Doplněk Vyhledávání RÚIAN

pro ArcGIS for Desktop

Dokumentace

Říjen 2015

ARCDATA PRAHA
esri Official Distributor

Obsah

1 Úvod

Tento dokument popisuje instalaci a použití doplňku Vyhledávání RÚIAN pro ArcGIS for Desktop. Doplněk poskytuje funkčnost fulltextového vyhledávání nad daty RÚIAN v relační enterprise geodatabázi.

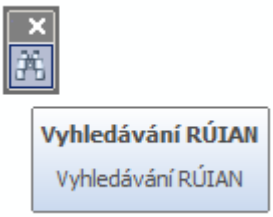
2 Doplněk Vyhledávání RÚIAN pro ArcGIS Desktop

Doplněk Vyhledávání RÚIAN představuje uživatelské rozhraní umožňující vyhledávání v datech RÚIAN v aplikaci ArcMap (licencní úroveň Basic, Standard, Advanced). Je vyžadováno, aby data RÚIAN byla uložena v enterprise geodatabázi Oracle, Microsoft SQL Server nebo PostgreSQL a odpovídala datovému modelu, který byl vytvořen nástrojem VFR Import.

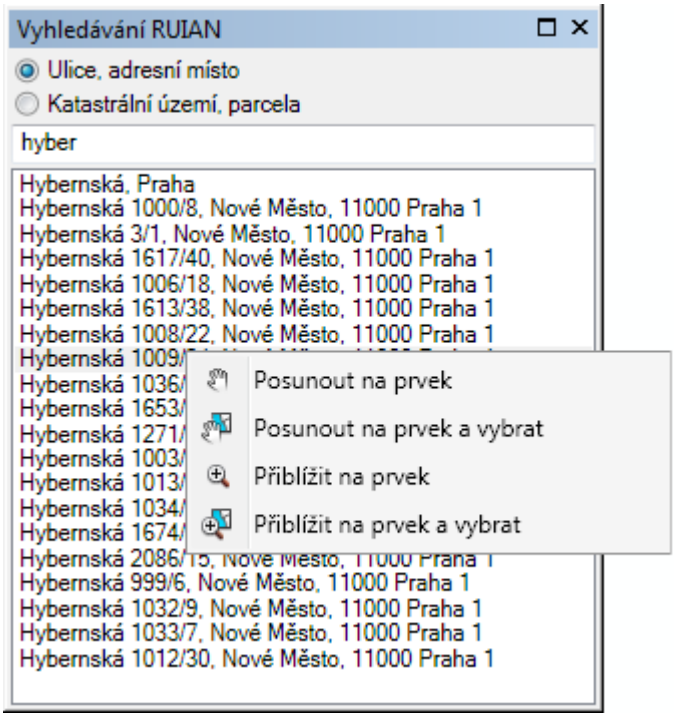
2.1 Instalace a spuštění

Doplněk Vyhledávání RÚIAN je distribuován jako doplněk pro aplikaci ArcMap ve formátu *.esriAddIn. Spuštěním tohoto souboru dojde k instalaci doplňku. Pro spuštění doplňku je třeba si v apliakci ArcMap zobrazit lištu nástrojů (*Toolbar*) Vyhledávání RÚIAN přes *Přizpůsobit > Lišty nástrojů > Vyhledávání RÚIAN (Customize > Toolbars > Vyhledávání RÚIAN)*. Tato nástrojová lišta obsahuje tlačítko, kterým lze spustit samotný doplněk.

Obr. 1



Obr. 2



2.2 Požadavky a konfigurace

Před používáním doplňku je potřeba doplněk správně nakonfigurovat a dále je také třeba nakonfigurovat databázi, nad níž bude vyhledávání probíhat. Konfigurace doplňku probíhá prostřednictvím textového JSON souboru, který je zapouzdřen uvnitř instalačního *.esriAddIn souboru. Pro konfiguraci je nutné provést následující:

1. Přejmenovat příponu *.esriAddIn na *.zip.
2. Rozbalit soubor VyhledavaniRUIANAddin.zip.
3. Přemístit se do adresáře .\Install\Config.
4. Otevřít soubor config.json v libovolném textovém editoru, provést úpravy a uložit změny.
5. Všechen rozbalený obsah znovu zabalit jako zip archiv.
6. Příponu *.zip přejmenovat zpět na *.esriAddIn

2.2.1 Popis parametrů konfiguračního souboru

Konfigurační soubor slouží především pro nastavení dat, nad kterými se má vyhledávat. Vyhledávat lze současně nad více třídami prvků, u každé třídy prvků však právě na jedním polem, nad kterým je vytvořen fulltextový index.

V konfiguračním souboru lze zadat tzv. definice fulltextového vyhledávání (`ftsDefinitions`). Touto definicí se rozumí skupina tříd prvků, nad kterými bude společně probíhat vyhledávání. Mezi jednotlivými definicemi se poté přepíná v uživatelském rozhraní doplňku a vyhledávání poté probíhá v aktuálně přepnuté definici. V konfiguračním souboru níže jsou uvedeny dvě takovéto definice, první slouží pro společné vyhledávání adresních míst a ulic, a druhá definice slouží pro společné vyhledávání parcel a katastrálních území.

U každé definice je potřeba zadat název (`ftsDefName`), cestu k *.sde souboru pro připojení do dané geodatabáze (`sdeConnectFile`), která obsahuje třídy prvků, nad kterými bude vyhledávání probíhat. Dále lze nastavit maximální počet výsledků vyhledávání (`maxLocations`), a minimální počet znaků, které musí být zadány, než se spustí vyhledávání (`minChars`). Volitelně lze pro každou definici vyhledávání nastavit prostorový filtr, kterým se má omezit vyhledávání (např. chceme vyhledávat jen v určitém okresu). Takovýto prostorový filtr se nastavuje pomocí parametru `spatialFilter`, který se skládá z názvu třídy prvků (`datasetName`) a hodnot OBJECTID těch prvků (`objectIds`), jejichž geometrie se má použít pro prostorový filtr.

Samotné třídy prvků, nad kterými se má vyhledávat, jsou definovány v rámci parametru `ftsDatasets`. U každé třídy prvků je potřeba povinně zadat její plný databázový název (`datasetName`), název pole, nad kterým se má vyhledávat (`fieldNameFts`) a požadavky na znaky, po jejichž zadání se má spustit vyhledávání (`searchStringReqs`). Pro parametr `searchStringReqs` jsou povolennými hodnotami `alpha` (vstupní řetězec musí obsahovat písmeno) anebo `alphanumeric` (vstupní řetězcec musí obsahovat písmeno i číslici). Pomocí parametru `searchStringReqs` lze optimalizovat rychlost vyhledávání, např. vyhledávání parcel se spustí až při zadání písmen i číslic.

Volitelnými parametry u každého `ftsDataset` jsou parametry `fieldNameUrlId`, `goToUrl a goToUrlLabel`. Tyto parametry slouží pro nastavení URL odkazu v kontextovém menu výsledků vyhledávání. V příkladu JSON níže jsou tyto parametry použity pro konfiguraci URL odkazu do *Nahlížení do KN*. Hodnota `fieldNameUrlId` akutálně vybraného výsledku se vloží na místo znaku {0} v URL adrese parametru `goToUrl`. Hodnota parametru

`goToUrlLabel` je použita v URL adrese parametru `goToUrl`. Hodnota parametru `goToUrlLabel` je použita jako popisek v kontextovém menu.

Dále je třeba u každé definice vyhledávání (`ftsDefinition`) uvést informace o tabulce (`ftsConfigTable`), která obsahuje předpisy fulltextových dotazů pro každé pole, nad kterým se má vyhledávat. U této tabulky se předpokládá, že se stejně jako výše zmíněné třídy prvků nachází ve stejné databázi a stejném schématu. U tabulky je potřeba uvést její název (`tableName`), název pole, které obsahuje název třídy prvků, nad kterou se má vyhledávat (`fieldNameFtsTable`), název pole, které obsahuje název pole dané třídy prvků, nad kterým se má vyhledávat (`fieldNameFts`), a také název pole, které obsahuje předpis fulltextového dotazu (`fieldNameFtsConfig`).

Ukázka JSON konfiguračního souboru:

```
{
	"ftsDefinitions":
	[
		{
			"ftsDefName": "Adresní místo",
			"sdeConnectFile": "c:\\ruian\\RUIANDB.sde",
			"maxLocations": "20",
			"minChars": "3",
			"spatialFilter":
			{
				"datasetName": "Okres",
				"objectIds" : "1,4,10,11"
			},
			"ftsDatasets":
			[
				{
					"datasetName": "dbname.dbuser.Ulice",
					"fieldNameFts": "ulice",
					"searchStringReqs": "alpha"

				},
				{
					"datasetName": "dbname.dbuser.AdresniMisto",
					"fieldNameFts": "Adresa",
					"searchStringReqs": "alpha"
				}
			],
			"ftsConfigTable":
			{
				"tableName": " dbname.dbuser.fts_tables",
				"fieldNameFtsTable": "table_name",
				"fieldNameFts": "field_name",
```

```
                        "fieldNameFtsConfig": "config"
                }
        },
        {
                "ftsDefName": "Parcela",
                "sdeConnectFile": "c:\\ruian\\RUIANDB.sde",
                "maxLocations": "20",
                "minChars": "3",
                "ftsDatasets":
                [
                        {
                                "datasetName": "dbname.dbuser.KatastralniUzemi",
                                "fieldNameFts": "nazev",
                                "searchStringReqs": "alpha"
                        },
                        {
                                "datasetName": " dbname.dbuser.ParcelaDefinicniBod",
                                "fieldNameFts": "KatastralniUzemiCisloParcely",
                                "searchStringReqs": "alphanumeric",
                                "fieldNameUrlId": "Id",
        "goToUrl": "http://nahlizenidokn.cuzk.cz/ZobrazObjekt.aspx?typ=parcela&id={0}",
                                "goToUrlLabel": "Nahlížení do KN"
                        }
                ],
                "ftsConfigTable":
                {
                        "tableName": " dbname.dbuser.fts_tables",
                        "fieldNameFtsTable": "table_name",
                        "fieldNameFts": "field_name",
                        "fieldNameFtsConfig": "config"
                }
        }
    ]
}
```


Vedle výše zmíněné konfigurace doplňku je třeba určitá konfigurace také na straně geodatabáze. Konkrétně jde o vytvoření fulltextových indexů nad danými atributy (povinné) a synonym (volitelné). Přesný postup konfigurace geodatabáze je obsažen v kapitole 3.

2.3 Funkcionalita a ovládání

Doplněk Vyhledávání RÚIAN umožňuje vyhledávat nad jakouli třídou prvků a právě nad jedním jejím polem, které má nastavený fulltextový index. Uživatelské rozhraní poskytuje textové pole, které slouží pro zadání vstupního řetězce, na základě kterého poté vyhledávání probíhá. Po zadání textového řetězce se automaticky spustí vyhledávání a výsledky jsou zobrazeny v části

okna pod vstupním textovým polem. Do části výsledků se lze přemístit pomocí myši anebo klávesnice (dvakrát šipka dolů anebo tabulátor).

Nad vybraným výsledkem lze pomocí klávesy *Enter* anebo dvojlikem myši vyvolat přiblížení v mapě. Nad vybraným výsledkem lze také vyvolat kontextové menu, které obsahuje funkce pro posun a zoom, a nebo posun a zoom s výběrem výsledku. Výběr funguje jen v případě, že je MXD přítomna vrstva, která má jako zdroj odpovídající třídu prvků, z které pochází výsledek.

2.4 Odinstalace doplňku

Doplněk Vyhledávání RÚIAN lze odinstalovat přímo v aplikaci ArcMap přes *Správce doplňků* (*Add-In Manager*) v nabídce *Přizpůsobit* (*Customize*).

3 Konfigurace fulltextových indexů nad daty RÚIAN

Následující kapitola popisuje ukázkovou konfiguraci fulltextových indexů pro účely lokalizace podle adresy, ulice, katastrálního území nebo parcely. Ukázky jsou uvedeny pro databázové platformy Oracle, SQL Server a PostgreSQL. Využití dat RÚIAN není podmínkou, analogickým způsobem lze využít jakákoliv jiná prostorová data.

3.1 Konfigurace fulltextových indexů v databázi Oracle

3.1.1 Požadavky/předpoklady

1) Skript vyžaduje úpravu práv pod SYS uživatelem pro spouštění CTX_DDL knihovny a přiřazení role CTXAPP uživateli s indexovanými daty RUIAN (viz 1. odstavec níže).
2) Další skripty předpokládají spuštění pod uživatelem vlastnícím RUIAN tabulky.
3) Schéma tabulek RUIAN odpovídá stavu, tak jak je importoval nástroj VFR Import (např. tabulka ADRESNIMISTO má níže indexovaný sloupec adresa)
4) Testováno pro Oracle 11Gr2. (není vyloučena nutnost změn v novějších verzích Oracle).
5) Po provedení úprav je doporučeno vyhledávání v podrobit důkladnému testování.

3.1.2 Úprava práv pro Oracle Text pod SYS uživatelem

V následujících grantech nahraďte RUIANUSER jménem uživatele vlastnícího RUIAN data.

Pro přípravu parametrů indexace je nezbytné přidělit práva pro spuštění funkcí databázové knihovny CTX_DDL.

```
GRANT EXECUTE ON CTXSYS.CTX_DDL to RUIANUSER;
```

Pro spuštění Oracle utility ctxload (nahrávající seznam definovaných synonym) je potřeba uživateli přiřadit roli CTXAPP.

```
GRANT CTXAPP TO RUIANUSER;
```

3.1.3 Změna typu sloupce adresa tabulky ADRESNIMISTO před indexací

V případě že tabulka ADRESNIMISTO obsahuje sloupec adresa jako typ nvarchar2, je potřeba tento sloupec převést na typ varchar2 (fulltextová indexace nepodporuje v geodatabázi defaultně definovaný textový typ nvarchar2). Datové typy sloupců tabulky ADRESNIMISTO lze zjistit následujícím příkazem.

```
describe RUIANUSER.ADRESNIMISTO
```

Změna na sloupec na typ varchar2 se jednoduše provede přidáním sloupce typu varchar2, nahráním dat do tohoto přidaného sloupce z původního sloupce adresa a přejmenováním obou zmíněných sloupců:

```
ALTER TABLE RUIANUSER.ADRESNIMISTO ADD ADRESA_NEW VARCHAR2(150 CHAR);
UPDATE RUIANUSER.ADRESNIMISTO SET ADRESA_NEW=ADRESA;
ALTER TABLE RUIANUSER.ADRESNIMISTO RENAME COLUMN ADRESA TO ADRESA_OLD;
ALTER TABLE RUIANUSER.ADRESNIMISTO RENAME COLUMN ADRESA_NEW TO ADRESA;
```

V případě že došlo k úspěšnému přidání sloupce, jeho naplnění hodnotami a přejmenování (doporučeno předem zkontrolovat) je možné původní sloupec přejmenovaný na ADRESA_OLD odstranit:

```
ALTER TABLE RUIANUSER.ADRESNIMISTO DROP COLUMN ADRESA_OLD;
```

3.1.4 Fulltextová indexace tabulky ADRESNIMISTO

Pro akcent nesensitivní (base_letter=yes) fulltextové vyhledávání je potřeba tento požadavek zahrnout do uživatelem definovaných fulltextových preferencí, použitých poté při samotné indexaci.

```
begin
ctxsys.ctx_ddl.create_preference ('cust_lexer','BASIC_LEXER');
ctxsys.ctx_ddl.set_attribute ('cust_lexer','base_letter','YES');
end;
```

Následně je možné tabulku ADRESNIMISTO indexovat s použitím, definovaných preferencí a prázdného stop-listu (seznamu slov vyřazených z indexace). Prázdný stop list zaručí, že do indexu budou zahrnuty i ty části geografických názvů, které by byly jinak vyloučeny (jako např. předložky „u“, „nad“).

```
CREATE INDEX RUIANUSER.ADRESNIMISTO_FTIDX ON RUIANUSER.ADRESNIMISTO (ADRESA)
INDEXTYPE IS CTXSYS.CONTEXT  PARAMETERS ('LEXER cust_lexer STOPLIST
CTXSYS.EMPTY_STOPLIST SYNC ( ON COMMIT)');
```

3.1.5 Definice slovníku adresních synonym

Pro správné vyhledávání v adresách běžně používaných zkratek a synonym, např. „kpt.“ ve smyslu kapitána, „bří“ ve smyslu „bratří“ (a obráceně), je potřeba definovat slovník synonym tzv. thesaurus. Oracle umožňuje vytvořit slovník synonym a tento do fulltextového vyhledávání zahrnout. Nejprve je nutné vytvořit slovníkový soubor, kde se k danému slovu definují všechna možná jeho synonyma, která se utilitou $ORACLE_HOME\BIN\ctxload.exe do databáze do nového thesauru nahrají.

Předpřipravený slovník synonym address_thesaurus_cz.txt je k dispozici v adresáři slovniky_synonym. Slovník obsahuje seznam nejběžněji využívaných synonym.

Více informací k práci s thesaurus souborem a jeho struktuře:

http://docs.oracle.com/cd/B28359_01/text.111/b28303/cthes.htm#BABEDIAG

Po nahrazení červeně vyznačených parametrů (jménem vlastníka RUIAN dat, jeho databázovým heslem, TNS aliasem databáze s RUIAN daty a cestou k souboru address_thesaurus_cz.txt) lze vytvořit thesaurus cz_address_thesaurus následovně:

```
ctxload -user RUIANUSER/RUIANPASSWD@TNS_ALIAS -thes -name cz_address_thesaurus -
file cesta k address_thesaurus_cz.txt souboru
```

3.1.6 Test fulltextové indexace tabulky ADRESNIMISTO a adresního slovníku synonym

Následující dva dotazy by měli vrátit stejné výsledkové sady obsahující vždy (v závislosti na indexovaných datech) jak adresy s názvem „Bratří Čapků“ tak s názvem „Bří. Čapků“:

```
select * from AdresniMisto where CONTAINS(Adresa,'cap% AND (bri|SYN(bri,
cz_address_thesaurus))')>0
```

```
select * from AdresniMisto where CONTAINS(Adresa,'cap% AND (bratri|SYN(bratri,
cz_address_thesaurus))')>0
```

3.1.7 Změna typu sloupce nazev tabulky KATASTRALNIUZEMI před fulltext indexací sloupce NAZEV

Podobně jako u výše indexované tabulky ADRESNIMISTO je potřeba před indexací tabulky KATASTRALNIUZEMI změnit datový typ z nvarchar2 na varchar2.

```
ALTER TABLE RUIANUSER.KATASTRALNIUZEMI ADD NAZEV_NEW VARCHAR2(48 CHAR);
```

```
UPDATE RUIANUSER.KATASTRALNIUZEMI SET NAZEV_NEW=NAZEV;
```

```
ALTER TABLE RUIANUSER.KATASTRALNIUZEMI RENAME COLUMN NAZEV TO NAZEV_OLD;
```

```
ALTER TABLE RUIANUSER.KATASTRALNIUZEMI RENAME COLUMN NAZEV_NEW TO NAZEV;
```

Po kontrole zda došlo ke zkopírování a přejmenování lze taktéž odstranit původní, přejmenovaný nvarchar2 sloupec.

```
ALTER TABLE RUIANUSER.KATASTRALNIUZEMI DROP COLUMN NAZEV_OLD;
```

3.1.8 Fulltextová indexace tabulky KATASTRALNIUZEMI

V případě, že již byla vytvořena preference (cust_lexer s parametrem base_letter=yes) pro akcent nesensitivní fulltext vyhledávání (viz výše indexace tabulky ADRESNIMISTO), lze tabulku KATASTRALNIUZEMI příkazem indexovat:

```
CREATE INDEX RUIANUSER.KATASTRALNIUZEMI_FTIDX ON RUIANUSER.KATASTRALNIUZEMI (NAZEV)
INDEXTYPE IS CTXSYS.CONTEXT PARAMETERS ('LEXER cust_lexer STOPLIST
CTXSYS.EMPTY_STOPLIST SYNC ( ON COMMIT)');
```

Podobně jako tabulku ADRESNIMISTO je tabulka KATASTRALNIUZEMI s prázdným stop-listem (pro zařazení do indexace názvů i např. předložek „u“ a „nad“ aj.).

3.1.9 Změna typu sloupce ulice tabulky ULICE před fulltext indexací sloupce ulice

Podobně jako u výše indexované tabulky ADRESNIMISTO je potřeba před indexací tabulky ULICE změnit datový typ z nvarchar2 na varchar2.

```
ALTER TABLE RUIANUSER.ULICE ADD ULICE_NEW VARCHAR2(97 CHAR);

UPDATE RUIANUSER.ULICE SET ULICE_NEW=ULICE;

ALTER TABLE RUIANUSER.ULICE RENAME COLUMN ULICE TO ULICE_OLD;

ALTER TABLE RUIANUSER.ULICE RENAME COLUMN ULICE_NEW TO ULICE;
```

Po kontrole zda došlo ke zkopírování a přejmenování lze taktéž odstranit původní, přejmenovaný nvarchar2 sloupec.

```
ALTER TABLE RUIANUSER.ULICE DROP COLUMN ULICE_OLD;
```

3.1.10 Fulltextová indexace tabulky ULICE

V případě, že již byla vytvořena preference (cust_lexer s parametrem base_letter=yes) pro akcent nesensitivní fulltext vyhledávání (viz výše indexace tabulky ADRESNIMISTO), lze tabulku ULICE příkazem indexovat:

```
CREATE INDEX RUIANUSER.ULICE_FTIDX ON RUIANUSER.ULICE (ULICE) INDEXTYPE IS
CTXSYS.CONTEXT PARAMETERS ('LEXER cust_lexer STOPLIST CTXSYS.EMPTY_STOPLIST SYNC (
ON COMMIT)');
```

Podobně jako tabulku ADRESNIMISTO je tabulka ULICE s prázdným stop-listem (pro zařazení do indexace názvů i např. předložek „u“ a „nad“ aj.).

3.1.11 Změna typu sloupce katastralniuzemicisloparcely tabulky PARCELADEFINICNIBOD před fulltext indexací sloupce katastralniuzemicisloparcely

Podobně jako u výše indexované tabulky ADRESNIMISTO je potřeba před indexací tabulky PARCELADEFINICNIBOD změnit datový typ z nvarchar2 na varchar2.

```
ALTER TABLE RUIANUSER.PARCELADEFINICNIBOD ADD KATASTRALNIUZEMICISLOPARCELY_N
VARCHAR2(60 CHAR);

UPDATE RUIANUSER.PARCELADEFINICNIBOD SET
KATASTRALNIUZEMICISLOPARCELY_N=KATASTRALNIUZEMICISLOPARCELY;

ALTER TABLE RUIANUSER.PARCELADEFINICNIBOD RENAME COLUMN
KATASTRALNIUZEMICISLOPARCELY TO KATASTRALNIUZEMICISLOPARCELY_O;

ALTER TABLE RUIANUSER.PARCELADEFINICNIBOD RENAME COLUMN
KATASTRALNIUZEMICISLOPARCELY_N TO KATASTRALNIUZEMICISLOPARCELY;
```

Po kontrole zda došlo ke zkopírování a přejmenování lze taktéž odstranit původní, přejmenovaný nvarchar2 sloupec.

```
A ALTER TABLE RUIANUSER.PARCELADEFINICNIBOD DROP COLUMN
KATASTRALNIUZEMICISLOPARCELY_O;
```

3.1.12 Fulltextová indexace tabulky PARCELADEFINICNIBOD

V případě, že již byla vytvořena preference (cust_lexer s parametrem base_letter=yes) pro akcent nesensitivní fulltext vyhledávání (viz výše indexace tabulky ADRESNIMISTO), lze tabulku PARCELADEFINICNIBOD příkazem indexovat:

```
CREATE INDEX RUIANUSER.PARCELADEFINICNIBOD_FTIDX ON RUIANUSER.PARCELADEFINICNIBOD
(KATASTRALNIUZEMICISLOPARCELY) INDEXTYPE IS CTXSYS.CONTEXT PARAMETERS ('LEXER
cust_lexer STOPLIST CTXSYS.EMPTY_STOPLIST SYNC ( ON COMMIT)');
```

Podobně jako tabulku ADRESNIMISTO je tabulka PARCELADEFINICNIBOD s prázdným stop-listem (pro zařazení do indexace názvů i např. předložek „u“ a „nad“ aj.).

3.1.13 Tvorba a konfigurace pomocné tabulky FTS_TABLES

Skript na vytvoření tabulky FTS_TABLES:

```
CREATE TABLE RUIANUSER.FTS_TABLES
(TABLE_NAME VARCHAR2(63 CHAR),
  FIELD_NAME VARCHAR2(63 CHAR),
      CONFIG VARCHAR2(4000 CHAR));
```

A příkazy na její naplnění defaultní konfigurací pro tabulky ADRESNIMISTO , KATASTRALNIUZEMI, ULICE a PARCELADEFINICNIBOD:

```
insert into RUIANUSER.FTS_TABLES (TABLE_NAME,FIELD_NAME,CONFIG) values
('ADRESNIMISTO','ADRESA','{"where":"contains(Adresa,replace(convert(''{0}'',''US7AS
CII''),''.'',''''))) > 0","string": "({0}% | SYN({0},
cz_address_thesaurus))","number": "({0} | SYN({0}, cz_address_thesaurus))" }');

insert into RUIANUSER.FTS_TABLES (TABLE_NAME,FIELD_NAME,CONFIG) values
('KATASTRALNIUZEMI','NAZEV','{"where":"contains(Nazev,replace(''{0}'',''.'',''''))
> 0","string": "{0}%","number": "{0}"  }');

insert into RUIANUSER.FTS_TABLES (TABLE_NAME,FIELD_NAME,CONFIG) values
('ULICE','ULICE','{"where":"contains(Ulice,replace(convert(''{0}'',''US7ASCII''),''
.'',''''))) > 0","string": "({0}% | SYN({0}, cz_address_thesaurus))","number": "({0}
| SYN({0}, cz_address_thesaurus))" }');

insert into RUIANUSER.FTS_TABLES (TABLE_NAME,FIELD_NAME,CONFIG) values
('PARCELADEFINICNIBOD','KATASTRALNIUZEMICISLOPARCELY','{"where":"contains(KATASTRAL
NIUZEMICISLOPARCELY,replace(''{0}'',''.'',''''))) > 0","string": "{0}%","number":
"{0}"  }');

COMMIT;
```

3.2 Konfigurace fulltextových indexů v databázi SQL Server

3.2.1 Požadavky/předpoklady

1) V následujících skriptech je třeba upravit červeně zvýrazněné jména databáze a schématu, kde se RUIAN data nachází (ve skriptu *RUIAN_DB* a *RUIAN_SCHEMA*)
2) Skript vyžaduje (minimálně pro nahrání upraveného slovníku synonym) práva role **sysadmin** či spuštění pod **systémovým administrátorem**.
3) Schéma tabulek AdresniMisto a KatastralniUzemi odpovídá stavu, tak jak je importoval nástroj VFR Import (např. tabulka AdresniMisto má, níže indexovaný, sloupec adresa)
4) Testováno pro SQL Server 2008 R2. (není vyloučena nutnost změn v novějších verzích SQL Serveru).
5) Po provedení úprav je doporučeno vyhledávání v obou tabulkách podrobit důkladnému testování.

3.2.2 Tvorba katalogu a definování vlastního stop-listu pro fulltextovou indexaci

V případě že SQL Server nemá definovaný katalog (akcent nesensitivní) pro fulltextové vyhledávání, je potřeba tento následujícím způsobem vytvořit. (V případě, že takový katalog již existuje, je možné ho použít dále ve skriptech vytvářejících fulltext indexy, místo právě vytvořeného)

```
USE [RUIAN_DB]
go
create fulltext catalog FTSCatalog with accent_sensitivity = off as default;
go
```

Je potřeba definovat prázdný uživatelský stop-list (seznam slov, které se vynechávají z indexace) pro eliminaci vyjmutí částí geografických názvu při indexaci. (Jeho použití zaručí, že se budou i v budoucích verzích obsahujících český/slovenský stop-list indexovat všechny části názvů včetně např. předložek „nad“/ „u“) .

```
create fulltext stoplist custom_stoplist;
go
```

3.2.3 Fulltextová indexace tabulky AdresniMisto

Fulltext index využívá při zápisu výskytu indexovaného slova, již v tabulce existujícího, unikátního, nesloženého, ne-nullového indexu (v geodatabázi tyto požadavky automaticky splňuje index primárního klíče). Jméno tohoto indexu lze zjistit pro třídu/tabulku AdresniMisto následujícím dotazem:

```
USE [RUIAN_DB]
go
```

```
SELECT Col.Column_Name, Tab.CONSTRAINT_NAME from
    INFORMATION_SCHEMA.TABLE_CONSTRAINTS Tab,
    INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE Col
WHERE
    Col.Constraint_Name = Tab.Constraint_Name
    AND Col.Table_Name = Tab.Table_Name
    AND Constraint_Type = 'PRIMARY KEY'
    AND Col.Table_Name = 'AdresniMisto';
```

Následující příkaz pro fulltextovou indexaci tabulky AdresniMisto je nutné tedy nejprve upravit o jméno právě zjištěného primárního klíče a schématu, kde se nachází indexovaná tabulka AdresniMisto.

```
create fulltext index on [RUIAN_SCHEMA].AdresniMisto (Adresa language
'Slovak')
key index <jméno zjištěného primárního klíče tabulky AdresniMisto>
on FTSCatalog with change_tracking = auto, stoplist = custom_stoplist;
go
```

Pozn.: Indexace byla otestována pro language=Slovak. Od verze SQL Server 2012 je možné využít pro indexaci language=Czech, a tedy příkazy pro vytvoření fulltext indexů zapisovat s nastavením `language 'Czech'`.

3.2.4 Fulltextová indexace tabulky KatastralniUzemi

Zjištění indexu primárního klíče tabulky KatastralniUzemi:

```
USE [RUIAN_DB]
go
SELECT Col.Column_Name, Tab.CONSTRAINT_NAME from
    INFORMATION_SCHEMA.TABLE_CONSTRAINTS Tab,
    INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE Col
WHERE
    Col.Constraint_Name = Tab.Constraint_Name
    AND Col.Table_Name = Tab.Table_Name
    AND Constraint_Type = 'PRIMARY KEY'
    AND Col.Table_Name = 'KATASTRALNIUZEMI'
```

Fulltext indexaci proveďte následujícím příkazem upraveným o jméno schématu a název zjištěného primárního klíče:

```
create fulltext index on [RUIAN_SCHEMA].KATASTRALNIUZEMI (Nazev language
'Slovak')
```

```
key index <jméno zjištěného primárního klíče tabulky KatastralniUzemi>
on FTSCatalog with change_tracking = auto, stoplist = custom_stoplist;
go
```

3.2.5 Fulltextová indexace tabulky Ulice

Zjištění indexu primárního klíče tabulky Ulice:

```
USE [RUIAN_DB]
go
SELECT Col.Column_Name, Tab.CONSTRAINT_NAME from
    INFORMATION_SCHEMA.TABLE_CONSTRAINTS Tab,
    INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE Col
WHERE
    Col.Constraint_Name = Tab.Constraint_Name
    AND Col.Table_Name = Tab.Table_Name
    AND Constraint_Type = 'PRIMARY KEY'
    AND Col.Table_Name = 'Ulice'
```

Fulltext indexaci proveďte následujícím příkazem upraveným o jméno schématu a název zjištěného primárního klíče:

```
create fulltext index on [RUIAN_SCHEMA].Ulice (Ulice language 'Slovak')
key index <jméno zjištěného primárního klíče tabulky Ulice>
on FTSCatalog with change_tracking = auto, stoplist = custom_stoplist;
go
```

3.2.6 Fulltextová indexace tabulky PARCELADEFINICNIBOD

Zjištění indexu primárního klíče tabulky PARCELADEFINICNIBOD:

```
USE [RUIAN_DB]
go
SELECT Col.Column_Name, Tab.CONSTRAINT_NAME from
    INFORMATION_SCHEMA.TABLE_CONSTRAINTS Tab,
    INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE Col
WHERE
    Col.Constraint_Name = Tab.Constraint_Name
    AND Col.Table_Name = Tab.Table_Name
    AND Constraint_Type = 'PRIMARY KEY'
    AND Col.Table_Name = 'Parceladefinicnibod'
```

Fulltext indexaci proveďte následujícím příkazem upraveným o jméno schématu a název zjištěného primárního klíče:

```
create fulltext index on [RUIAN_SCHEMA].Parceladefinicnibod
(katastralniuzemicisloparcely language 'Slovak')
key index <jméno zjištěného primárního klíče tabulky Parceladefinicnibod>
on FTSCatalog with change_tracking = auto, stoplist = custom_stoplist;
go
```

3.2.7 Test provedené fulltextové indexace

Následující 4 dotazy do správně fulltextově indexovaných tabulek by neměly vrátit chybu.

```
Cannot use a CONTAINS or FREETEXT predicate…
```

```
select * from AdresniMisto where CONTAINS(Adresa,'tri & facky & suchdol')
select * from KatastralniUzemi where CONTAINS(Nazev, 'krecovice &
neveklova')
select * from Ulice where CONTAINS(Ulice,'tri & facky & suchdol')
select * from Parceladefinicnibod where
CONTAINS(katastralniuzemicisloparcely, 'krecovice & neveklova')
```

Pozn. plnění fulltext indexů provádí proces na pozadí, a po indexaci velké tabulky toto plnění může chvíli trvat. V případě spuštění kontrolních dotazů, ve chvíli kdy se index teprve plní, může dotaz díky tomu trvat dlouho. Zda je index již naplněn, lze poznat podle 0% CPU využití procesem fdhost, případně následujícím dotazem:

```
SELECT
    CAT.name,
    FULLTEXTCATALOGPROPERTY(cat.name,'ItemCount') ItemCount,
    FULLTEXTCATALOGPROPERTY(cat.name,'MergeStatus') MergeStatus,
    FULLTEXTCATALOGPROPERTY(cat.name,'PopulateCompletionAge')
PopulateCompletionAge,
    FULLTEXTCATALOGPROPERTY(cat.name,'PopulateStatus') PopulateStatus,
    FULLTEXTCATALOGPROPERTY(cat.name,'ImportStatus') ImportStatus
FROM sys.fulltext_catalogs AS cat
```

PopulateStatus=1 značí u fulltext katalogu jeho aktuální plnění (plnění některého z jeho indexů).

3.2.8 Úprava slovníku synonym

Pro správné vyhledávání v adresách běžně používaných zkratek a synonym. Např. „ kpt.“ ve smyslu kapitána, „bří“ ve smyslu „bratří“ (a obráceně), je potřeba definovat slovník synonym tzv. thesaurus. SQL Server tento slovník umožňuje definovat v souborech defaultně v umístění (pro SQL Server 2008 R2).

<cesta k SQL Server datovým souborům \MSSQL10_50.MSSQLSERVER\MSSQL\FTDATA\

V tomto umístění lze pro, ve fulltext indexech použitou slovenštinu, upravit soubor tssky.xml

Předpřipravený slovník synonym tssky.xml je k dispozici v adresáři slovniky_synonym. Slovník obsahuje seznam nejběžněji využívaných synonym. Tímto tedy lze nahradit (nejlépe zazálohovaný) soubor tssky.xml v uvedeném umístění.

Po této úpravě se nový seznam synonym nahraje (pod sys admin uživatelem či uživatelem se sysdamin rolí) následujícím příkazem:

```
USE [RUIAN_DB]
go
EXEC sys.sp_fulltext_load_thesaurus_file 1051;
GO
```

Kde 1051 je kód pro, ve fulltext indexu specifikovanou, slovenštinu. Znamenající v tomto kontextu, že k tomuto jazyku se daná synonyma váží. (Kód lze zkontrolovat dotazem do pohledu sys.fulltext_languages)

Pozn. V případě, že nahrání slovníku synonym končí chybovou hláškou … Reason: 15105, může být problém s právy k danému tssky.xml souboru. Toto lze řešit pod Administrátorem dočasnou úpravou práv k souboru tak, aby na čtení měl právo kdokoliv, slovník lze následně nahrát a poté práva upravit do výchozího nastavení.

Slovník lze také prostřednictvím souboru tssky.xml upravovat a rozšiřovat o další synonyma a zkratky. Stačí do daného xml přidávat tagy <expansion> definující seznam všech synonym k jednomu slovu. A po uložení souboru , změny opět prostřednictvím funkce `sys.sp_fulltext_load_thesaurus_file` do slovníku nahrát.

Např. pro zkratku „bří.“ ve smyslu „bratří“.

…

<expansion>

_{bri}

_{bratri}

</expansion>

…

Z provedených testů však vyplývá, že je vhodné se vyhnout speciální znakům a víceslovním synonymům a je vhodné podrobit vyhledání definovaných synonym důkladnému ověření nalezených záznamů.

Pozn.: Pro případ využití češtiny od verze SQL Server 2012 je ve stejném umístění připravený slovník synonym tscze.xml. Příkaz pro jeho nahrání má potom zápis:

```
USE [RUIAN_DB]
go
```

```
EXEC sys.sp_fulltext_load_thesaurus_file 1029;
GO
```

kde 1029 je kód pro, ve fulltext indexu specifikovanou, češtinu.

Blíže k úpravě slovníku synonym (thesauru) viz dokumentace:

http://msdn.microsoft.com/en-us/library/ms142491%28v=sql.105%29.aspx

3.2.9 Test definovaných synonym

Funkčnost v předešlém kroku definovaných a nahraných synonym lze ověřit kontrolou výsledků dotazů na výskyt daného slova (v příkladu použitém „bří“ a „bratří“)

```
select * from AdresniMisto where CONTAINS(Adresa,'Čapků &
FORMSOF(THESAURUS,"bří")')
```

```
select * from AdresniMisto where CONTAINS(Adresa,'Čapků &
FORMSOF(THESAURUS,"bratří")')
```

Výsledek by měl (s ohledem na v tabulce AdresniMisto obsažená data) obsahovat jak záznamy „Bratří Čapků“ tak „Bří Čapků“ a měl by být pro oba dotazy shodný.

3.2.10 Tvorba a konfigurace pomocné tabulky FTS_TABLES

Skript na vytvoření tabulky FTS_TABLES:

```
USE [RUIAN_DB]
go
CREATE TABLE [RUIAN_SCHEMA].FTS_TABLES
(table_name nvarchar(128),
field_name  nvarchar(128),
config  nvarchar(4000)
);
go
```

A příkazy na její naplnění defaultní konfigurací pro tabulky AdresniMisto, KatastralniUzemi, Ulice a Parceladefinicnibod:

```
insert into [RUIAN_SCHEMA].FTS_TABLES (table_name,field_name,config) values
('ADRESNIMISTO','Adresa','{"where": "CONTAINS(Adresa, ''{0}'')","string":
"(\"{0}*\" | FORMSOF(THESAURUS,'{0}'))","number": "(\"{0}\" |
FORMSOF(THESAURUS,'{0}'))"}');
go
```

```
insert into [RUIAN_SCHEMA].FTS_TABLES (table_name,field_name, config)
values ('KATASTRALNIUZEMI','Nazev','{"where":
"CONTAINS(Nazev,''{0}'')","string": "\"{0}*\"","number": "{0}"}');

go

insert into [RUIAN_SCHEMA].FTS_TABLES (table_name,field_name,config) values
('ULICE','ulice','{"where": "CONTAINS(ulice, ''{0}'')","string": "(\"{0}*\"
| FORMSOF(THESAURUS,'{0}'))","number": "(\"{0}\" |
FORMSOF(THESAURUS,'{0}'))"}');
go

insert into [RUIAN_SCHEMA].FTS_TABLES (table_name,field_name, config)
values ('PARCELADEFINICNIBOD','katastralniuzemicisloparcely','{"where":
"CONTAINS(katastralniuzemicisloparcely,''{0}'')","string":
"\"{0}*\"","number": "{0}"}');
go
```

3.3 Konfigurace fulltextových indexů v databázi PostgreSQL

Následující kapitola popisuje úpravy v databázi PostgreSQL, které jsou nezbytné pro správnou funkčnost doplňku Vyhledávání RUIAN.

3.3.1 Požadavky/předpoklady

1) Postup vyžaduje úpravy provedené pod administračním postgres uživatelem.
2) Další skripty předpokládají spuštění pod uživatelem vlastnícím RUIAN tabulky.
3) Schéma tabulek RUIAN odpovídá stavu, tak jak je importoval nástroj VFR Import (např. tabulka ADRESNIMISTO má, níže indexovaný, sloupec adresa)
4) Testováno pro PostgreSQL 9.2. (není vyloučena nutnost změn v novějších verzích PostgreSQL).
5) Po provedení úprav je doporučeno vyhledávání v obou tabulkách podrobit důkladnému testování.

3.3.2 Úpravy pod administračním postgres uživatelem

Před samotnou indexací RUIAN tabulek je potřeba pod postgres administrátorem databáze provést v databázi několik úprav. Je potřeba

- Nainstalovat a upravit funkci unaccent v extenzi unaccent.
- Nahrát soubor se slovníkem synonym.
- Vytvořit a nakonfigurovat fulltext konfiguraci, kterou bude indexace a vyhledávání používat.

Pro správné vyhledávání a převod řetězců bez ohledů na diakritiku je potřeba nainstalovat (v PostgreSQL extenzích defaultně obsaženou extenzi unaccent)

```
CREATE EXTENSION unaccent;
```

V této extenzi je nainstalována funkce unaccent, kterou je však pro prohledávání v to_tsvector funkci upravit a definovat ji jako IMMUTABLE.(Pouze deklaruje že funkce pro daný vstupní řetězec vždy vrátí stejný výsledek.)

```
CREATE OR REPLACE FUNCTION public.unaccent(text)
  RETURNS text AS
'$libdir/unaccent', 'unaccent_dict'
  LANGUAGE c IMMUTABLE STRICT
  COST 1;
```

Pro další použití je také vhodné aby se vlastníkem funkce unaccent stal uživatel vlastnící RUIAN data , aby funkci mohli spouštět i další uživatelé (nahraďte řeťezec RUIANUSER jménem Vašeho RUIAN data vlastnícího uživatele).

```
ALTER FUNCTION public.unaccent(text) OWNER TO RUIANUSER;

GRANT EXECUTE ON FUNCTION public.unaccent(text) TO public;
```

Před další tvorbou fulltextové konfigurace je potřeba nahrát předvyplněný soubor slovníku synonym synonym_address_cz.syn do umístění share\tsearch_data (obvykle do c:\Program Files\PostgreSQL\9.2\share\tsearch_data\). Předpřipravený slovník synonym synonym_address_cz.syn je k dispozici v adresáři slovniky_synonym. Slovník obsahuje seznam nejběžněji využívaných synonym.

Následně je potřeba vytvořit na základě uvedeného souboru slovník synonym cz_syn kde parametr synonyms musí být shodný se jménem (bez přípony) souboru se synonymy.

```
CREATE TEXT SEARCH DICTIONARY public.cz_syn
(TEMPLATE=synonym,synonyms='synonym_address_cz');
```

Po jeho vytvoření již nic nebrání vytvoření a nastavení fulltext search konfigurace, která synonyma využívá.

```
CREATE TEXT SEARCH CONFIGURATION public.simple_syn (

  PARSER = "default"

);


ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR asciihword WITH
cz_syn,simple;

ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR asciiword WITH
cz_syn,simple;

ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR email WITH
simple;

ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR file WITH simple;

ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR float WITH
simple;

ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR host WITH simple;

ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR hword WITH
cz_syn,simple;

ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR hword_asciipart
WITH simple;

ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR hword_numpart
WITH simple;

ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR hword_part WITH
cz_syn,simple;

ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR int WITH simple;

ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR numhword WITH
simple;
```

```
ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR numword WITH
simple;

ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR sfloat WITH
simple;

ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR uint WITH simple;

ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR url WITH simple;

ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR url_path WITH
simple;

ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR version WITH
simple;

ALTER TEXT SEARCH CONFIGURATION public.simple_syn ADD MAPPING FOR word WITH
cz_syn,simple;
```

3.3.3 Úpravy provedené pod vlastníkem RUIAN dat

Pod vlastníkem dat je potřeba provést následující:

- Indexovat fulltext indexem tabulky Adresnimisto, Katastralniuzemi, Ulice a Parceladefinicnibod.
- Vytvořit a naplnit daty (nakonfigurovat) pomocnou tabulku FTS_TABLES.

Pod vlastníkem RUIAN tabulek je potřeba, pro prohledávání adres a parcelních bodů, fulltextově indexovat tabulky Adresnimisto,Katastralniuzemi, Ulice a Parceladefinicnibod. Pro tuto indexaci je potřeba použít výše definovanou „simple_syn“ konfiguraci a taktéž pomocí regulárních výrazů upravit indexovaný řetězec (indexovat i řetězce se složeným číslem orientačním atd.).

```
CREATE INDEX adresnimisto_ftidx

  ON adresnimisto

  USING gin

  (to_tsvector('simple_syn'::regconfig, unaccent((((replace(adresa::text,
'/'::text, ' '::text) || ' '::text) || COALESCE("substring"(adresa::text, '[0-
9]+/[0-9]+[A-Za-z]?'::text), ''::text)) || ' '::text) ||
COALESCE(ltrim("substring"(adresa::text, '/[0-9]+'::text), '/'::text),
''::text))));


CREATE INDEX katastralniuzemi_ftidx

  ON katastralniuzemi

  USING gin

  (to_tsvector('simple_syn'::regconfig, unaccent((replace(nazev::text, '/'::text, '
'::text) || ' '::text) || COALESCE("substring"(nazev::text, '[0-9]+/[0-9]+'::text),
''::text))));
```

```
CREATE INDEX ulice_ftidx
  ON ulice
  USING gin
  (to_tsvector('simple_syn'::regconfig, unaccent(ulice::text)));

CREATE INDEX parceladefinicnibod_ftidx
  ON parceladefinicnibod
  USING gin
  (to_tsvector('simple_syn'::regconfig,
(replace(katastralniuzemicisloparcely::text, '/'::text, ' '::text) || ' '::text) ||
COALESCE("substring"(katastralniuzemicisloparcely::text, '[0-9]+/[0-9]+'::text),
''::text)));
```

Doplněk Vyhledávání RUIAN používá k dynamickému sestavení where podmínky s fulltextovým vyhledáváním, záznamy uložené v pomocné tabulce FTS_TABLES. Tato obsahuje v JSON formátu zapsanou část podmínky spolu s pravidly pro dotazování na hodnotu řetězcovou a číselnou. Aplikace na základě vstupní dotazované sekvence slov slova vyhodnotí a dle toho, zda jde o číslo či řetězec, je na základě nalezených pravidel cyklicky s logickým operátorem (& ~ "AND") do podmínky doplní. Tabulka FTS_TABLES musí být umístěna ve stejné databázi a schématu jako indexované a aplikacemi přistupované RUIAN tabulky. Uživatel, který k databázi prostřednictvím SOE a ADD-IN přistupuje, musí data v tabulce FTS_TABLES vidět.

Tabulka FTS_TABLES se vytvoří a pro tabulky adresnimisto,katastralniuzemi,ulice a parceladefinicnibod naplní následovně:

```
CREATE TABLE fts_tables
(
  table_name character varying(63),
  field_name character varying(63),
  config character varying(4000)
)
WITH (
  OIDS=FALSE
);
insert into FTS_TABLES (table_name,field_name,config) values
('adresnimisto','adresa','{"where": "to_tsvector(''simple_syn''::regconfig,
unaccent((((replace(adresa::text, ''/''::text, '' ''::text) || '' ''::text) ||
COALESCE(\"substring\"(adresa::text, ''[0-9]+/[0-9]+[A-Za-z]?''::text),
''''::text)) || '' ''::text) || COALESCE(ltrim(\"substring\"(adresa::text, ''/[0-
```

```
9]+''::text), ''/''::text), ''''::text)))@@
to_tsquery(''simple_syn'',unaccent(''{0}''))","string": "{0}:*","number": "{0}"}')

insert into FTS_TABLES (table_name,field_name,config) values
('katastralniuzemi','nazev','{"where": "to_tsvector(''simple_syn''::regconfig,
unaccent((replace(nazev::text, ''/''::text, '' ''::text) || '' ''::text) ||
COALESCE(\"substring\"(nazev::text, ''[0-9]+/[0-9]+''::text), ''''::text)))  @@
to_tsquery(''simple_syn'',unaccent(''{0}''))","string": "{0}:*","number": "{0}"}');

insert into FTS_TABLES (table_name,field_name,config) values
('ulice','ulice','{"where": "to_tsvector(''simple_syn''::regconfig,
unaccent(ulice::text)) @@ to_tsquery(''simple_syn'',unaccent(''{0}''))","string":
"{0}:*","number": "{0}"}');

insert into FTS_TABLES (table_name,field_name,config) values
('parceladefinicnibod','katastralniuzemicisloparcely','{"where":
"to_tsvector(''simple_syn''::regconfig,
(replace(katastralniuzemicisloparcely::text, ''/''::text, '' ''::text) || ''
''::text) || COALESCE(\"substring\"(katastralniuzemicisloparcely::text, ''[0-
9]+/[0-9]+''::text), ''''::text))  @@
to_tsquery(''simple_syn'',unaccent(''{0}''))","string": "{0}:*","number": "{0}"}');
```

V případě že zvolený nástroj (např. psql) automaticky neprovadí commit:

```
commit;
```

3.3.4 Úpravy slovníku synonym

Použitý slovník adresních synonym lze uživatelem rozšířit či upravit, je však potřeba dbát určitých pravidel. Slovník se skládá jednoduše ze dvou sloupců textového souboru (s příponou .syn), kde v prvním sloupci je uvedené synonymum a v druhém sloupci jeho význam. Tedy duplicitně se slova mohou opakovat pro více synonym k jednomu slovu v druhém sloupci.

Např.

…

mudr doktora

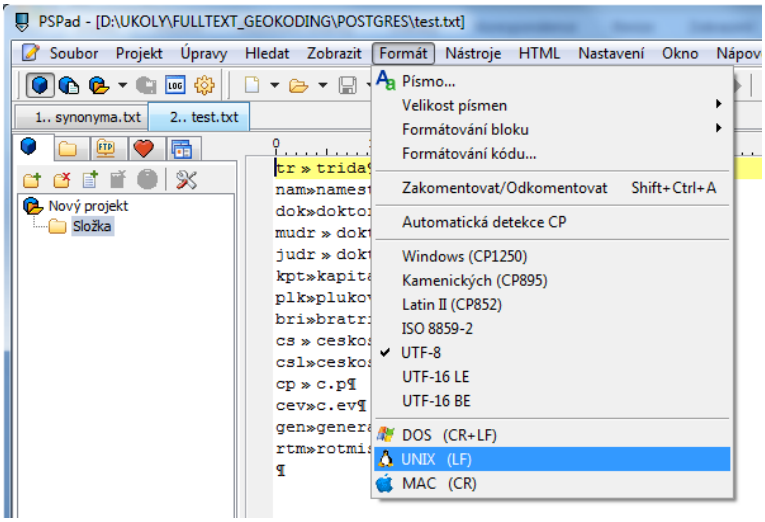
judr doktora

…

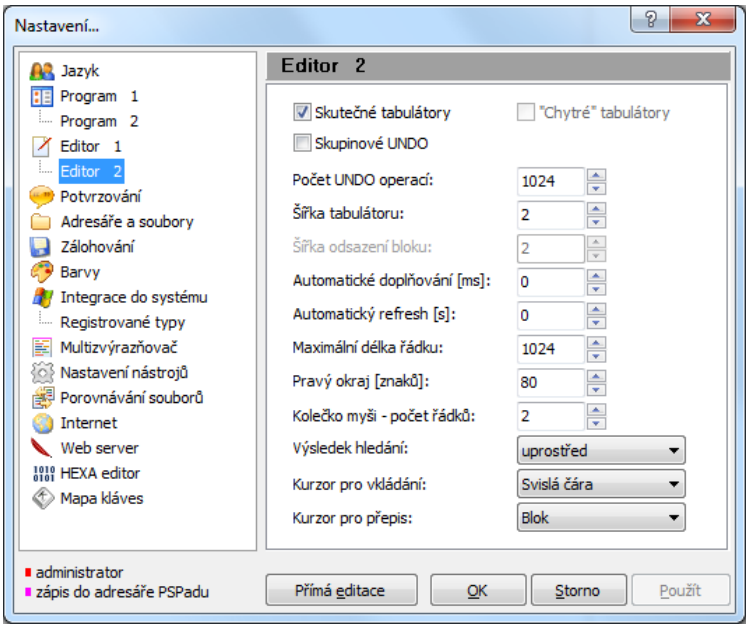
Doporučujeme vyhnout se nealfabetickým znakům, diakritice, tečkám na koncích slov, a víceslovným synonymům.

Soubor vyžaduje, aby mezera mezi sloupci byl znak tabulátoru #09 a konce řádků označoval pouze znak #0A (line feed bez znaku #0D carriage return)

Např. v programu PSPad lze tento soubor editovat s následujícím nastavením.



Kliknutím na menu Formát s volbou UNIX (LF) se při editaci pro oddělení řádků požije pouze znak #0A



Zaškrtnutím volby „Skutečené tabulátory“ v menu Nastavení>Nastavení programu...>v záložce Editor 2 se vynutí použití znaku #09 pro oddělující tabulátor.

Po úpravě souboru synonym se změny projeví pouze u připojení, které se připojili až po provedení změn či u těch, které slovník prvně použili až po provedení změn. U již existujících připojení, které slovník použili již před změnami, se automaticky změny neprojeví. Pro aktualizaci slovníku cz_syn i u těchto lze pod postgres administratátorem spustit následující:

```
ALTER TEXT SEARCH DICTIONARY public.cz_syn ( dummy );
```

Jakoukoliv změnu v slovníku synonym je potřeba následně podrobit důkladnému otestování.

Více informací k úpravě slovníku synonym:

http://www.postgresql.org/docs/9.2/static/textsearch-dictionaries.html