. Samotný výpočet počtu průjezdů pak vypadal takto:

```
WITH mezivypocet_01 AS (
      SELECT datumcas1, datumcas2, id FROM
vybrane_body_frydek_t_y2008m06t26 GROUP BY datumcas1, datumcas2, id ORDER
BY datumcas1
), mezivypocet_2_01 AS (
      SELECT fry_01.jid, fry_01.datumcas1, fry_01.datumcas2,
      fry_01.pocatek, fry_01.konec, fry_01.delka, fry_01.datumcas,
      fry_01.poloha, fry_01.pid, mez_01.id
            FROM vyber_frydek_t_t26 AS fry_01, mezivypocet_01 AS mez_01
      WHERE fry_01.datumcas1 = mez_01.datumcas1
            AND fry_01.datumcas2 = mez_01.datumcas2
), vypocet_1_01 AS (
      SELECT count(datumcas1), id AS id1
            FROM mezivypocet_2_01
      WHERE id = 1 AND datumcas >= '2008-06-23 00:00:00'
            AND datumcas < '2008-06-24 00:00:00'
      GROUP BY datumcas1, id
), vypocet_2_01 AS (
      SELECT count(datumcas1), id AS id2
            FROM mezivypocet_2_01
      WHERE id = 2 AND datumcas >= '2008-06-23 00:00:00'
            AND datumcas < '2008-06-24 00:00:00'
      GROUP BY datumcas1, id
), vypocet_3_01 AS (
      SELECT count(id1) AS id1 FROM vypocet_1_01
), vypocet_4_01 AS (
      SELECT count(id2) AS id2 FROM vypocet_2_01
)
```

ad 7) Když mám výpočet analýzy hotov, tak výsledky uložím do tabulky, kterou jsem předem už vytvořil.

```
INSERT INTO analyzy_frydek_t_y2008m06t26 (
      SELECT vypocet_3_01.id1, vypocet_4_01.id2
            FROM vypocet_3_01, vypocet_4_01);
```

ad 8) Na závěr, když mám analýzu vypočtenou, nechám si ji zobrazit, abych věděl, jak výpočet proběhl a jaké hodnoty byly vypočteny.

```
SELECT * FROM analyzy_frydek_t_y2008m06t26;
```

4.3 Interpretace analýz

Pro interpretaci statistických výsledků je ideální vytvořit nějaké grafy, případně je zanést do mapy. Já jsem si pro svou publikaci výsledků rozhodl využít Microsoft Excel, kde vlastně mohu výsledné hodnoty vložit do tabulky a následně i z dané tabulky vytvářet grafy.

Do budoucna mám v plánu vytvořit vizualizace svých výsledků v Rku, případně vytvořit webovou mapovou aplikaci využívající Open Layers, v které se budou vizualizovat jednotlivé počty průjezdů v jednotlivých časových intervalech.

Ve své práci jsem počítal čtyři druhy statistických analýz. Počítal jsem počty průjezdu vozidel nad vybranými komunikacemi v jednotlivých dnech a týdnech, díky čemuž jsem se mohl dále zabývat hodinovou dopravní zatížeností a vývojem počtu vozidel v jednotlivých dnech. Výsledky statistických analýz jsem po dokončení SQL dotazu vždy uložil do nové tabulky. Tyto nové tabulky jsem vkládal do Excelu a vytvářel dvourozměrné sloupcové grafy.

Díky těmto grafům jsem byl nakonec schopen určit statistické závěry. Na některých vybraných silničních komunikacích však nastával problém s nedostatkem dat, převážně v obdobích víkendu. Tento problém vznikl kvůli tomu, že žádné vozidlo z celkové flotily se na dané silniční komunikaci neprojíždělo, proto nebudu tyto výsledky vyobrazovat v závěrečném popisu analýz, vyberu jen několik analýz, z každé zkoumané silniční komunikace.

Mezi první trasy jsem začal analyzovat silniční komunikaci mezi Frýdkem-Místkem a Ostravou. Jednalo se o silnici R56. Prostorové okno jsem měl vytvořené na úrovni průmyslové zóny, více ve screenshotech z QGISu, kde jsem si své výsledky vizualizoval. Analyzoval jsem vždy oba směry jízd na silniční komunikaci.

4.3.1 Počty průjezdů vozidel v hodinových intervalech

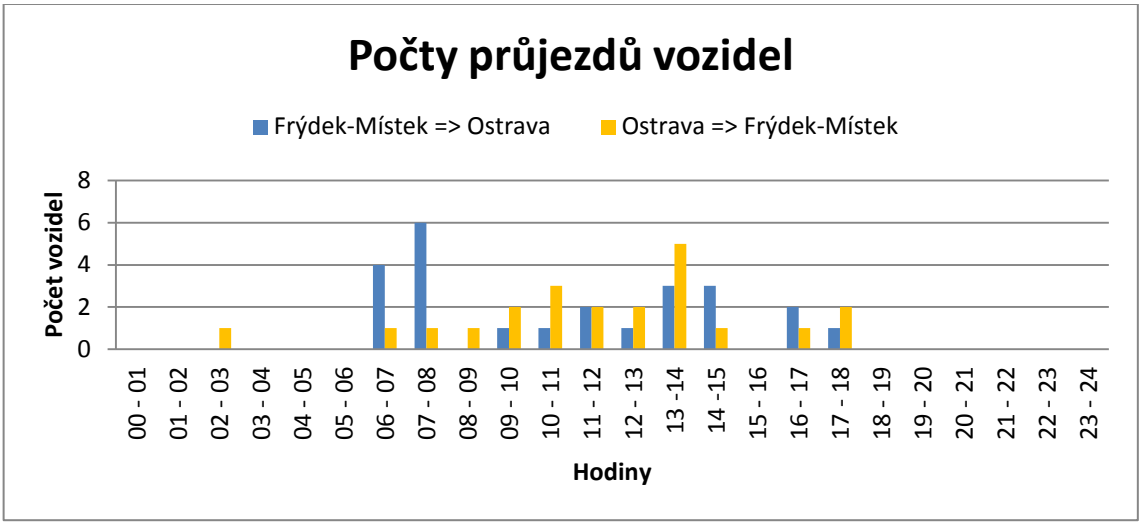
Dané počty průjezdů vozidel v hodinových intervalech mi bude zachycovat, kolik vozidel se pohybovalo po vybrané silnici v daném hodinovém intervalu. Díky těmto počtům si můžu udělat přibližný obrázek toho, jak asi vypadala hustota dopravy.

Tabulky uložené jak v databázi, tak v Excelu obsahovaly informace o počtech průjezdů vozidel za jednotlivé hodinové intervaly a měla vždy stejnou strukturu a vypadala následovně:

Čas	Frýdek-Místek => Ostrava	Ostrava => Frýdek-Místek
00 - 01	0	0
01 - 02	0	0
02 - 03	0	1
03 - 04	0	0
04 - 05	0	0
05 - 06	0	0
06 - 07	4	1
07 - 08	6	1
08 - 09	0	1
09 - 10	1	2
10 - 11	1	3
11 - 12	2	2
12 - 13	1	2
13 -14	3	5
14 -15	3	1
15 - 16	0	0
16 - 17	2	1
17 - 18	1	2
18 - 19	0	0
19 - 20	0	0
20 - 21	0	0
21 - 22	0	0
22 - 23	0	0
23 - 24	0	0

Tabulka 3: Tabulka počtů průjezdů vozidel v hodinových intervalech

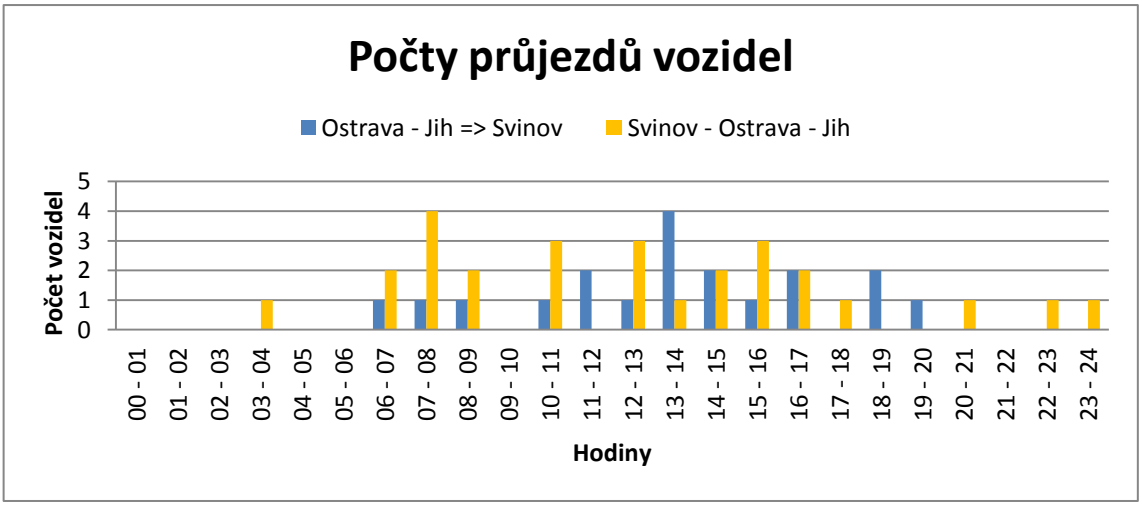
Data uložená v Excelu jsem následně interpretoval jako dvourozměrné sloupcové grafy. Grafy vždy vystihovaly počty vozidel pohybující se v obou směrech vozovky v jednotlivých hodinových intervalech. Pro ukázku jsem vybral jeden z grafů, který mám uložený v Excelu.



Graf 1: Počet průjezdů vozidel ze dne 23.6.2008 na trase Frýdek-Místek, Ostrava

V grafickém výstupu je krásně patrné, kolik vozidel projíždělo na vybrané trase v daném časovém intervalu. Modrou barvou je zde vyobrazena cesta mezi Frýdkem-Místek a Ostravou. A žlutou barvou je zobrazena trasa mezi Ostravou a Frýdkem-Místkem. V grafu si můžeme všimnout, že hustota dopravy ve vybraném dnu není konzistentní, ale je v čase proměnlivá. Charakteristika dopravy vycházela v průběhu pracovních dnů obdobně. Velký rozdíl však nastával o víkendu, kde jezdilo minimu vozidel.

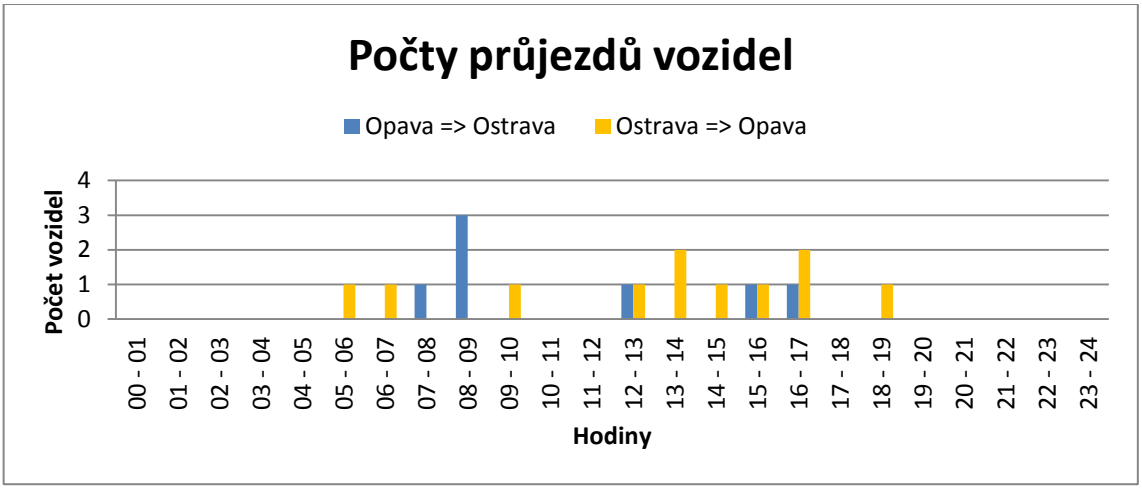
K druhé trase, nad kterou jsem prováděl analýzy, byla trasa mezi Ostravou-Jih a Svinovem. Jednalo se o silnici Rudná, jejíž číslo je 11. Graf, zobrazující počty pohybujících se vozidel nad vybranou vozovkou, vypadal takto:



Graf 2: Počet průjezdů vozidel ze dne 2.6.2008 na trase Ostrava-Jih, Svinov

V grafickém výstupu můžeme opět sledovat množství vozidel pohybujících se v daném úseku silnice. Nadále byly analyzovány oba směry silniční komunikace. Počet vozidel pohybující se po silnici nebyl v celém časovém horizontu konstantní. V ranních a večerních hodinách byly počty pohybujících se vozidel po silnicích minimální a jejich maxima dosahovala ve všední den.

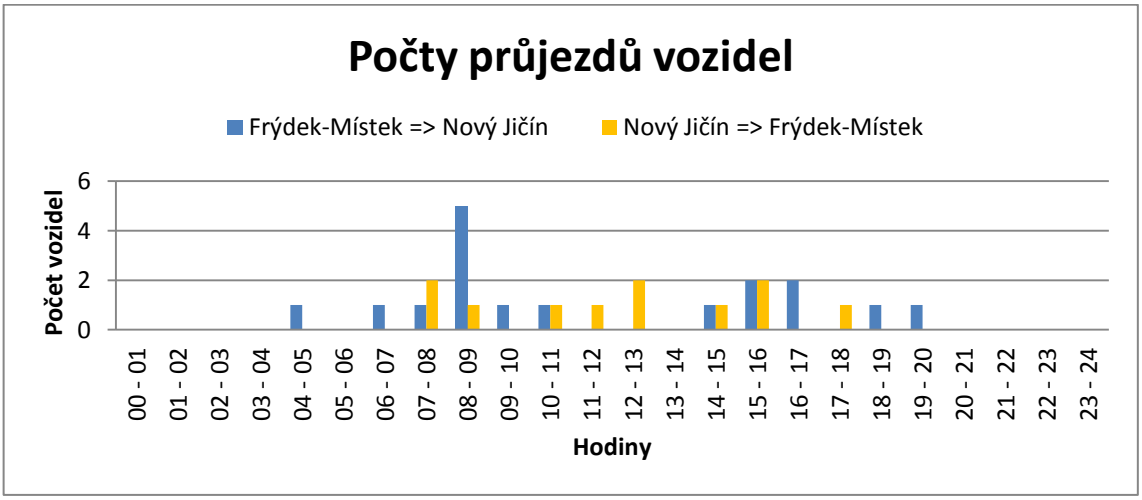
Pro svou třetí trasu jsem zvolil vozovku mezi Ostravou a Opavou. Jednalo se o silnici s názvem Opavská a číslem 11. Dvourozměrný sloupcový graf s počty pohybujících se vozidel nad vybranou silnicí vypadal takto:



Graf 3: Počet průjezdů vozidel ze dne 30.6.2008 na trase Ostrava, Opava

V grafu můžeme sledovat počty pohybujících se vozidel nad vybranou silnicí. Graf mi zachycuje výsledky z obou směrů silnice. Počty pohybujících se vozidel na dané silnici jsou mnohem menší a proto i v grafu zachycení počtu vozidel je skromný.

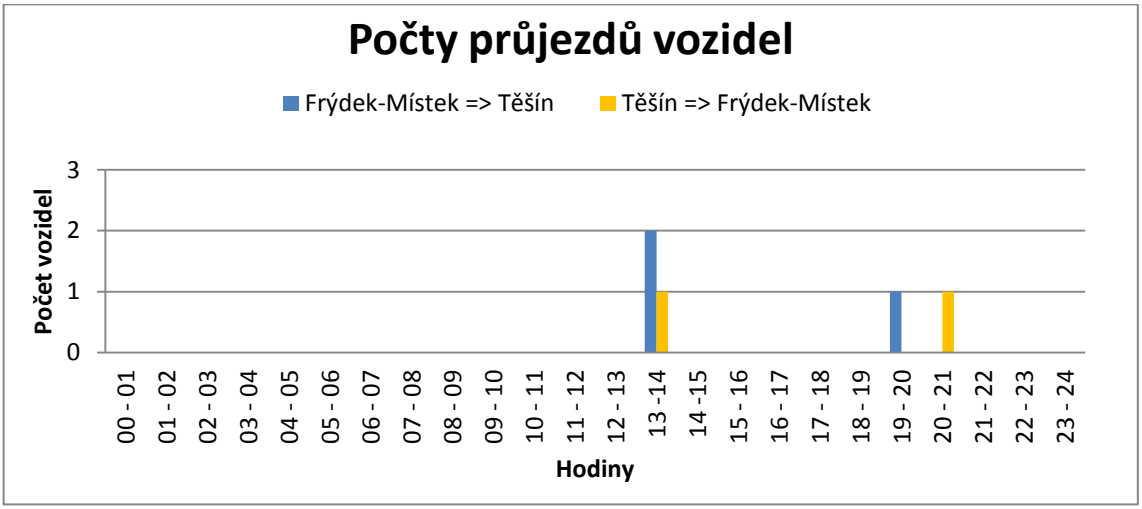
Čtvrtou trasu, kterou jsem si vybral, se nachází mezi Frýdkem-Místkem a Novým Jičínem. Jednalo se silnici s číslem 648. Následný graf s týdenní analýzou vypadal takto:



Graf 4: Počet průjezdů vozidel ze dne 25.6.2008 na trase Frýdek-Místek, Nový Jičín

V grafu můžeme opět pozorovat množství automobilů pohybujících se v určité části silnice. Opět byly analyzovány oba směry silniční komunikace. Množství průjezdů vozidel pohybující po vybrané silnici je poměrně malá, protože se nejedná o silnici, přes kterou by pravděpodobně flotila vozidel často jezdila.

Poslední trasa, kterou jsem si vybral, se nachází mezi Frýdkem-Místkem a Českým Těšínem. Dle vlastního uvážení jsem očekával, že by se zde mohlo vyskytovat velké množství průjezdů vozidel. Výsledek však byl odlišný od mé úvahy. Počty průjezdů byly velice nízké a v některých dnech nebylo takřka co analyzovat. Jednalo se o silnici s číslem 648. Poslední graf této analýzy vypadal takto:



Graf 5: Počet průjezdů vozidel ze dne 3.6.2008 na trase Frýdek-Místek, Český Těšín

Jak už bylo řečeno výše, tak na této trase byl počet průjezdů vozidel opravdu nízký, provézt jakýkoliv závěr není možný. Tento nízký počet vozidel vyskytující se na této trase, byl s velkou pravděpodobností zaviněn tím, že jsem zpracovával pouze vozidla pohybující se v České republice, případně vozidla, která se pohybovala mezi Českou republikou a sousedními státy a naopak. Avšak na této trase se mohly pohybovat vozidla, která začínala svou jízdu v polském Těšíně, případně někde jinde v Polsku a směřovala směrem na Slovensko. Tyto trasy začínající v zahraničí a končící taky v zahraničí nemám v datech zachycené.

4.3.2 Počty průjezdů vozidel v denních intervalech

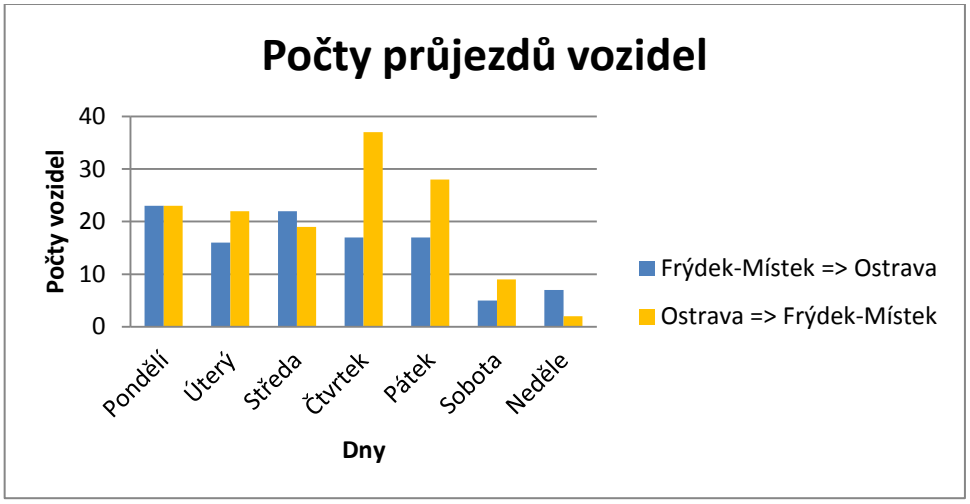
Dané počty průjezdů vozidel v denních intervalech mi bude zachycovat, kolik vozidel se pohybovalo po vybrané silnici v daném denní intervalu. Díky těmto počtům si můžu udělat přibližný obrázek toho, jak asi vypadala hustota dopravy.

Tabulky, obsahující informace o počtech průjezdů vozidel v týdenních intervalech, měla taky pevně danou strukturu. A to tabulky uložené v databázi tak i v Excelu a vypadaly následovně:

Den	Frýdek-Místek => Ostrava	Ostrava => Frýdek-Místek
Pondělí	23	23
Úterý	16	22
Středa	22	19
Čtvrtek	17	37
Pátek	17	28
Sobota	5	9
Neděle	7	2

Tabulka 4: Počty průjezdů vozidel v denních intervalech

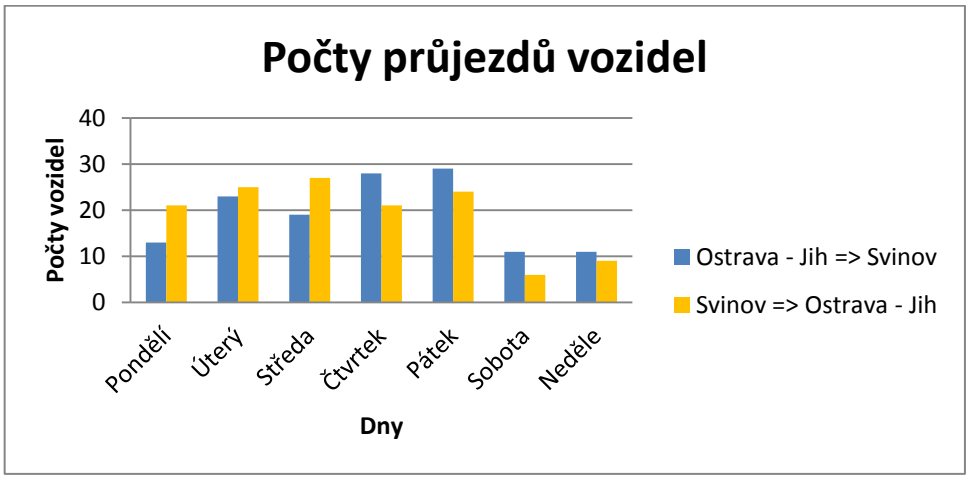
Grafický výsledek týdenní statistické analýzy na trase Frýdek-Místek a Ostrava vizualizován v Excelu byl vytvořen pomocí dvourozměrného sloupcového grafu. Grafy vystihovaly počty vozidel pohybujících po vozovkách v jednotlivých dnech v týdnu. Tyto grafy následně vypadaly takto:



Graf 6: Počet průjezdů vozidel z 26. týdne na trase Frýdek-Místek, Ostrava

Charakter pohybu vozidel byl takový, že pohyb vozidel z počátku týdne byl konstantní. Koncem pracovního týdne se nárůst pohybu flotily vozidel zvětšil, což může mít za následek snahu firem dohánět to, co nestihli v předchozích dnech. Víkend je charakterizován minimálním pohybem vozidel z flotily, čímž můžeme usoudit, že firemní automobily se přes víkend příliš nevyužívají.

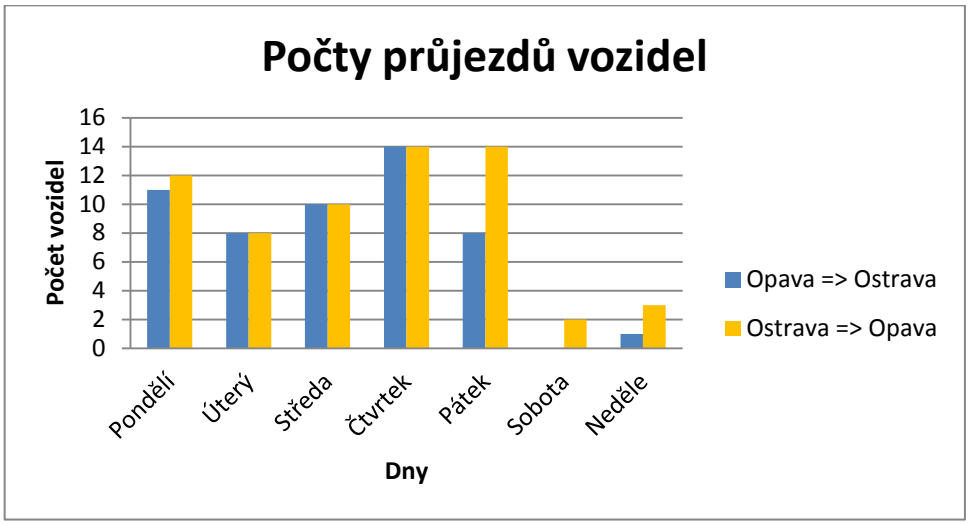
Pro svou druhou týdenní statistickou analýzu jsem si k popisu opět zvolil trasu mezi Ostravou-Jih a Svinovem. Jednalo se o silnici Rudná, jejíž číslo je 11. Grafický výstup následně vypadal takto:



Graf 7: : Počet průjezdů vozidel z 26. týdne na trase Ostrava-Jih, Svinov

V grafu je zřejmé, jak se dopravní zátěž měnila v jednotlivých dnech. Největší skokový rozdíl byl mezi pracovními dny a víkendem. Jak jsem již zmínil v mé práci, přes víkend se firemní vozidla zdaleka tak nevyužívají jak během pracovního týdne.

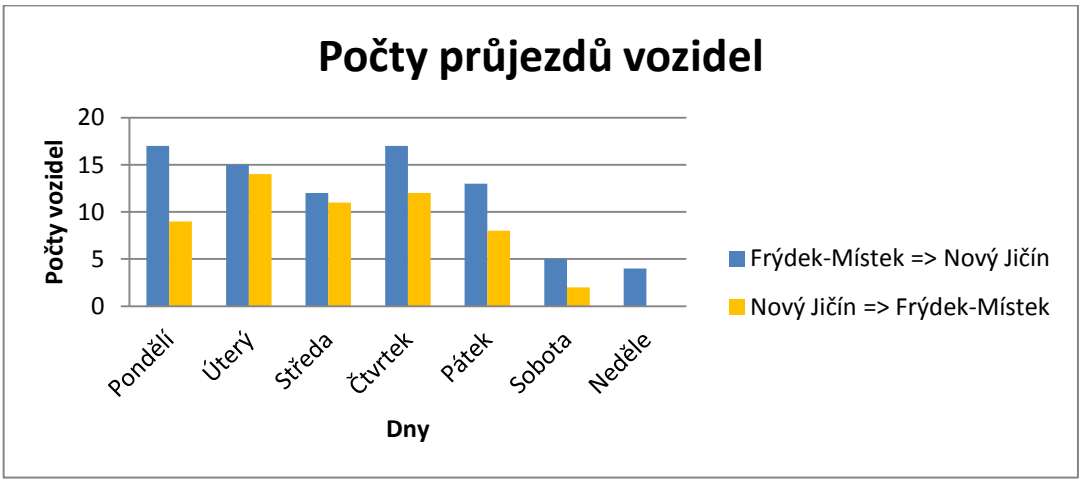
Ke své třetí týdenní statické analýze jsem si zvolil trasu, mezi Ostravou a Opavou. Jednalo se o silnici s názvem Opavská a číslem 11. Dvourozměrný sloupcový graf vypadal následovně:



Graf 8: Graf počtu průjezdů vozidel z 26. týdne, na trase Ostrava, Opava

V týdenní analýze vycházely počty průjezdu v obou směrech shodně, až na pátek, kde došlo k značnému rozdílu. Víkend vychází stále oproti zbytku týdne rozdílně. Ve srovnání s jinými trasami se o víkendu firemní automobily příliš nevyžívají.

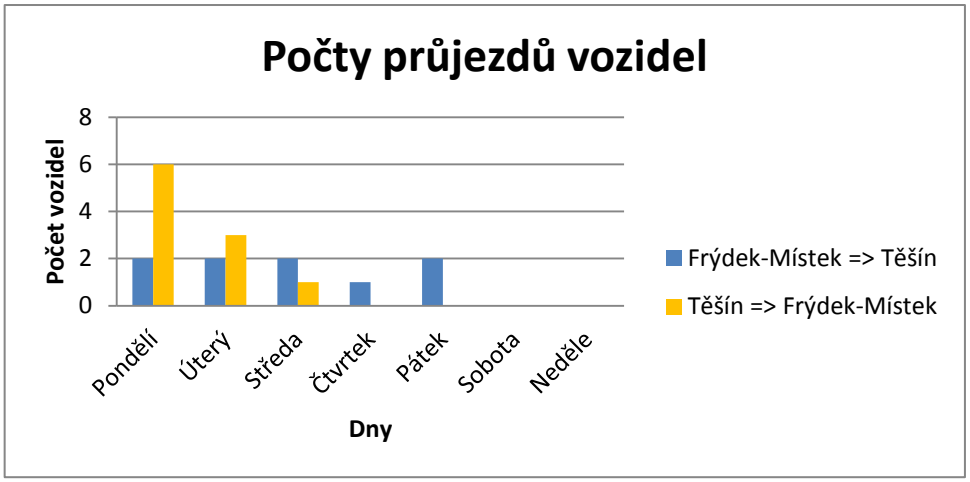
Pro svou čtvrtou týdenní statickou analýzu jsem si zvolil trasu, mezi Frýdkem-Místkem a Novým Jičínem. Jednalo se silnici s číslem 648. Výsledný graf v Excelu vypadal takto:



Graf 9: Počet průjezdů vozidel z 25. týdne na trase Frýdek-Místek, Nový Jičín

V grafu je zřejmé, jak se charakter dopravy měnil v jednotlivých dnech. Největší skokový rozdíl byl mezi pracovními dny a víkendem. Dále si můžeme povšimnout, že každý ze směrů měl vrchol dopravní zátěže v různých dnech.

A poslední graf týdenní statistické analýzy mezi Frýdkem-Místkem a Českým Těšínem vypadal takto:



Graf 10: Počet průjezdů vozidel z 23. týdne na trase Frýdek-Místek, Český Těšín

Počty vozidel v jednotlivých pracovních dnech ve směru na Český Těšín vycházely obdobně. Za to ve směru na Frýdek-Místek byl charakter počtu pohybujících se vozidel klesající.

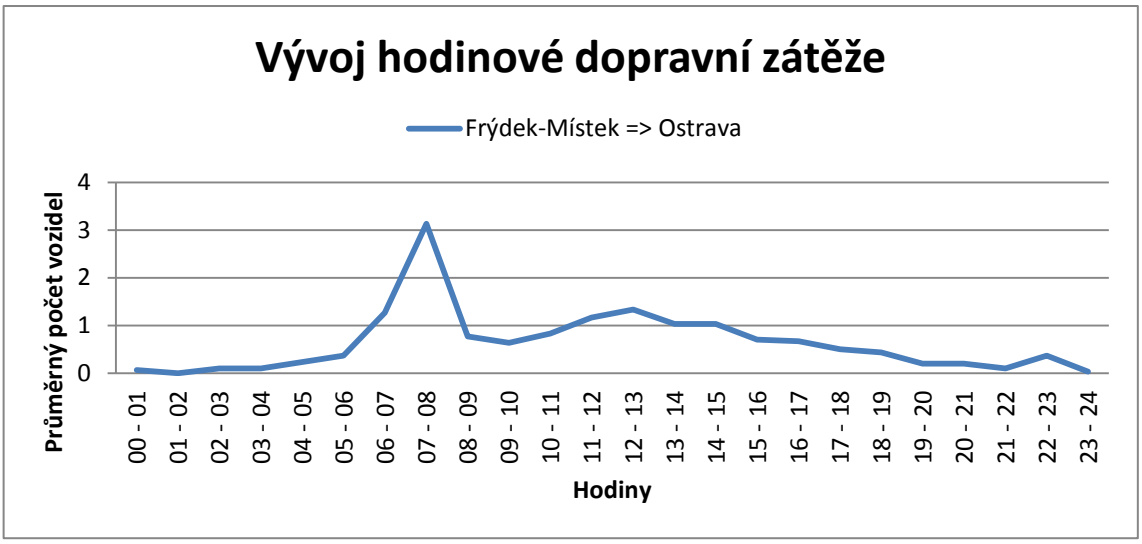
4.3.3 Vývoj hodinové dopravní zátěže

Tabulky, obsahující informace o hodinovém rozložení dopravní zátěže za celý měsíc červen, měly taky pevně danou strukturu. Byly tříděny do jednotlivých hodinových intervalů, v kterých jsem sledoval sumy a průměry hodinového rozložení dopravní zátěže za rok 2008 měsíc červen. Sumy mi ve výsledku charakterizovaly počty vozidel pohybujících se po vybraných silničních komunikacích. Za to průměry mi zachycovaly průměrný počet vozidel pohybujících se po silnicích v měsíci červnu roku 2008. Tyto tabulky byly uložené v databázi tak i v Excelu a vypadaly následovně:

Čas	Průměr	Suma
00 - 01	0,00	0
01 - 02	0,07	2
02 - 03	0,07	2
03 - 04	0,03	1
04 - 05	0,13	4
05 - 06	0,23	7
06 - 07	0,67	20
07 - 08	1,63	49
08 - 09	1,30	39
09 - 10	0,80	24
10 - 11	1,03	31
11 - 12	0,93	28
12 - 13	1,33	40
13 - 14	1,00	30
14 - 15	1,20	36
15 - 16	1,23	37
16 - 17	1,67	50
17 - 18	1,13	34
18 - 19	0,70	21
19 - 20	0,37	11
20 - 21	0,10	3
21 - 22	0,20	6
22 - 23	0,17	5
23 - 24	0,07	2

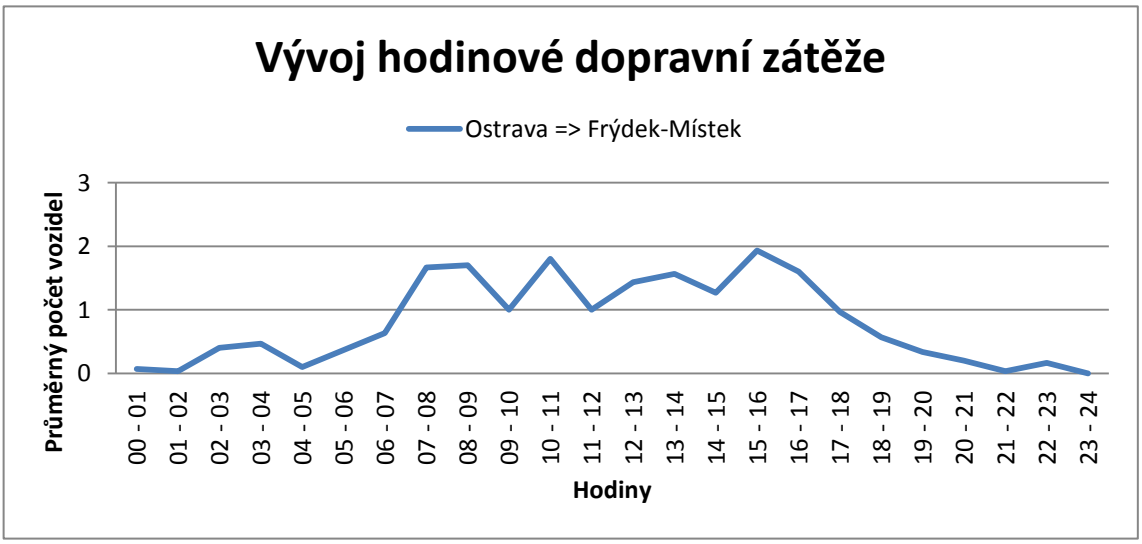
Tabulka 5: Hodnoty dopravní zátěže v hodinových intervalech

Data uložená v Excelu jsem následně interpretoval jako spojnicové grafy. Grafy vždy vystihovaly sumy a průměry hodinové zátěže nad vybranými silnicemi. Díky tomu, že sumy mají přímou závislost na průměry, tak je nebudu ve své práci ani publikovat. Grafy byly vždy tvořeny pro oba směry silničních komunikací. Průměrná měsíční dopravní zátěž byla klasifikována do jednotlivých hodinových intervalů. Výsledné grafy pro trasu mezi Frýdkem-Místek a Ostravou byly interpretovány následovně:



Graf 11: Průměrná dopravní denní zátěž na trase Frýdek-Místek, Ostrava

Graf pro trasu mezi Ostravou a Frýdkem-Místek vypadal takto:

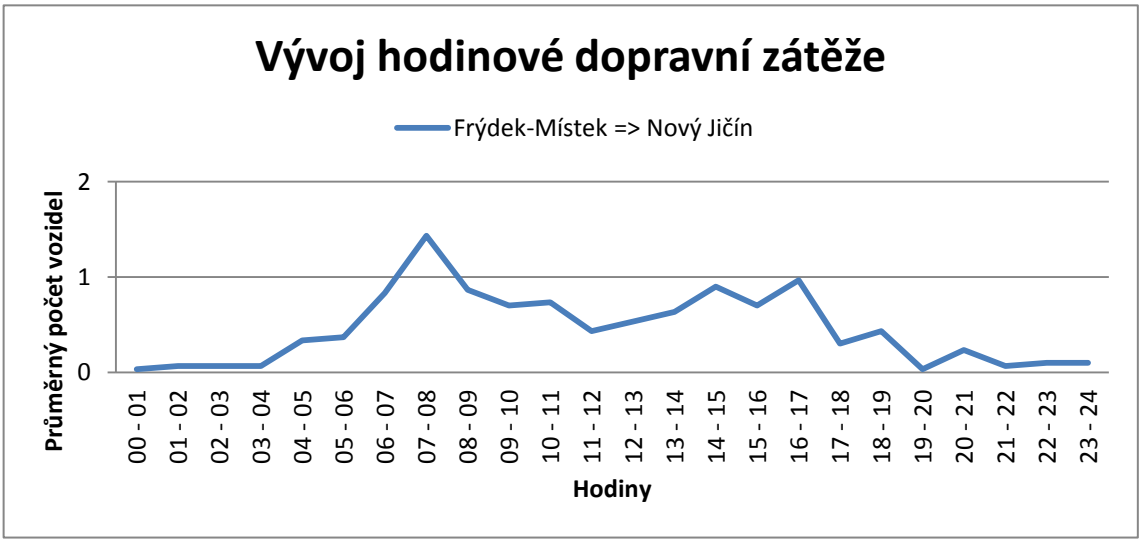


Graf 12: Průměrná dopravní denní zátěž na trase Ostrava, Frýdek-Místek

Z grafů lze krásně pozorovat, kdy bývá průměrná hustota dopravy nejvyšší. Na trase mezi Frýdkem-Místkem a Ostravou bývá průměrná hustota dopravy největší v ranních hodinách v době, když se občané přesouvají za zaměstnáním do Ostravy - mezi 6 až 8 hodinnou ranní. Na opačné trase, mezi Ostravou a Frýdkem-Místkem, hustota dopravy dosahuje maxima v několika vrcholech: v ranních hodinách mezi 7 až 9 hodinou a ve

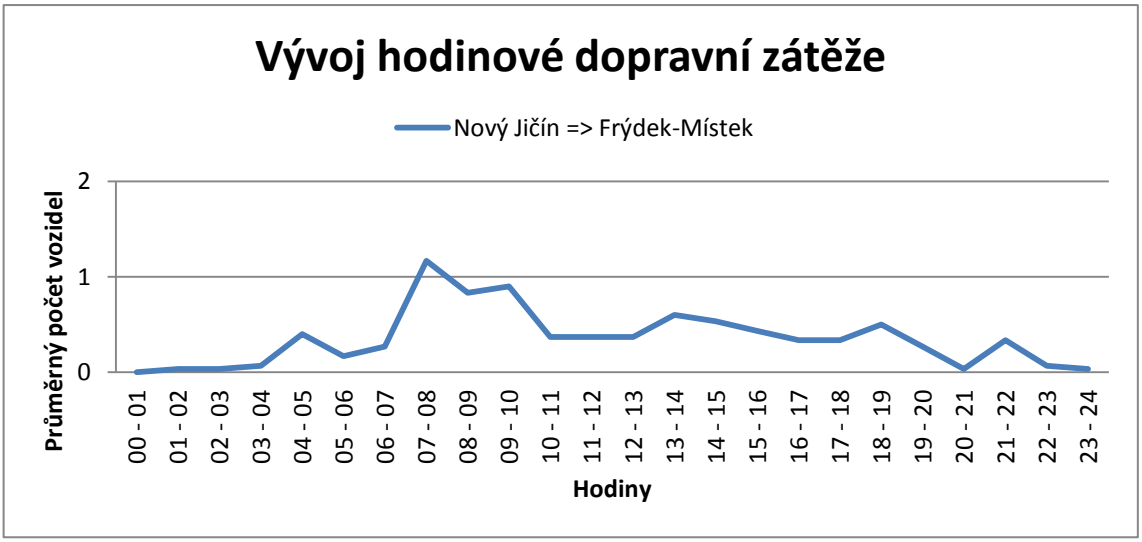
večerních hodinách 15 až 17 hodinou. Tyto vrcholy charakterizují migrací občanů do svých zaměstnání a cestu zpět.

Druhá trasa, na které jsem sledoval průměrnou dopravní zátěž, byla trasa mezi Frýdkem-Místkem a Novým Jičínem. Výsledný graf vypadal následovně:



Graf 13: Průměrná dopravní denní zátěž na trase Frýdek-Místek, Nový Jičín

Pro kompletní představu bylo samozřejmé zachytit i trasu mezi Novým Jičínem a Frýdkem-Místem. Výsledný graf proto vypadal takto:

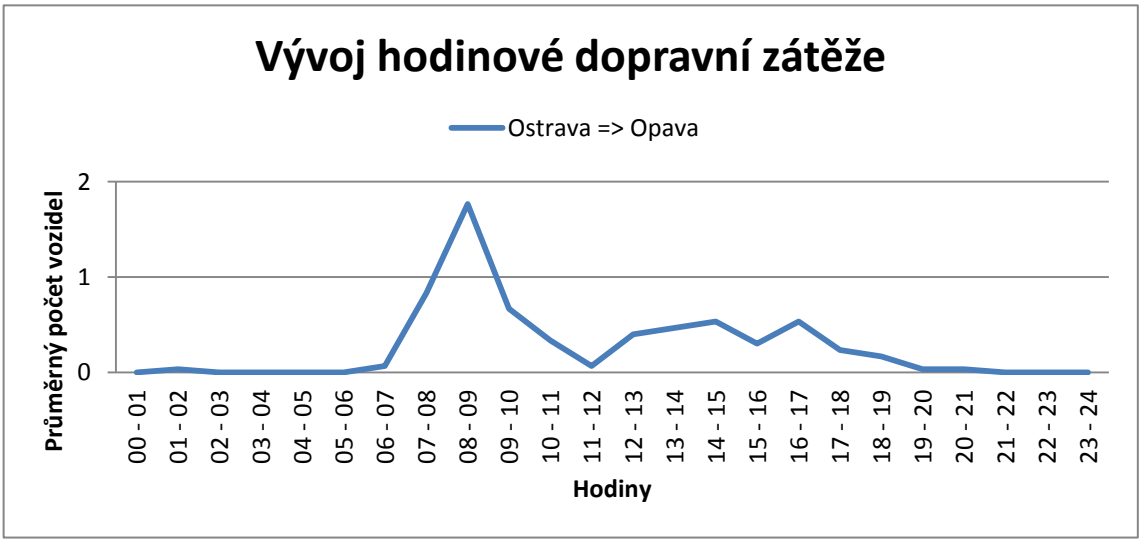


Graf 14: Průměrná dopravní denní zátěž na trase Nový Jičín, Frýdek-Místek

Z grafů lze i nyní krásně pozorovat průměrná maxima, kdy doprava dosahuje nejvyšší průměrné hustoty. Opět dosahuje v obou směrech průměrných maxim v ranních hodinách, kdy se občané přesouvají do svých zaměstnání. Rozdíl ovšem nastává na trase Frýdek-Místek a Nový Jičín, kde druhý větší vrchol nalezneme ve večerních hodinách a je

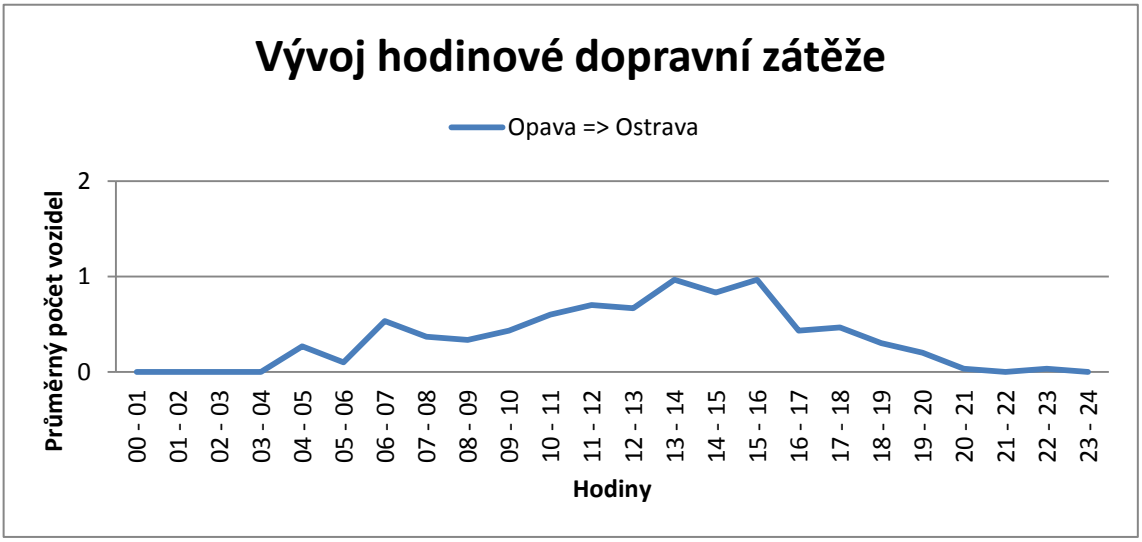
vytvořen migrací občanů ze zaměstnání domů, na rozdíl od trasy Nový Jičín a Frýdek-Místek bylo vytvořeno hned několik menších vrcholů v průběhu zbytku dne.

Pro svou třetí analýzu, kterou jsem zkoumal, jsem použil trasu mezi Ostravou a Opavou. Výsledný graf byl interpretován takto:



Graf 15: Průměrná dopravní denní zátěž na trase Ostrava, Opava

A pro úplnost také graf, který zachycuje trasu mezi Opavou a Ostravou:

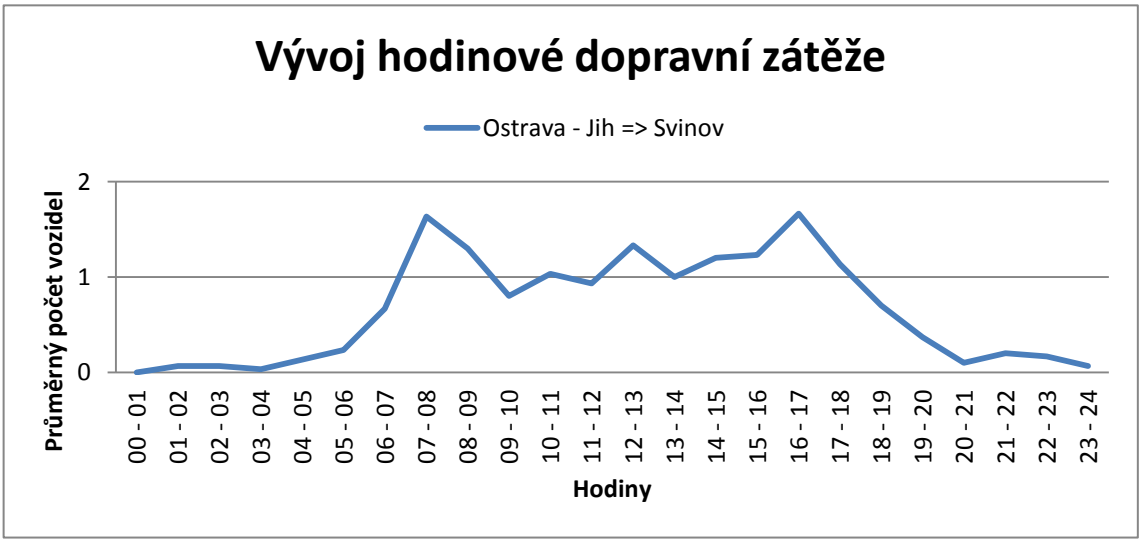


Graf 16: Průměrná dopravní denní zátěž na trase Opava, Ostrava

Na trasách mezi Ostravou a Opavou, následně i opačně, je krásně viditelná migrace obyvatelstva mezi těmito dvěma městy. V prvním případě je na trase mezi Ostravou a Opavou nádherně vidět průměrnou maximální hustotu zatížení dopravy v ranních hodinách, zato v odpoledních hodinách není průměrná hustota dopravy tak vysoká, ale je konstantní. Oproti průměrné hustotě dopravy na trase mezi Opavou a Ostravou je jev

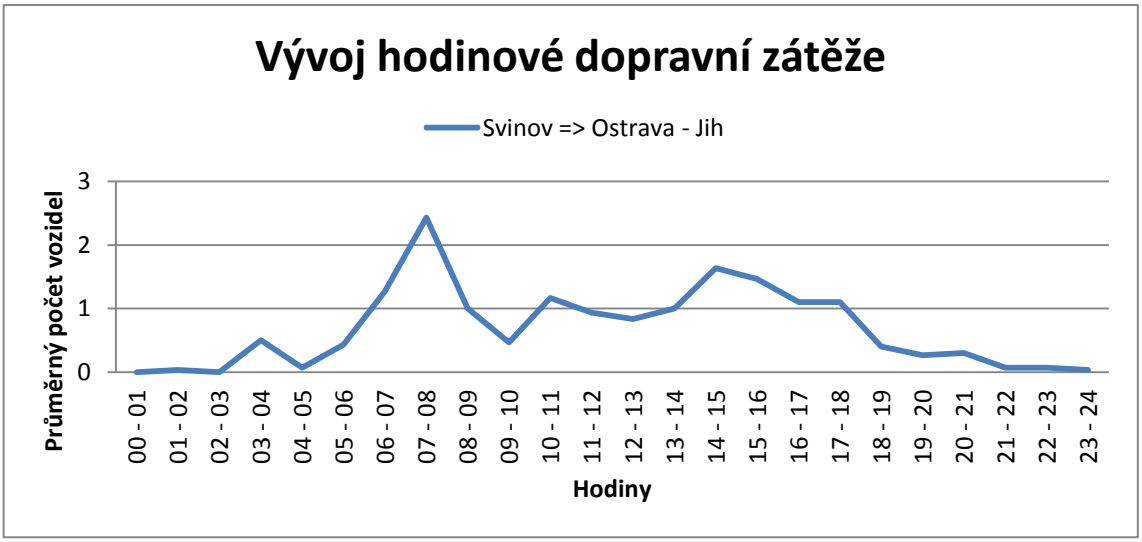
opačný, hlavní vrchol se nachází odpoledne. Tato migrace obyvatelstva je zapříčiněna přesunem obyvatel do zaměstnání a následně přesun z práce domů.

Pro svou další tvorbu analýz průměrné zatížení dopravy jsem vybral trasu mezi Ostravou-Jih a Svinovem. Takto vypadal jeho graf:



Graf 17: Průměrná dopravní denní zátěž na trase Ostava-Jih, Svinov

A pro druhý směr mezi Svinovem a Ostravou-Jih byl vytvořen graf, který vypadal následovně:

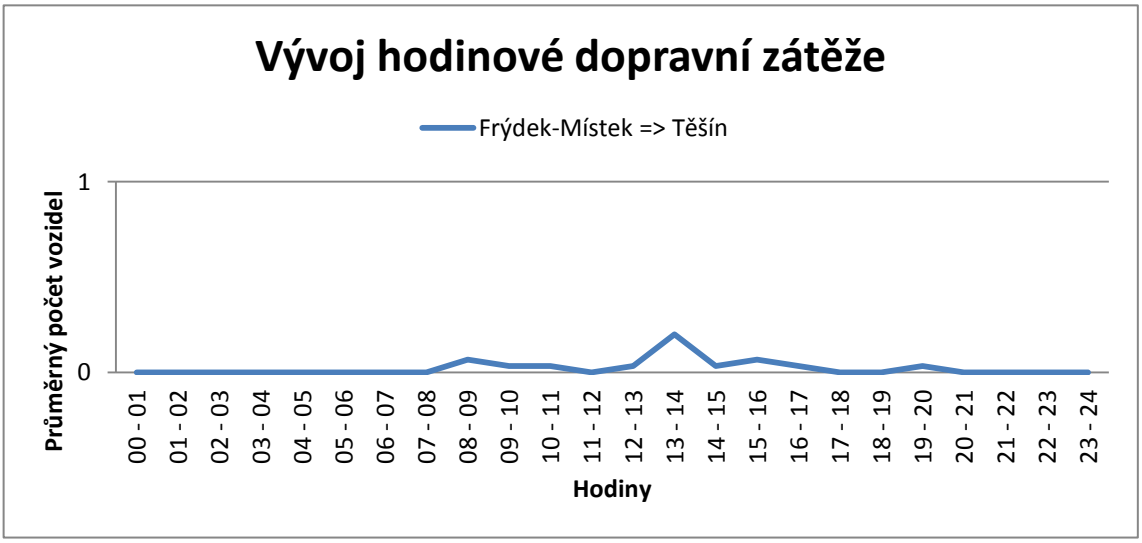


Graf 18: Průměrná dopravní denní zátěž na trase Svinov, Ostrava-Jih

Na těchto grafech si můžeme všimnout vždy dvou vrcholů, které mi interpretují, že v daném časovém intervalu dosahovala průměrná hustota dopravy průměrného maxima. Díky tomu, že se vrcholy nachází v obou případech v ranních a odpoledních hodinách, se mohu domnívat, že vrchol v ranních hodinách je vytvořen migrací obyvatelstva do

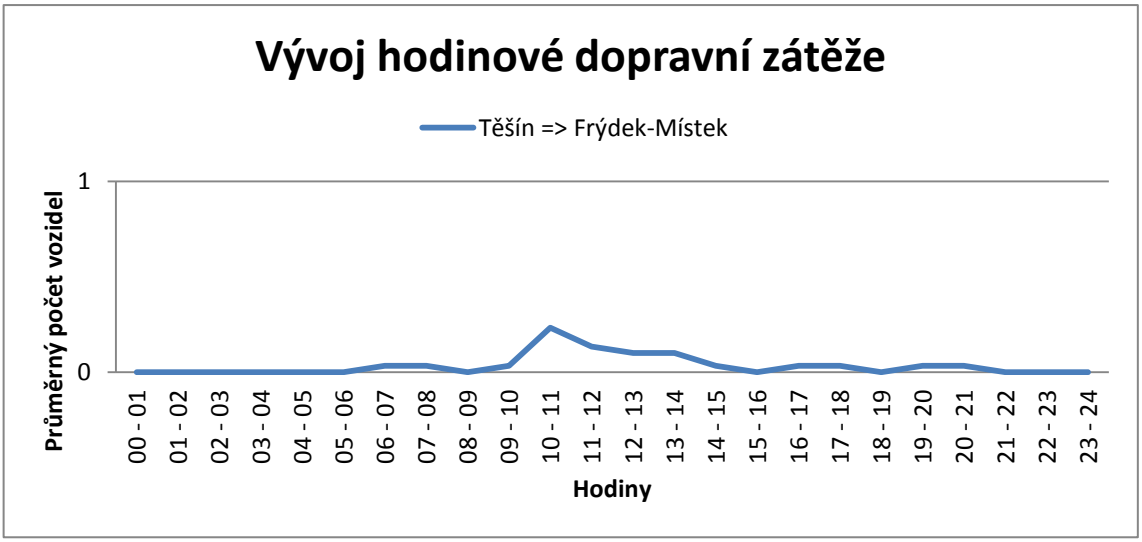
zaměstnání. A vrchol vyskytující se v odpoledních hodinách je vytvořen migrací obyvatelstva ze zaměstnání zpět domů.

Poslední grafy zabývající se průměrnou hodinovou dopravní zátěží zachycují trasu mezi Frýdkem-Místek a Českým Těšínem. Grafy vypadají takto:



Graf 19: Průměrná dopravní denní zátěž na trase Frýdek-Místek, Český Těšín

A grafy zachycující druhý směr trasy mezi Českým Těšínem a Frýdkem-Místkem vypadaly následovně:



Graf 20: Průměrná dopravní denní zátěž na trase Český Těšín, Frýdek-Místek

Grafy vyskytující se na obou trasách vycházely velice obdobně, nejvyšší průměrná hustota dopravy se vyskytovala v denních hodinách, za to menší vrcholy se nacházely kolem něj a to v ranních a odpoledních hodinách. Maximální průměrná hustota dopravy nacházející se v denních hodinách mohla vzniknout migrací dopravců, kteří převážejí zboží mezi Českou republikou a Polskem, případně opačně.

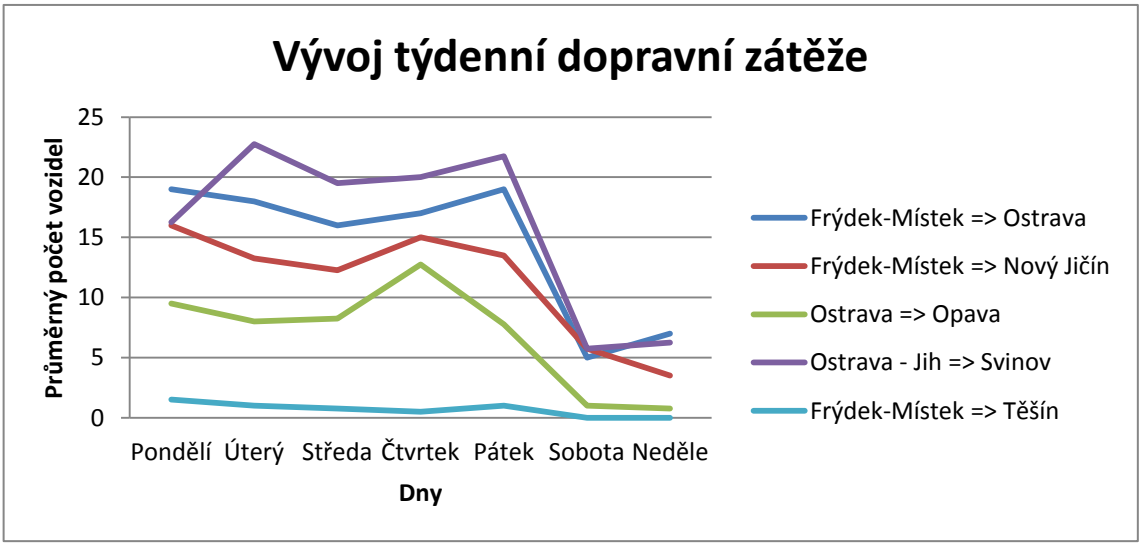
4.3.4 Vývoj denní dopravní zátěže

Tabulky obsahující informace o denní rozložení dopravní zátěže za celý měsíc červen, měly taky pevně danou strukturu. Byly tříděny do jednotlivých denních intervalů, v kterých jsem sledoval sumy a průměry denního rozložení dopravní zátěže za rok 2008 měsíc červen. Sumy mi ve výsledku charakterizovaly počty vozidel pohybujících se po vybraných silničních komunikacích. Za to průměry mi zachycovaly průměrný počet vozidel pohybujících se po silnicích v měsíci červnu roku 2008. Tyto tabulky byly uložené v databázi tak i v Excelu a vypadaly následovně:

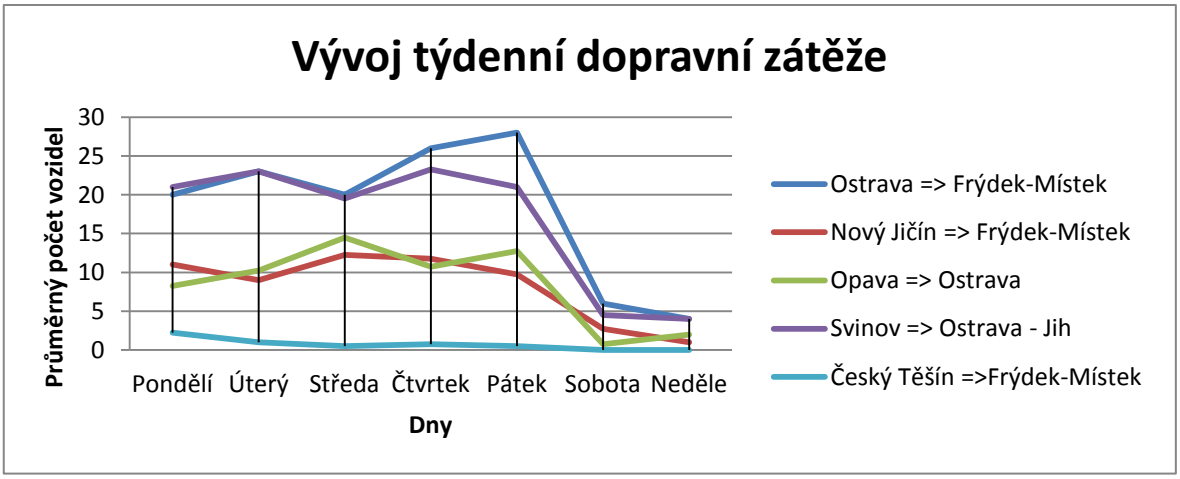
Čas	Průměr	Suma
Pondělí	16,25	84
Úterý	22,75	92
Středa	19,50	78
Čtvrtek	20,00	93
Pátek	21,75	84
Sobota	5,75	18
Neděle	6,25	16

Tabulka 6: Hodnoty dopravní zátěže v denních intervalech

Data uložená v Excelu jsem následně interpretoval jako spojnicové grafy. Grafy vždy vystihovaly sumy a průměry denní zátěže nad vybranými silnicemi. Díky tomu, že sumy mají přímou závislost na průměry, tak je nebudu ve své práci ani publikovat. Grafy byly vždy tvořeny pro oba směry silniční komunikace. Měsíční dopravní zátěž byla klasifikována do jednotlivých denních intervalů. Výsledné grafy pro jednotlivé trasy byly interpretovány následovně:



Graf 21: Průměrná dopravní týdenní zátěž na vybraných trasách



Graf 22: Průměrná dopravní týdenní zátěž na vybraných trasách

Z grafů můžeme vyčíst, jak se vyvíjela dopravní zátěž v jednotlivých dnech. V grafu je krásně viditelné, že v průběhu pracovních dnů je dopravní zatížení relativně konstantní, jediný velký extrém nastává až o víkendu. Tento extrém vznikl dle mé hypotézy tak, že o víkendu se příliš nevyužívají firemní automobily, proto je výskyt vozidel o víkendu tak malý. Kdyby se jednalo o vozidla určena k běžné činnosti, tak bych mohl očekávat, že o víkendu nastanou velké migrace obyvatelstva a to směrem k rekreačním střediskům, k horám nebo k chatovým oblastem.

Následné malé extrémy, nacházející se v pracovních dnech mohou být důsledkem několika faktorů. Jelikož neznám poskytovatele dat a ani typy vozidel, které tato data sbírala, tak se mohu pouze domnívat, jak tyto malé extrémy vnikly.

Jednou z hypotéz je, že se jedná o nějaké dopravce. Tím by se mohlo usuzovat, že v pátek bývá dopravní zátěž největší, protože se snaží dohnat resty ze zbytku týdne.

Další hypotézou je, že na vybraných silnicích v určité dny probíhaly nějaké komplikace, např. plánovaná údržba nebo nastala dopravní kolize. Díky těmto faktorům se mohlo několik řidičů rozhodnout a zvolit jinou silniční komunikaci, která by byla pro ně schůdnější.

5 ZÁVĚR

Cílem práce bylo se seznámit se zpracováním dopravních dat v prostředí prostorové databáze.

Prvními kroky bylo důležité seznámit se s daty a zvolení vhodných aplikací, které budu využívat při svých zpracováních dat. Jedinou mou podmínkou ke zvolení vhodného softwaru bylo, že program musí být nejlépe open-source.

Když jsem zvolil a nainstaloval vhodný software. Bylo potřeba se seznámit s prostředím PostgreSQL/PostGIS a nastudovat manuály, jak zacházet s daty v databázi. Po nastudování základních kroků bylo potřeba započíst samotnou práci s daty. K těmto krokům patřilo vytvoření databáze a následný import dat. Když byla data importována, mohl jsem začít se samotným zpracováním dat.

Jakmile byla data zpracována, bylo potřeba se poohlédnout po datech, která by sloužila jako pomocná data pro následné analýzy. Mezi tato data patřila silniční síť z OpenStreetMap.

Když byla veškerá data připravena, vytvořil jsem finální strukturu pomocí partitioningu. Díky partitioningu jsem byl schopen vytvořit pro mé potřeby snad dokonalou tabulkovou strukturu. Díky ní se mi podařilo zrychlit a zefektivnit celou mou práci.

Poslední a nejdůležitější části bylo vytvoření postupů pro výpočet analýz a následnou interpretací jejich výsledků. Výsledky byly interpretovány dvojím způsobem. A to pomocí výstupních tabulek v databázi, které jsem následně přenesl do MS Excelu, v kterém jsem posléze vytvářel grafické výstupy.

V této práci vidím veliký potenciál, protože jsem dokázal využít open-source software, který je schopen efektivně zpracovat ohromné množství dat a vypočíst potřebné analýzy a tím konkurovat i komerčním produktům. Díky zvolené struktuře jsem také schopen využívat velkou sadu dalších nástrojů, pro publikaci na webu pomocí mapové aplikace, nebo pro výpočet složitějších analýz pomocí různých sofistikovaných programovacích jazyků.

SEZNAM POUŽITÉ LITERATURY

1. About. *Notepad++ Home.* [Online] 2011. [Citace: 27. 4 2014.] http://notepad-plus-plus.org/.

2. CASE Studio 2. *Internet Archive: Digital Library of Free Books, Movies, Music & Wayback Machine.* [Online] 08. 11 2004. [Citace: 27. 4 2014.] https://archive.org/details/tucows_349753_CASE_Studio_2.

3. PostGIS. *PostGIS – FreeGIS portál.* [Online] 12. 11 2013. [Citace: 27. 4 2014.] http://freegis.fsv.cvut.cz/gwiki/PostGIS.

4. PostgreSQL. *PostgreSQL.* [Online] 31. 3 2014. [Citace: 27. 4 2014.] http://postgres.cz/wiki/PostgreSQL.

5. Úvod. *pgAdmin: Nástroje pro administraci a správu PostgreSQL.* [Online] 2013. [Citace: 27. 4 2014.] http://www.pgadmin.org/?lang=cs_CZ.

6. QGIS. *Wikipedie, otevřená encyklopedie.* [Online] 19. 4 2014. [Citace: 27. 4 2014.] http://cs.wikipedia.org/wiki/QGIS.

7. Welcome. *GeoServer.* [Online] 16. 4 2014. [Citace: 27. 4 2014.] http://geoserver.org/display/GEOS/Welcome.

8. Numeric Types. *PostgreSQL.* [Online] [Citace: 27. 4 2014.] http://www.postgresql.org/docs/9.1/static/datatype-numeric.html.

9. Chapter 4. Using PostGIS: Data Management and Queries. *PostGIS.* [Online] [Citace: 27. 4 2014.] http://postgis.refractions.net/documentation/manual-1.5/ch04.html.

10. SED. *Wikipedie, otevřená encyklopedie.* [Online] 20. 1 2014. [Citace: 27. 4 2014.] http://cs.wikipedia.org/wiki/Sed.

11. PostGIS 2.1.3dev Manual. *PostGIS.* [Online] [Citace: 29. 4 2014.] http://postgis.net/docs/manual-2.1/.

12. PostgreSQL 9.1.13 Documentation. *PostgreSQL.* [Online] [Citace: 29. 4 2014.] http://www.postgresql.org/docs/9.1/interactive/index.html.

13. REAL TIME ROME. *MIT SENSEable City Lab.* [Online] [Citace: 29. 4 2014.] http://senseable.mit.edu/realtimerome/.

14. ITO Map. *ITO World.* [Online] 2014. [Citace: 29. 4 2014.] http://www.itoworld.com/.

15. Visualising the Pulse of the City. *InfoVis.net en Español.* [Online] 6. 11 2007. [Citace: 29. 4 2014.] http://www.infovis.net/printMag.php?lang=2&num=190.

SEZNAM OBRAZKŮ

Obrázek 1: Screenshot surových dat jízd....4
Obrázek 2: Screenshot surových dat poloh....4
Obrázek 3: Screenshot z QGISu náhodně zvolené jízda propojená s polohami....5
Obrázek 4: Screenshot z QGISu, zobrazující mračno bodů tvořící tabulku jízdy....15
Obrázek 5: Screenshot z QGISu zobrazující vizualizaci metody BOUNDARY 15
Obrázek 6: Screenshot z QGISu zobrazující mračno bodů zpracované metodami BOUNDARY a ST_Intersect nad geometrii s počátečními body 16
Obrázek 7: Screenshot z QGISu zobrazující mračno koncových bodů vázané na počáteční body 17
Obrázek 8: Screenshot z QGISu zobrazující mračno bodů zpracované metodami BOUNDARY a ST_Intersect nad geometrii s koncovými body....17
Obrázek 9: Screenshot z QGISu zobrazující mračno počátečních bodů vázané na koncové body 18
Obrázek 10:Screenshot z QGISu zobrazující mračno bodů propojením tabulek 18
Obrázek 11: Screenshot z databáze, zobrazující datový výstup 19
Obrázek 12: Screenshot z QGISu, zobrazující mračno bodů na nežádoucích místech 20
Obrázek 13: Screenshot z QGISu, zobrazující vektorový buffer 21
Obrázek 14: Screenshot z QGISu, zobrazující body nad jednotlivými směry silnice 23
Obrázek 15: Screenshot datového modelu....24

SEZNAM GRAFŮ

Graf 1: Počet průjezdů vozidel ze dne 23.6.2008 na trase Frýdek-Místek, Ostrava 30
Graf 2: Počet průjezdů vozidel ze dne 2.6.2008 na trase Ostrava-Jih, Svinov 30
Graf 3: Počet průjezdů vozidel ze dne 30.6.2008 na trase Ostrava, Opava 31
Graf 4: Počet průjezdů vozidel ze dne 25.6.2008 na trase Frýdek-Místek, Nový Jičín 31
Graf 5: Počet průjezdů vozidel ze dne 3.6.2008 na trase Frýdek-Místek, Český Těšín 32
Graf 6: Počet průjezdů vozidel z 26. týdne na trase Frýdek-Místek, Ostrava 34
Graf 7: : Počet průjezdů vozidel z 26. týdne na trase Ostrava-Jih, Svinov 34
Graf 8: Graf počtu průjezdů vozidel z 26. týdne, na trase Ostrava, Opava 35
Graf 9: Počet průjezdů vozidel z 25. týdne na trase Frýdek-Místek, Nový Jičín 35
Graf 10: Počet průjezdů vozidel z 23. týdne na trase Frýdek-Místek, Český Těšín.....36
Graf 11: Průměrná dopravní denní zátěž na trase Frýdek-Místek, Ostrava.....38
Graf 12: Průměrná dopravní denní zátěž na trase Ostrava, Frýdek-Místek.....38
Graf 13: Průměrná dopravní denní zátěž na trase Frýdek-Místek, Nový Jičín.....39
Graf 14: Průměrná dopravní denní zátěž na trase Nový Jičín, Frýdek-Místek.....39
Graf 15: Průměrná dopravní denní zátěž na trase Ostrava, Opava 40
Graf 16: Průměrná dopravní denní zátěž na trase Opava, Ostrava 40
Graf 17: Průměrná dopravní denní zátěž na trase Ostava-Jih, Svinov 41
Graf 18: Průměrná dopravní denní zátěž na trase Svinov, Ostrava-Jih 41
Graf 19: Průměrná dopravní denní zátěž na trase Frýdek-Místek, Český Těšín 42
Graf 20: Průměrná dopravní denní zátěž na trase Český Těšín, Frýdek-Místek 42
Graf 21: Průměrná dopravní týdenní zátěž na vybraných trasách 43
Graf 22: Průměrná dopravní týdenní zátěž na vybraných trasách 44

SEZNAM TABULEK

Tabulka 1: Požívaných datových typů [9] 7
Tabulka 2: Používaných datových typů pro nastavení geometrie [10] 7
Tabulka 3: Tabulka počtů průjezdů vozidel v hodinových intervalech 29
Tabulka 4: Počty průjezdů vozidel v denních intervalech 33
Tabulka 5: Hodnoty dopravní zátěže v hodinových intervalech 37
Tabulka 6: Hodnoty dopravní zátěže v denních intervalech 43

PŘÍLOHY

Přílohy jsou na přiloženém CD.