Testovací projekt v prostředí MarushkaDesign

OBSAH

1 Průvodce testovacím projektem v MarushkaDesignu

1.1 Cíl testovacího projektu

Cílem testovacího projektu v MarushkaDesignu je předvedení funkčnosti a možností, které MarushkaDesign nabízí. V testovacím projektu si ukážeme krok po kroku, jak vytvořit funkční projekt mapové kompozice a jakým způsobem vytvořit webovou mapovou publikaci dat, která máme k dispozici. Výsledkem bude interaktivní webová mapová kompozice v prostředí Marushky, která bude obsahovat základní přehledku, bude umožňovat základní prohlížení dat, vkládání a editaci databázových údajů. Webová Marushka bude v tomto případě umět zobrazovat informace o prvcích, bude je umět zlokalizovat, bude umět kreslit nové grafické elementy a mazat uživatelem nakreslené elementy z datového skladu. K jednotlivým skupinám elementů bude zobrazována legenda, samozřejmostí je i možnost tisku do formátu PDF, PNG nebo na tiskárnu.

Hlavní datový sklad bude uložen v prostředí **SQLite**, pro jehož naplnění nám poslouží výkresy ESRI shape files fiktivních dat vodovodní sítě, která byla zakreslena nad reálnou katastrální mapou katastrálního území Pardubice - Pardubičky. Databázové prostředí SQLite předpokládá mírně pokročilou uživatelskou znalost jazyka SQL.

„Ostré projekty“ se od tohoto testovacího mohou odlišovat – např. při hlavním databázovém zdroji Oracle a vektorových datech ve formátu wkb není třeba řešit komplikace související s omezením ESRI shape files, u kterých si musíme poradit s problémy souvisejícími např. s buňkami. V již existujících GS projektech máme k dispozici knihovny buněk, knihovny uživatelských stylů apod., ale tím se pak nemusíme vůbec zabývat. Základní principy jsou ale v prostředí ESRI shape files, které většina uživatelů dobře zná, dobře vysvětlitelné.

Předmětem tohoto návodu k vytvoření projektu není podrobný popis prostředí MarushkaDesignu, ale poskytuje jenom základní a přitom co nejširší spektrum možností, které MarushkaDesign nabízí. Ke komplexní představě o funkcích a práci v prostředí MarushkaDesignu se můžeme dočíst v manuálu, který je součástí instalačního balíčku MarushkaDesignu.

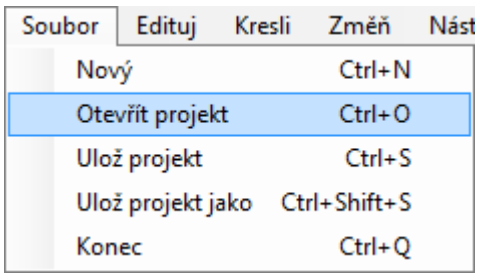
1.2 Soubory testovacího projektu

Kromě již vytvořeného hlavního souboru `projekt.xml` a databázového souboru `test_project.db3` se ve složce Tutorialu nachází ještě dvě další složky: SHP a Soubory. Soubory umístěné v těchto složkách využijeme při samostatné tvorbě projektu. Ve složce SHP jsou to zdrojové ESRI shape files a ve složce Soubory jsou to především dva xml soubory – `Bunky.xml` – knihovna vektorových buněk potřebná pro zobrazení buněk z bodů uložených v shape files a `NewWMS.xml` – podprojekt obsahující sloučené vrstvy katastrální mapy z wms zdroje ČÚZK. Dále jsou to soubory `png` s podtržítkem na začátku (budou sloužit jako rastrové buňky pro vytvoření legendy), soubory `gif` začínající znaky „rb_“ – rovněž rastrové buňky legendy a soubor `pripojka.jpg`, který nám poslouží jako vzorový obrázek při práci s dokumenty. Nakonec je to ještě soubor SQL_Create.sql, který obsahuje zakládací skripty nutné k vytvoření potřebných databázových tabulek.

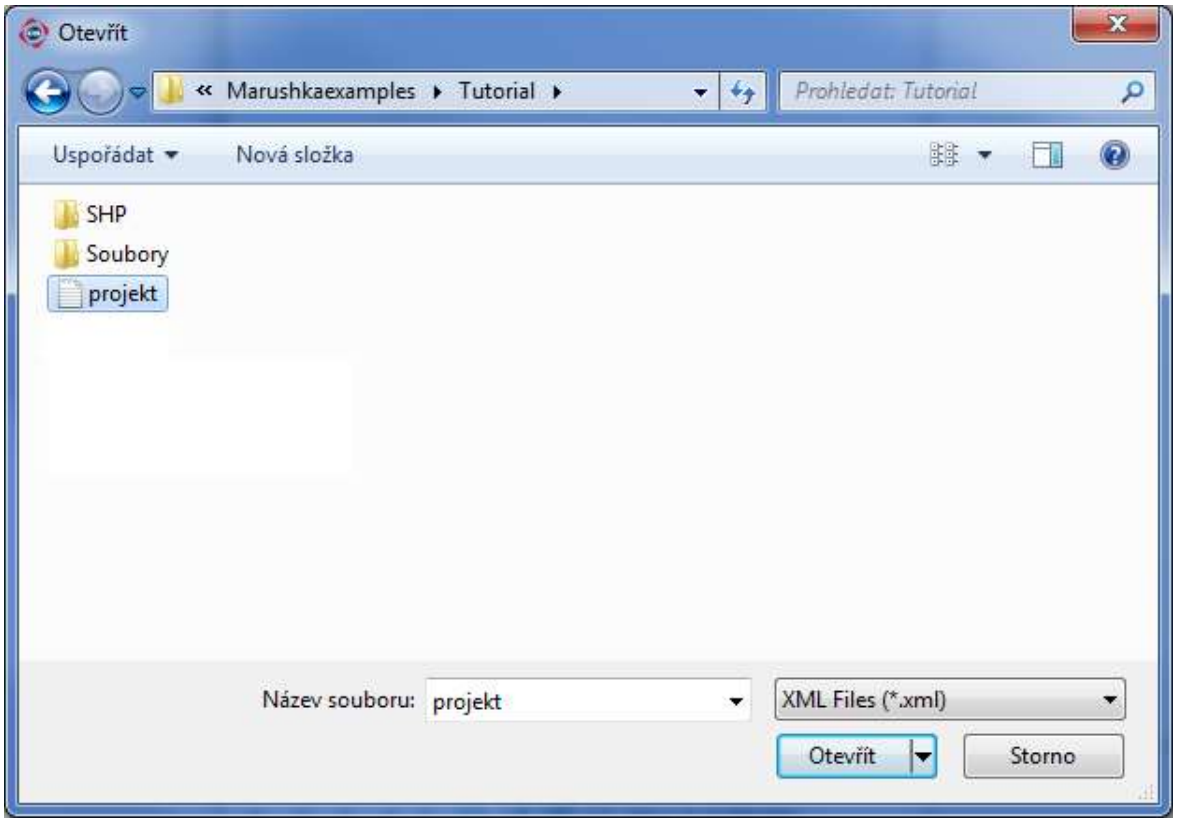
1.3 Otevření vytvořeného projektu

Projekt si uložíme do složky **C:MarushkaExamples\Tutorial**, po spuštění MarushkaDesignu (v systémech Windows Vista a vyšších je vhodné soubor MarushkaDesign.exe „spouštět jako správce“ kvůli zápisům logu do systémové složky Program Files) otevřeme projekt – soubor - **C:\MarushkaExamples\Tutorial\projekt.xml**. V otevřeném projektu si sami můžeme vyzkoušet funkce, které jsou v tomto projektu vytvořeny a které si následně vyzkoušíme v prostředí MarushkaDesignu vytvořit.

- Otevření projektu:

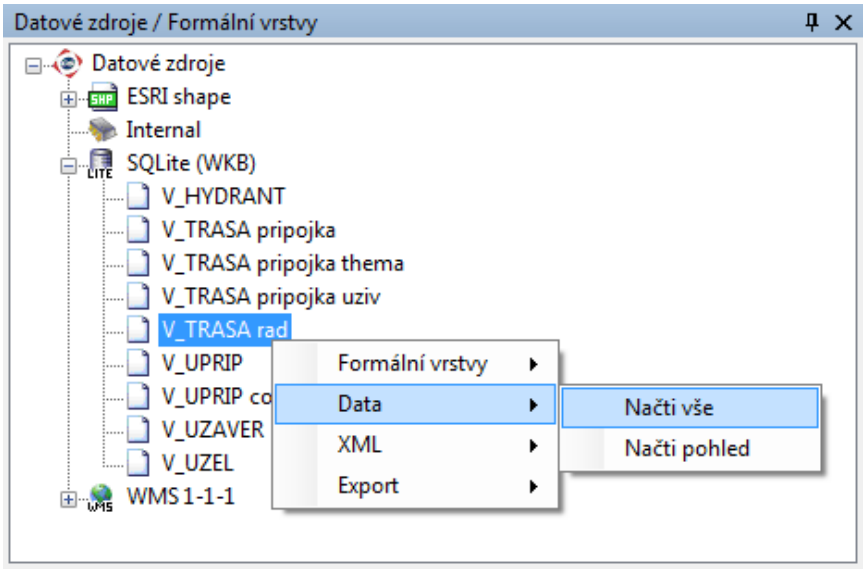


- Ze složky MarushkaExamples\Tutorial si otevřeme soubor projekt.xml:



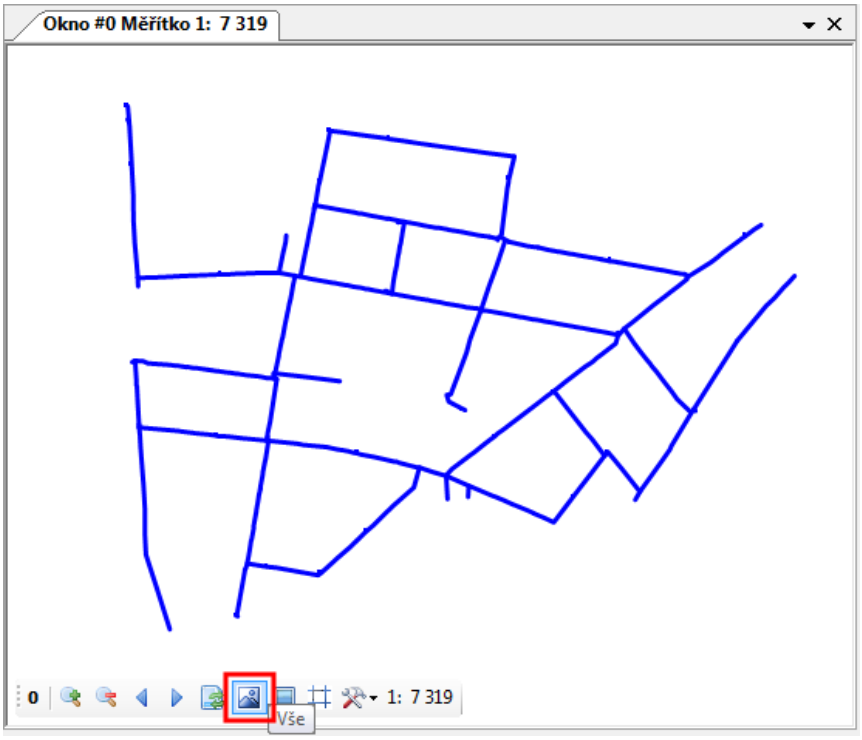
Po otevření projektu se nám načtou veškerá data v něm uložená. Objeví se nám všechny *Datové zdroje* obsažené v našem projektu. Pro okamžité zobrazení dat v něm obsažených je vhodný následující postup:

- V *Datových zdrojích* vybereme ze zdroje SQLite vrstvu „*V_TRASA rad*“, která se po kliknutí levým tlačítkem myši zvýrazní, a pravým tlačítkem zobrazíme kontextové menu, z něhož vybereme „*Data – Načti vše*“.



Marushka bude několik okamžiků pracovat za nás a zobrazí všechny prvky, které jsou uloženy ve formální vrstvě *V_TRASA rad*. V případě, že nic nevidíme, tak Marushka zobrazila data mimo viditelnou oblast zobrazeného mapového okna.

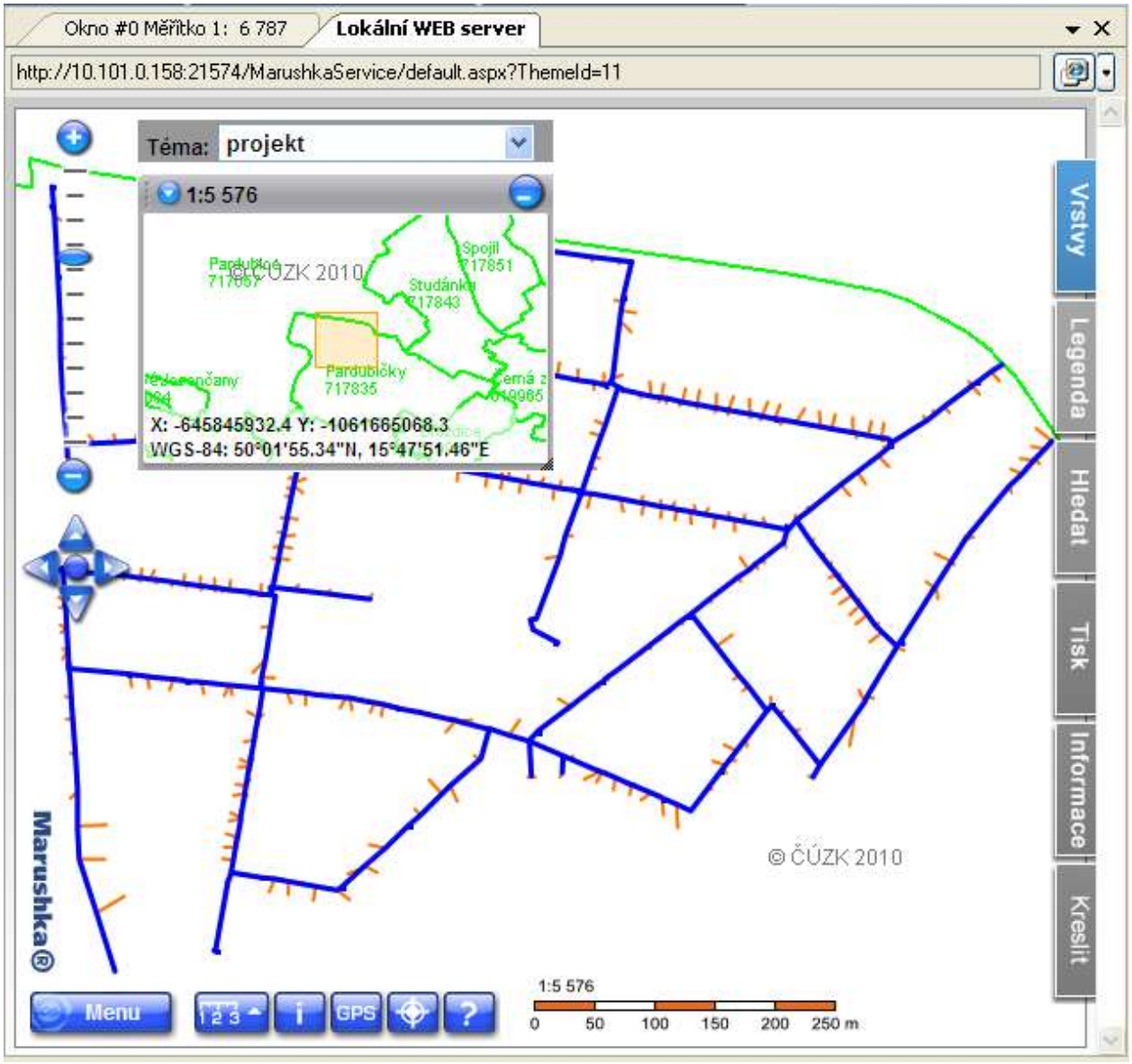
- Zobrazení všech dat v mapovém okně provedeme pomocí kliknutí levého tlačítka myši na ikonu zobrazenou na následujícím obrázku a dostaneme zhruba příslušný výřez dat:



Díky takto definovanému výřezu mapového okna si můžeme data zobrazit i v lokálním WEB serveru v prostředí MarushkaDesignu.

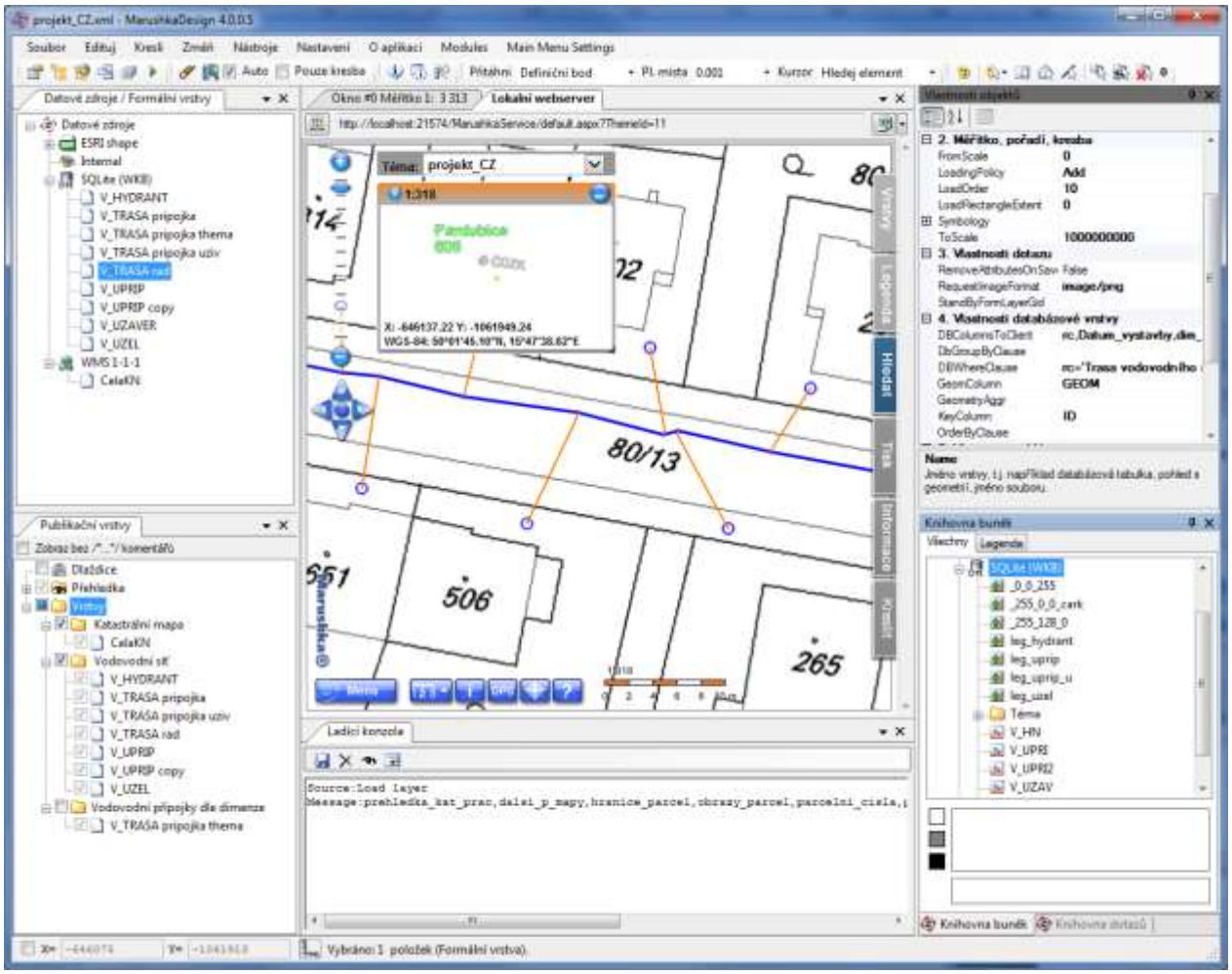
- Lokální WEB server spustíme zvýrazněnou ikonou (z panelu nástrojů *Vrstvy a grafická data*):

- Po spuštění lokálního WEB serveru dostaneme následující obrázek:



Následuje ukázka celého prostředí MarushkaDesignu s otevřeným projektem a náhledem v lokálním WEB serveru:

Nyní už záleží jenom na nás, jak moc si budeme hrát s vytvořeným projektem a sami budeme zkoumat existující možnosti. V následujících kapitolách si už ukážeme, jak tento projekt vytvoříme sami.



2 Datová struktura a složení projektu

Hlavním nositelem datové struktury projektu jsou:

1) Zjednodušená fiktivní data vodovodní sítě zakreslená nad reálnou katastrální mapou (ze zdrojových ESRI shape files budou naimportována do databázového datového skladu SQLite).
2) Veřejná wms služba z ČÚZK, umožňující zobrazení katastrální mapy pod vektorovými daty vodovodní sítě.

2.1 Popis datového modelu vodovodní sítě

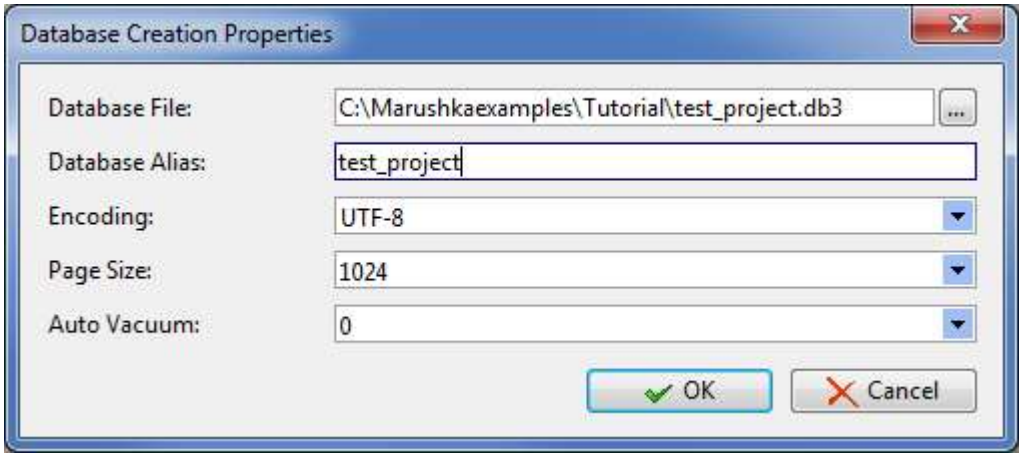
Vodovodní síť se v našem případě skládá z liniových **objektů trasy** vodovodní sítě (řadů a přípojek), definující průběh naší inženýrské sítě. Další vektorová data tvoří **buňky zařízení**, které se na trase vodovodní sítě vyskytují. Jsou to **fiktivní uzly** v křížení více řadů nebo řadem s přípojkou, **hydranty** a především **ukončení přípojek**, které jsou v našem případě i nositeli negrafických údajů o zákaznícovi. Ve výsledné mapové publikaci nejsou zobrazovány uzávěry na přípojkách (příp. řadech).

3 Vytvoření datové struktury v SQLite

Jako zdrojová data máme k dispozici soubory ESRI shape files. Práce s daty v takto uloženém datovém skladu by ve webové publikaci byla značně omezená a nebylo by možné využít databázových vlastností jednotlivých elementů, proto je vhodné si tento zdroj převést do databázové datové struktury. Pro jednoduchost, účelnost a proto, že toto řešení je dostupné každému uživateli, jsme zvolili databázové prostředí SQLite, které je plně otevřené. Databázová struktura prostředí SQLite je souborového charakteru, je tedy velmi snadno přenositelná mezi různými počítačovými stanicemi. Pro správu databázových entit jsme zvolili databázového klienta SQLite Expert Personal (http://www.sqliteexpert.com/download.html). Jedná se o plně dostačujícího SQL managera, práce v něm je poměrně intuitivní, nicméně pro práci mimo rozsah našeho tutoriálu vyžaduje znalosti jazyka SQL.

3.1 Vytvoření databáze

Novou databázi v prostředí klienta SQLite Expert Personal vytvoříme pomocí položky menu *File – New database*. Zobrazí se nám dialogové okno s vlastnostmi vytvářené databáze. Pomocí tlačítka ... vybereme cestu a název k novému databázovému souboru, ostatní vlastnosti ponecháme beze změny.

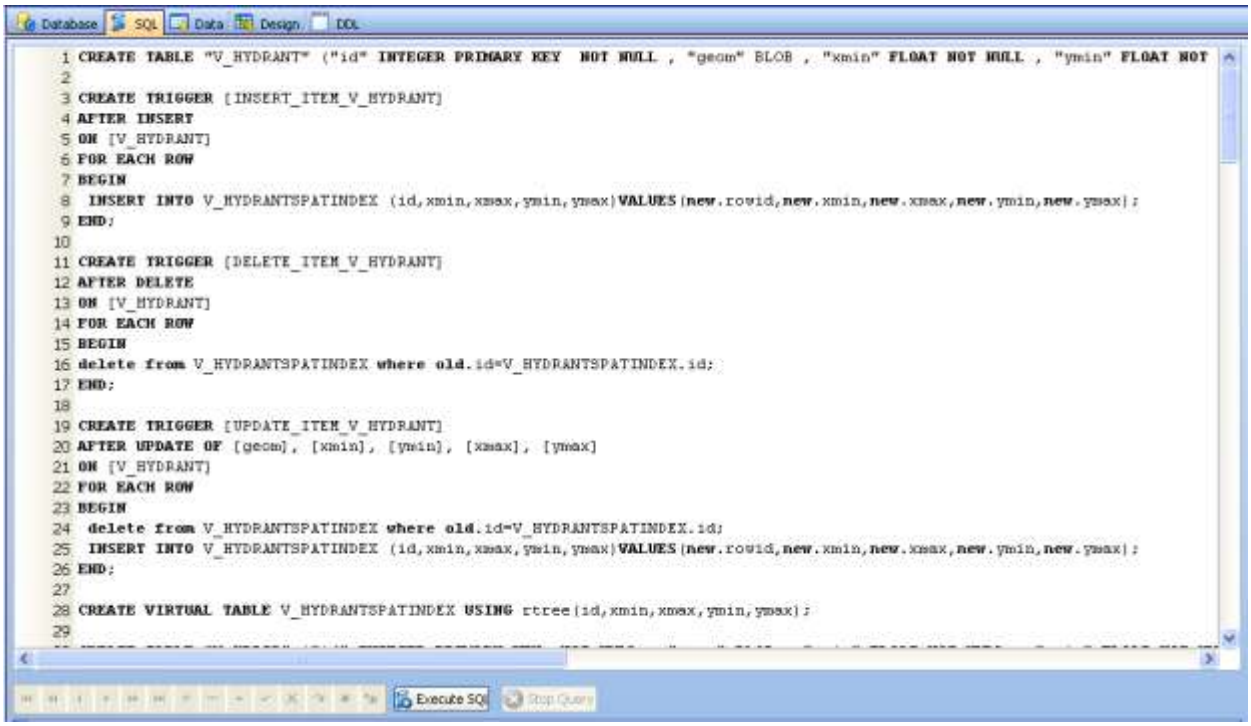


3.2 Zakládací skripty databázových tabulek

Každá grafická tabulka se skládá z několika povinných sloupců a k této tabulce navíc musí existovat tabulka **virtuální** pro prostorovou indexaci. Všechny tabulky mají podobnou strukturu, musí obsahovat sloupce ID, GEOM, XMIN, YMIN, XMAX, YMAX. Další sloupce vytvořených tabulek slouží pro potřeby našeho vytvářeného projektu. Význam sloupců v jednotlivých tabulkách, které nejsou povinné z hlediska struktury jakéhokoliv GS projektu, budou vysvětleny později při samotné konfiguraci projektu.

Jediná negrafická tabulka, kterou v našem případě použijeme, je tabulka „`demo_doc`“. Jde o tabulku dokumentů, ve které budou uloženy jakékoliv soubory náležející konkrétním grafickým elementům.

Zakládací skripty ke všem potřebným tabulkám můžeme najít v souboru `SQL_Create.sql`, který nalezneme v balíčku Tutorialu ve složce `Soubory`. Text v něm obsažený postačí zkopírovat do schránky, vložit do SQLite expert manageru a spustit tlačítkem Execute SQL:




```
CREATE TABLE "V_HYDRANT" ("id" INTEGER PRIMARY KEY  NOT NULL , "geom" BLOB , "xmin" FLOAT NOT NULL , "ymin" FLOAT NOT

CREATE TRIGGER [INSERT_ITEM_V_HYDRANT]
AFTER INSERT
ON [V_HYDRANT]
FOR EACH ROW
BEGIN
 INSERT INTO V_HYDRANTSPATINDEX (id,xmin,xmax,ymin,ymax)VALUES(new.rowid,new.xmin,new.xmax,new.ymin,new.ymax);
END;

CREATE TRIGGER [DELETE_ITEM_V_HYDRANT]
AFTER DELETE
ON [V_HYDRANT]
FOR EACH ROW
BEGIN
delete from V_HYDRANTSPATINDEX where old.id=V_HYDRANTSPATINDEX.id;
END;

CREATE TRIGGER [UPDATE_ITEM_V_HYDRANT]
AFTER UPDATE OF [geom], [xmin], [ymin], [xmax], [ymax]
ON [V_HYDRANT]
FOR EACH ROW
BEGIN
 delete from V_HYDRANTSPATINDEX where old.id=V_HYDRANTSPATINDEX.id;
 INSERT INTO V_HYDRANTSPATINDEX (id,xmin,xmax,ymin,ymax)VALUES(new.rowid,new.xmin,new.xmax,new.ymin,new.ymax);
END;

CREATE VIRTUAL TABLE V_HYDRANTSPATINDEX USING rtree(id,xmin,xmax,ymin,ymax);
```

3.2.1 Grafické tabulky

Hydrant

```
CREATE TABLE "V_HYDRANT" ("id" INTEGER PRIMARY KEY  NOT NULL , "geom" BLOB ,
"xmin" FLOAT NOT NULL , "ymin" FLOAT NOT NULL , "xmax" FLOAT NOT NULL ,
"ymax" FLOAT NOT NULL, "rc" VARCHAR(255), `angle`  float);
CREATE TRIGGER [INSERT_ITEM_V_HYDRANT]
AFTER INSERT
ON [V_HYDRANT]
FOR EACH ROW
BEGIN
 INSERT INTO V_HYDRANTSPATINDEX
(id,xmin,xmax,ymin,ymax)VALUES(new.rowid,new.xmin,new.xmax,new.ymin,new.ymax)
;
END;

CREATE TRIGGER [DELETE_ITEM_V_HYDRANT]
AFTER DELETE
ON [V_HYDRANT]
FOR EACH ROW
BEGIN
delete from V_HYDRANTSPATINDEX where old.id=V_HYDRANTSPATINDEX.id;
END;

CREATE TRIGGER [UPDATE_ITEM_V_HYDRANT]
AFTER UPDATE OF [geom], [xmin], [ymin], [xmax], [ymax]
ON [V_HYDRANT]
FOR EACH ROW
BEGIN
 delete from V_HYDRANTSPATINDEX where old.id=V_HYDRANTSPATINDEX.id;
 INSERT INTO V_HYDRANTSPATINDEX
(id,xmin,xmax,ymin,ymax)VALUES(new.rowid,new.xmin,new.xmax,new.ymin,new.ymax)
;
END;
```

```
CREATE VIRTUAL TABLE V_HYDRANTSPATINDEX USING rtree(id,xmin,xmax,ymin,ymax);
```

Ukončení přípojky

```
CREATE TABLE "V_UPRIP" ("id" INTEGER PRIMARY KEY  NOT NULL , "geom" BLOB ,
"xmin" FLOAT NOT NULL , "ymin" FLOAT NOT NULL , "xmax" FLOAT NOT NULL ,
"ymax" FLOAT NOT NULL, "rc" VARCHAR(255), `jmeno`  text, `prijmeni`  text,
`mesto`  text, `userdraw`  INTEGER);

CREATE TRIGGER [INSERT_ITEM_V_UPRIP]
AFTER INSERT
ON [V_UPRIP]
FOR EACH ROW
BEGIN
 INSERT INTO V_UPRIPSPATINDEX
(id,xmin,xmax,ymin,ymax)VALUES(new.rowid,new.xmin,new.xmax,new.ymin,new.ymax)
;
END;

CREATE TRIGGER [DELETE_ITEM_V_UPRIP]
AFTER DELETE
ON [V_UPRIP]
FOR EACH ROW
BEGIN
delete from V_UPRIPSPATINDEX where old.id=V_UPRIPSPATINDEX.id;
END;

CREATE TRIGGER [UPDATE_ITEM_V_UPRIP]
AFTER UPDATE OF [geom], [xmin], [ymin], [xmax], [ymax]
ON [V_UPRIP]
FOR EACH ROW
BEGIN
 delete from V_UPRIPSPATINDEX where old.id=V_UPRIPSPATINDEX.id;
 INSERT INTO V_UPRIPSPATINDEX
(id,xmin,xmax,ymin,ymax)VALUES(new.rowid,new.xmin,new.xmax,new.ymin,new.ymax)
;
END;

CREATE VIRTUAL TABLE V_UPRIPSPATINDEX USING rtree(id,xmin,xmax,ymin,ymax);
```

Uzávěr

```
CREATE TABLE "V_UZAVER" ("id" INTEGER PRIMARY KEY  NOT NULL , "geom" BLOB ,
"xmin" FLOAT NOT NULL , "ymin" FLOAT NOT NULL , "xmax" FLOAT NOT NULL ,
"ymax" FLOAT NOT NULL, "rc" VARCHAR(255));

CREATE TRIGGER [INSERT_ITEM_V_UZAVER]
AFTER INSERT
ON [V_UZAVER]
FOR EACH ROW
BEGIN
 INSERT INTO V_UZAVERSPATINDEX
(id,xmin,xmax,ymin,ymax)VALUES(new.rowid,new.xmin,new.xmax,new.ymin,new.ymax)
;
END;

CREATE TRIGGER [DELETE_ITEM_V_UZAVER]
AFTER DELETE
ON [V_UZAVER]
```

```
FOR EACH ROW
BEGIN
delete from V_UZAVERSPATINDEX where old.id=V_UZAVERSPATINDEX.id;
END;

CREATE TRIGGER [UPDATE_ITEM_V_UZAVER]
AFTER UPDATE OF [geom], [xmin], [ymin], [xmax], [ymax]
ON [V_UZAVER]
FOR EACH ROW
BEGIN
 delete from V_UZAVERSPATINDEX where old.id=V_UZAVERSPATINDEX.id;
 INSERT INTO V_UZAVERSPATINDEX
(id,xmin,xmax,ymin,ymax)VALUES(new.rowid,new.xmin,new.xmax,new.ymin,new.ymax)
;
END;

CREATE VIRTUAL TABLE V_UZAVERSPATINDEX USING rtree(id,xmin,xmax,ymin,ymax);
```

Fiktivní uzel

```
CREATE TABLE "V_UZEL" ("id" INTEGER PRIMARY KEY  NOT NULL , "geom" BLOB ,
"xmin" FLOAT NOT NULL , "ymin" FLOAT NOT NULL , "xmax" FLOAT NOT NULL ,
"ymax" FLOAT NOT NULL, "rc" VARCHAR(255));

CREATE TRIGGER [INSERT_ITEM_V_UZEL]
AFTER INSERT
ON [V_UZEL]
FOR EACH ROW
BEGIN
 INSERT INTO V_UZELSPATINDEX
(id,xmin,xmax,ymin,ymax)VALUES(new.rowid,new.xmin,new.xmax,new.ymin,new.ymax)
;
END;

CREATE TRIGGER [DELETE_ITEM_V_UZEL]
AFTER DELETE
ON [V_UZEL]
FOR EACH ROW
BEGIN
delete from V_UZELSPATINDEX where old.id=V_UZELSPATINDEX.id;
END;

CREATE TRIGGER [UPDATE_ITEM_V_UZEL]
AFTER UPDATE OF [geom], [xmin], [ymin], [xmax], [ymax]
ON [V_UZEL]
FOR EACH ROW
BEGIN
 delete from V_UZELSPATINDEX where old.id=V_UZELSPATINDEX.id;
 INSERT INTO V_UZELSPATINDEX
(id,xmin,xmax,ymin,ymax)VALUES(new.rowid,new.xmin,new.xmax,new.ymin,new.ymax)
;
END;

CREATE VIRTUAL TABLE V_UZELSPATINDEX USING rtree(id,xmin,xmax,ymin,ymax);
```

Trasa

```
CREATE TABLE [V_TRASA] (
  [id] INTEGER NOT NULL PRIMARY KEY,
```

```
  [geom] BLOB,
  [xmin] FLOAT NOT NULL,
  [ymin] FLOAT NOT NULL,
  [xmax] FLOAT NOT NULL,
  [ymax] FLOAT NOT NULL,
  [rc] VARCHAR(255),
  [Datum_vystavby] DATE,
  [dim_prip] INTEGER,
  [zakazka] integer,
  [userdraw] INTEGER);

CREATE TRIGGER [INSERT_ITEM_V_TRASA]
AFTER INSERT
ON [V_TRASA]
FOR EACH ROW
BEGIN
 INSERT INTO V_TRASASPATINDEX
(id,xmin,xmax,ymin,ymax)VALUES(new.rowid,new.xmin,new.xmax,new.ymin,new.ymax)
;
END;

CREATE TRIGGER [DELETE_ITEM_V_TRASA]
AFTER DELETE
ON [V_TRASA]
FOR EACH ROW
BEGIN
delete from V_TRASASPATINDEX where old.id=V_TRASASPATINDEX.id;
END;

CREATE TRIGGER [UPDATE_ITEM_V_TRASA]
AFTER UPDATE OF [geom], [xmin], [ymin], [xmax], [ymax]
ON [V_TRASA]
FOR EACH ROW
BEGIN
 delete from V_TRASASPATINDEX where old.id=V_TRASASPATINDEX.id;
 INSERT INTO V_TRASASPATINDEX
(id,xmin,xmax,ymin,ymax)VALUES(new.rowid,new.xmin,new.xmax,new.ymin,new.ymax)
;
END;

CREATE VIRTUAL TABLE V_TRASASPATINDEX USING rtree(id,xmin,xmax,ymin,ymax);
```

3.2.2 Tabulka dokumentů

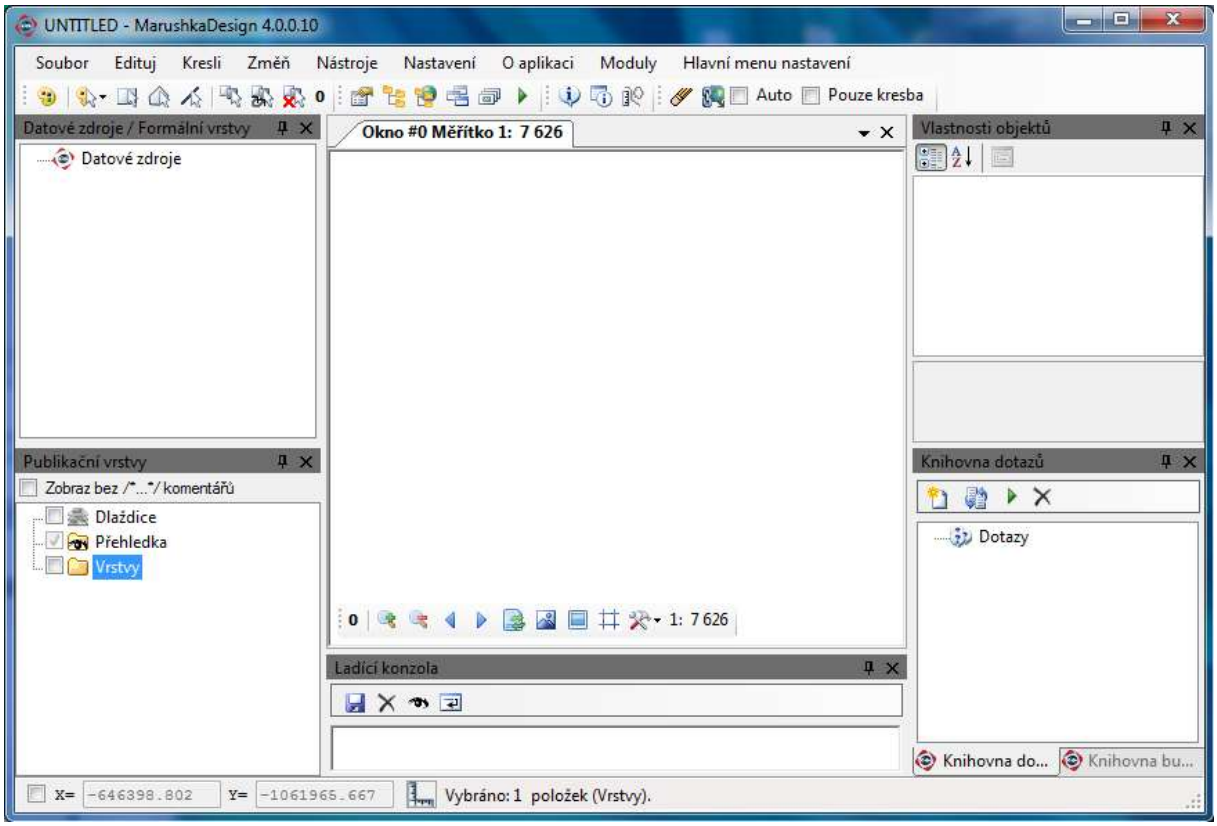
```
CREATE TABLE "demo_doc" ("ID" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
BLOB_TYPE VARCHAR(50),LABEL TEXT,BIRTH_DATE DATETIME,DOC
BLOB,ID_GRAPHICS_ELEMENT INTEGER, `STRING_ID`  varchar(256));

CREATE TRIGGER demo_doc_insert_date AFTER INSERT ON demo_doc
BEGIN
UPDATE demo_doc SET BIRTH_DATE = DATETIME('NOW')  WHERE rowid = new.rowid;
UPDATE demo_doc SET STRING_ID = 'ID'||new.rowid WHERE rowid = new.rowid;
END;
```

Po úspěšném vytvoření všech tabulek jsme připraveni přejít k dalšímu kroku, a to k samotnému importu dat do datového skladu.

4 Učíme Marushku číst a psát

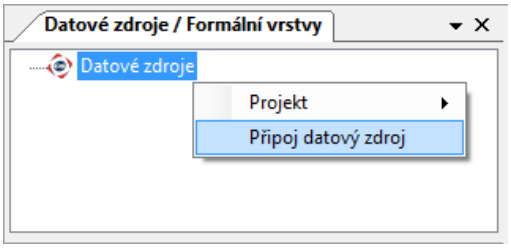
Abychom mohli číst z databázového datového skladu, musíme nejprve data ze zdrojových ESRI shape files zapsat do databáze.

V našem novém projektu vidíme v tuto chvíli pouze prázdné prostředí MarushkaDesignu, které může vypadat takto (záleží na individuální uživatelově konfiguraci, jak si okna uspořádá – více o ovládání a práci v prostředí MarushkaDesignu se můžeme dočíst v manuálu):

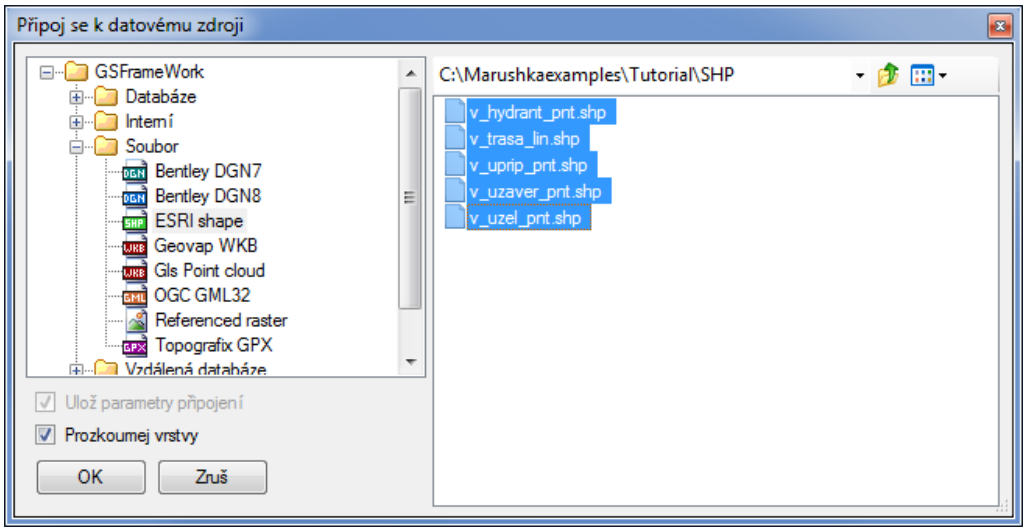


4.1 Připojení datových zdrojů

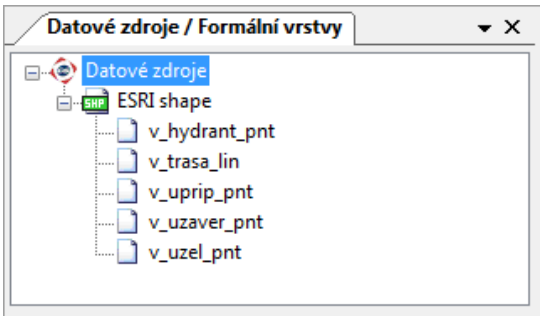
- Připojíme první datový zdroj a tím budou data z ESRI shape files (kontextové menu vyvoláme pravým tlačítkem myši na *Datových zdrojích*):



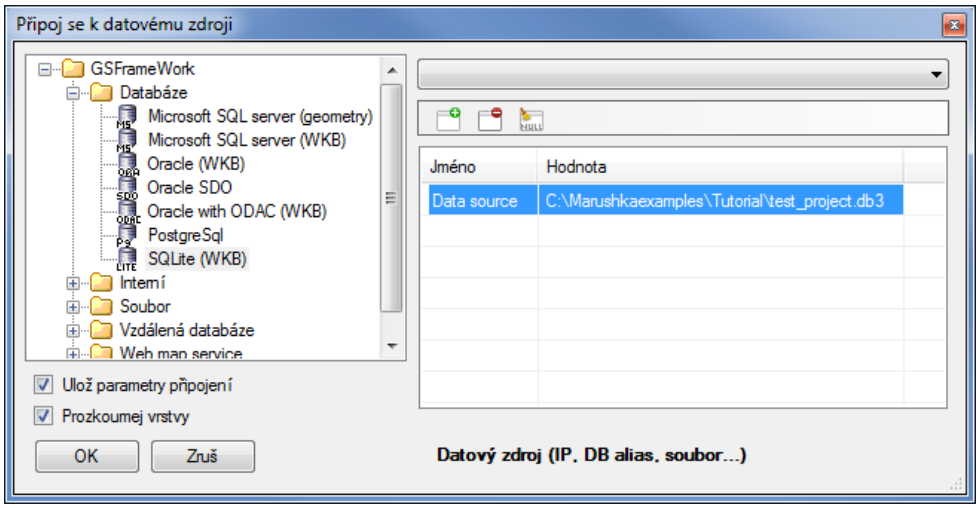
- Objeví se nám dialogové okno umožňující výběr jednotlivých shape filů, které můžeme připojovat jednotlivě nebo hromadně pomocí standardních kláves pro multiselect (CTRL nebo SHIFT + levé tlačítko myši):



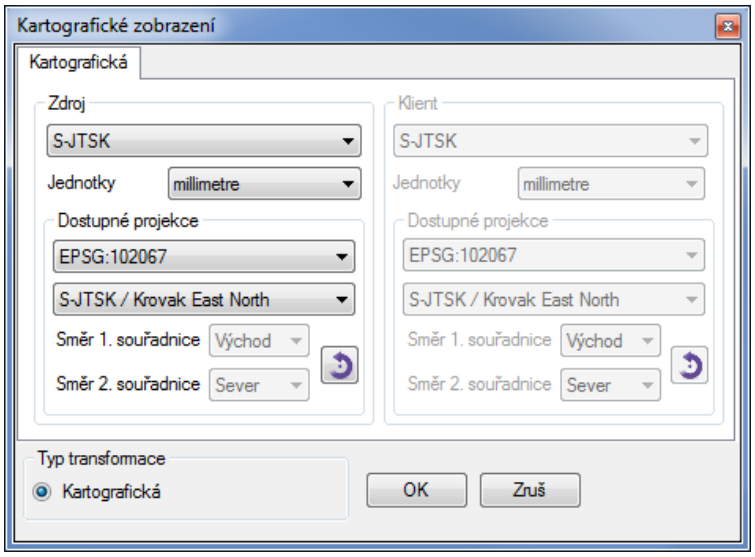
- Po kliknutí na tlačítko OK se změní okno *Datových zdrojů*, ve kterém nám přibyla položka ESRI shape s pěti uzly:



- Takto připravená data potřebujeme v tuto chvíli zapsat do databázového datového skladu. Připojíme si tedy už námi vytvořenou SQLite databázi. Opět přes kontextové menu *Datových zdrojů* přidáme další datový zdroj, tentokrát *Databázi – SQLite (WKB)*. Do pole hodnota doplníme cestu k naší databázi:



- Po potvrzení tlačítkem OK se nám objeví dialogové okno pro zadání zdrojové kartografické projekce. Jako zdrojový souřadný systém zvolíme *S-JTSK* v jednotkách *milimetre*. Projekci ponecháme beze změny: *EPSG: 102067*.

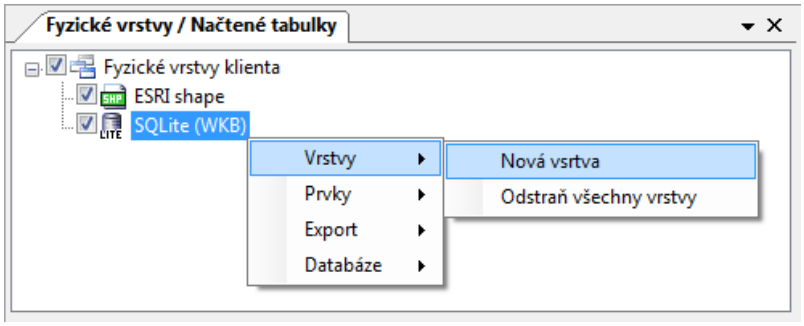


Potvrdíme tlačítkem OK a v *Datových zdrojích* vidíme okamžitě další změnu v tom, že se nám objevila další větev – tentokrát *SQLite (WKB)* opět s pěti uzly.

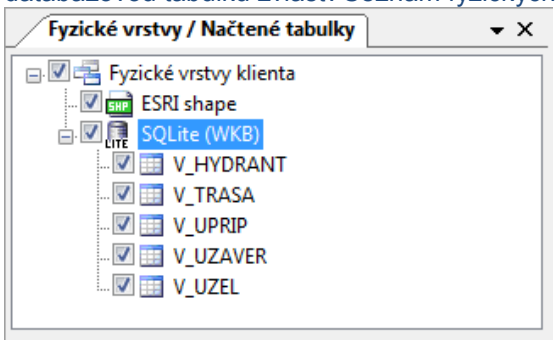
4.2 Import dat z ESRI shape files do SQLite

Pro samotný import do jakéhokoliv jiného datového skladu je nutná existence příslušných fyzických vrstev, do kterých budou zdrojová data směřovat. **Fyzické vrstvy** vytvoříme v cílovém datovém zdroji.

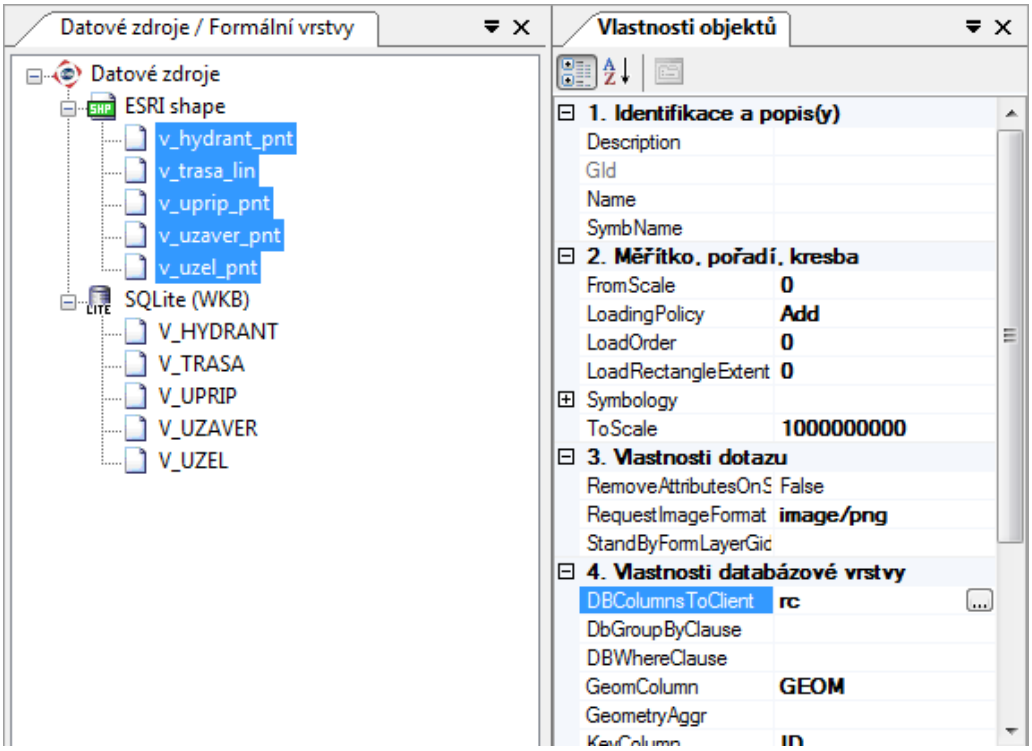
Přepneme se do okna *Fyzických vrstev* a v datovém zdroji *SQLite (WKB)* postupně vytvoříme všech pět fyzických vrstev, které budeme potřebovat. Kontextové menu pro vytvoření nové fyzické vrstvy vyvoláme opět pravým tlačítkem myši na datovém zdroji:



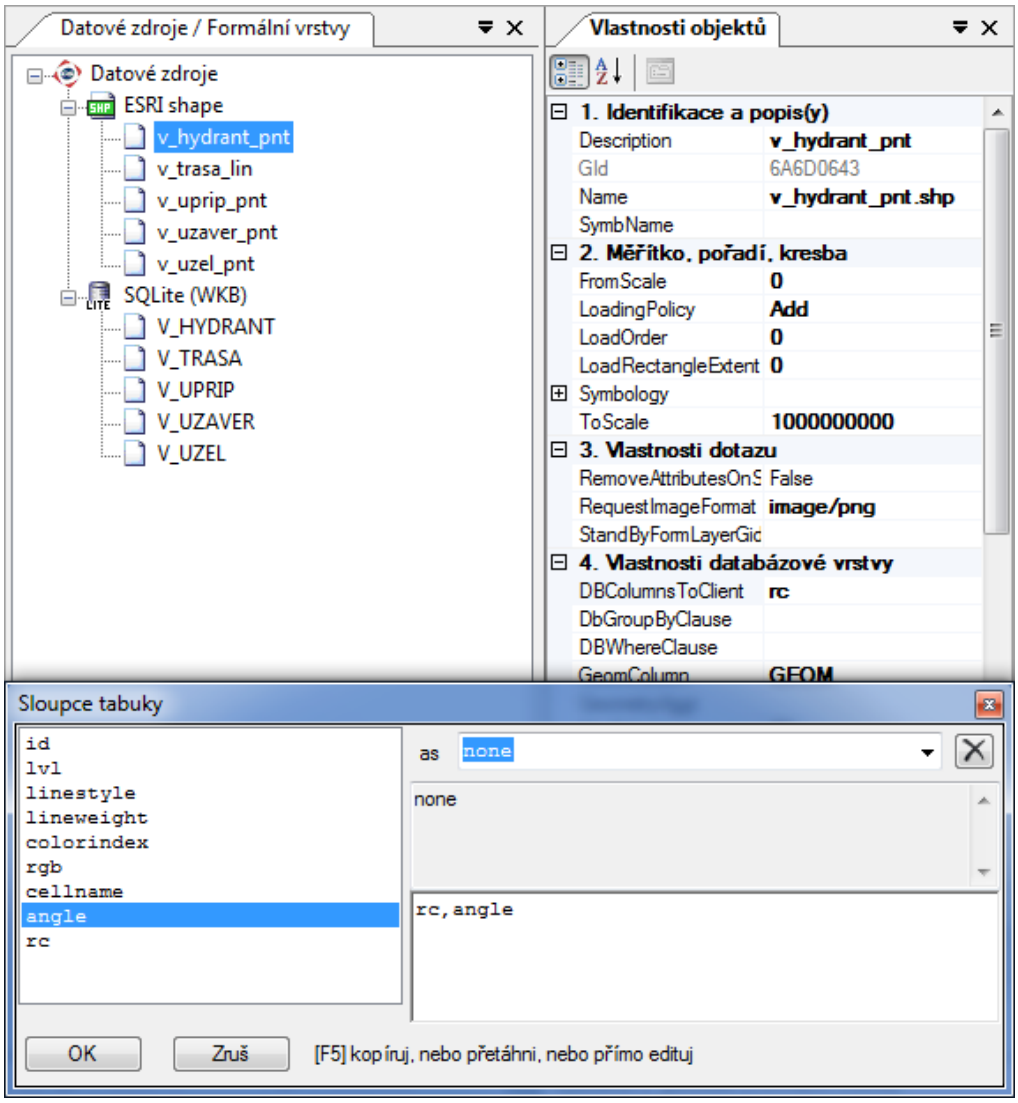
V dialogovém okně pro název vrstvy napíšeme jméno první fyzické vrstvy (název databázové tabulky, do které budou data směřovat). Tento postup opakujeme celkem pětkrát, pro každou databázovou tabulku zvlášť. Seznam fyzických vrstev pak bude vypadat takto:



Do takto vytvořených vrstev už můžou směřovat naše data. Vzhledem k tomu, že data v ESRI shape files obsahují i **atributy**, které budeme chtít využívat, tak je nutné zajistit, aby se i tyto atributy dostaly do databázového datového skladu. Ke všem formálním vrstvám z datového skladu ESRI shape je potřeba doplnit atributy ve vlastnostech každé vrstvy. U všech vrstev budeme vyžadovat atribut *rc*, který tedy můžeme ve vlastnostech vyplnit hromadně:

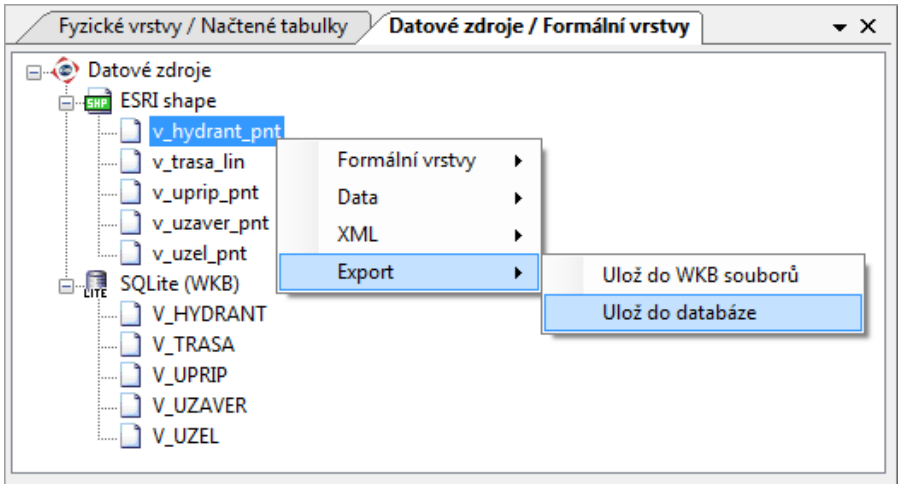


Ze dvou formálních vrstev (v_hydrant_pnt, a v_uprip_pnt) budeme potřebovat ještě další atributy. Změnu provedeme u každé formální vrstvy zvlášť. Ve formální vrstvě v_hydrant_pnt potřebujeme atribut *angle*. Přidání provedeme v dialogovém okně vlastností k dané vrstvě – *DBColumnsToClient*. Klávesou F5 přetáhneme požadovaný atribut:

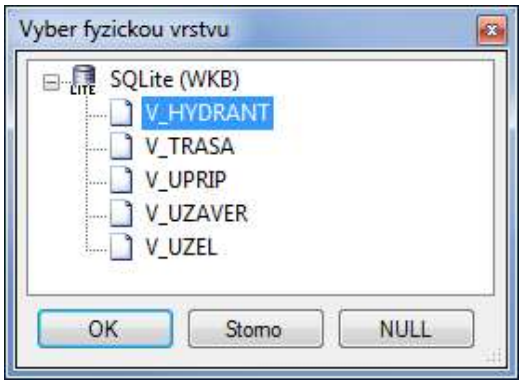


Obdobným způsobem doplníme atributy *jmeno*, *prijmeni* a *mesto* u formální vrstvy *v_uprip_pnt*.

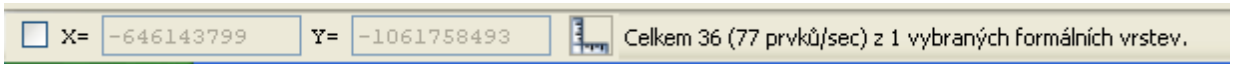
Takto připravená data můžeme konečně zapsat do databázového datového skladu. V jednotlivých formálních vrstvách datového skladu ESRI shape vyvoláme kontextové menu pravým tlačítkem myši.



Nyní potvrdíme volbu *Ulož do databáze*. V následně otevřeném dialogovém okně zvolíme odpovídající fyzickou vrstvu databázového datového skladu:



Po potvrzení tlačítkem OK dostaneme otázku, jestli chceme zachovat původní ID. Vzhledem k tomu, že prvky v ESRI shape files nepocházejí z databáze, tak musíme kliknout na tlačítko: *„Ne“*. Následně proběhne import do databáze, jehož stav můžeme sledovat ve stavovém řádku MarushkaDesignu:

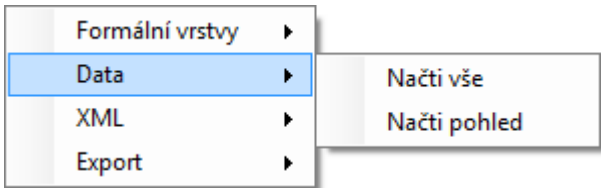


Stejný postup následně aplikujeme pro všechny zbývající formální vrstvy z datového skladu ESRI shape a všechny tyto vrstvy tak zapíšeme do databázového datového skladu SQLite (WKB). Postup tedy opakujeme pro vrstvy: *v_trasa_lin, v_uprip_pnt, v_uzaver_pnt* a *v_uzel_pnt*. Marushku jsme tak tedy naučili „**psát do databáze**.“

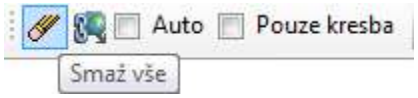
4.3 Zobrazení dat v MarushkaDesignu z databázového datového zdroje

V databázovém datovém skladu SQLite teď máme připravena data k dalšímu použití. Ze zdroje ESRI shape jsme nepřebrali grafické atributy a tak se naučíme změnit grafickou symbologii dat a **naučíme Marushku kreslit** tak, jak nám to bude vyhovovat. Datový zdroj ESRI shape v tuto chvíli definitivně opustíme a dále budeme pracovat pouze s databázovým datovým zdrojem SQLite. Od této chvíle budeme pracovat především s následujícími ovládacími prvky.

- Kontextové menu formální vrstvy – *Data* – *Načti vše,* příp. *Načti pohled* (umožní nám načíst data formální vrstvy po změnách, které jsme provedli v projektu):



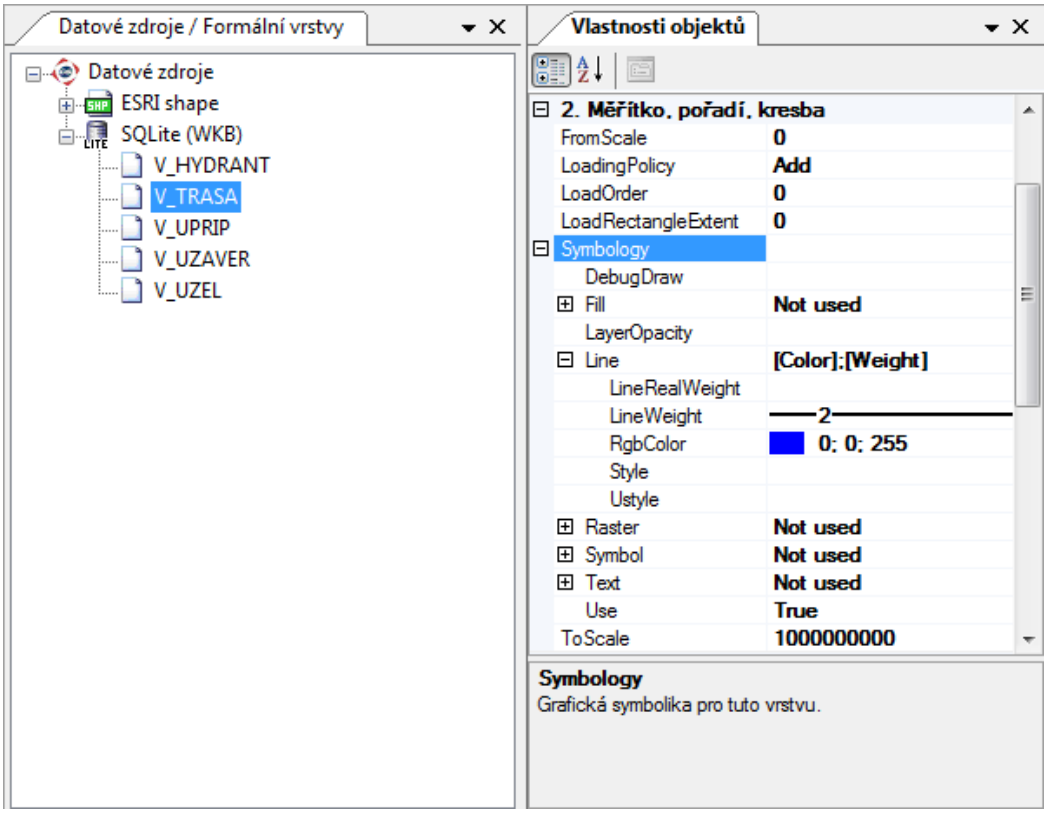
- Ikona *Smaž vše* z panelu nástrojů *Načti data* (z výkresu odstraní všechny fyzické vrstvy – data, která byla načtena – tuto ikonu je nezbytné aktivovat po každé změně formální vrstvy před stažením aktuálních dat, jinak by se v mapovém okně změna nijak neprojevila – platí, že zobrazeny budou pouze prvky, které ještě nejsou součástí výkresu v mapovém okně):



- Ikona *Vše* z menu mapového okna slouží pro zobrazení všech prvků ve fyzických vrstvách dostupných v daném okamžiku ve výkresu mapového okna:

4.3.1 Zobrazení liniových elementů

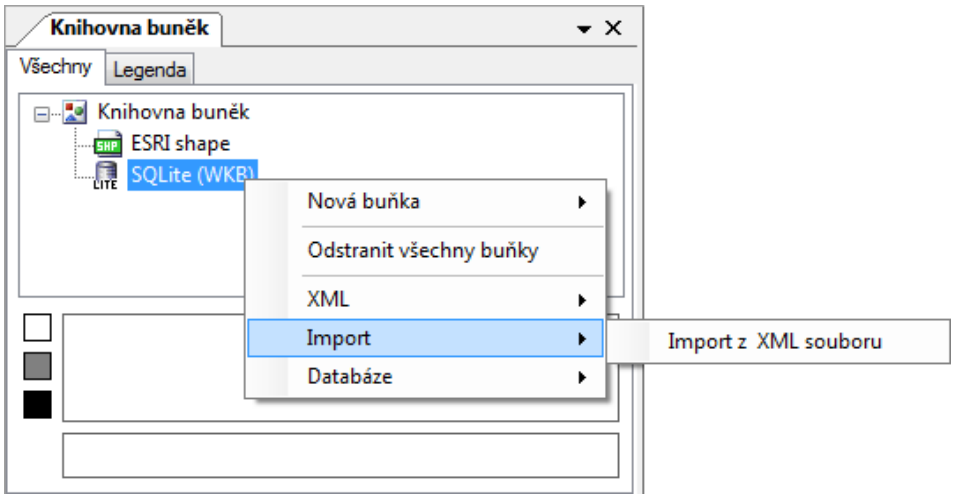
Dvojici příkazů *Smaž vše* a *Načti vše*, příp. *Načti pohled* si brzy osvojíme a budeme je provádět prakticky intuitivně po každé změně ve formální vrstvě a pro testování funkčnosti nové změny. Vyzkoušíme si to na příkladu. Smažeme vše a načteme všechna data z formální vrstvy *V_TRASA*. Ve výsledku vidíme všechny prvky trasy v šedivém provedení, které se nám nelíbí. Vodovodní síť budeme chtít zobrazit modrou barvou, tloušťky dvě. Změníme tedy symbologii ve vlastnosti formální vrstvy *V_TRASA Symbology* zobrazované vrstvy:



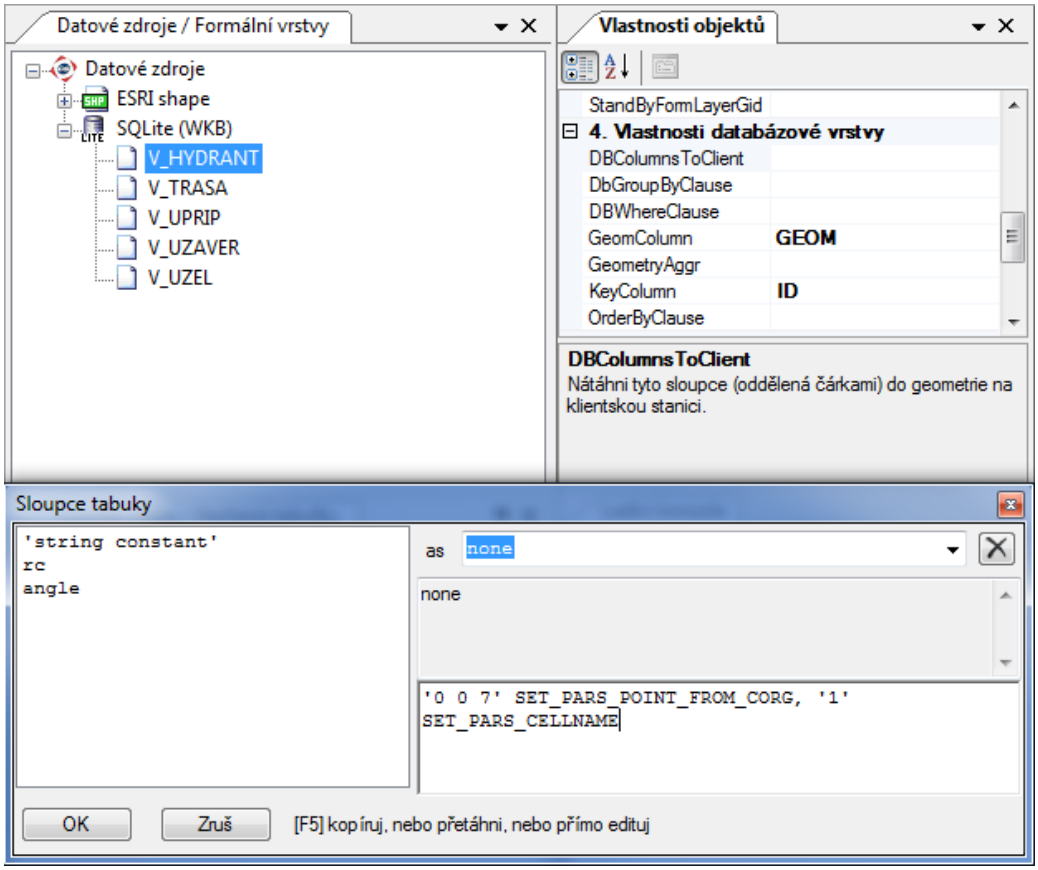
Výsledek této změny si můžete okamžitě prohlédnout už známou kombinací dvou příkazů ikony *Smaž vše* a volbou *Načti vše* z kontextového menu formální vrstvy *V_TRASA*.

Už jsme na projektu udělali poměrně dost práce, tak by bylo na místě připomenutí o průběžném ukládání projektu. Je vhodné zvyknout si na něj a používat např. klávesovou zkratku CTRL + S pro průběžné ukládání projektu. K automatickému uložení dochází pouze v případě spouštění lokálního WEB serveru, o kterém se více dozvíme později.

Ukázali jsme si, jak jednoduše změnit symbologii liniových objektů. Co se týče symbologie buněk, tak v tomto případě už to tak jednoduché není. Základ tvoří připojená knihovna buněk, jejíž prvky budeme využívat pro zobrazení jednotlivých elementů. Otevřeme si Knihovnu buněk (z menu *Nástroje – Knihovny – Knihovna buněk*). Objeví se nám dialogové okno, ve kterém můžeme připojit knihovnu buněk ze souboru `Bunky.xml`, který je součástí tohoto tutoriálu. Kontextové menu vyvoláme pravým tlačítkem myši a v dialogovém okně pro výběr souboru vybereme soubor v cestě tutoriálu (`c:\MarushkaExamples\Tutorial\Soubory\Bunky.xml`).



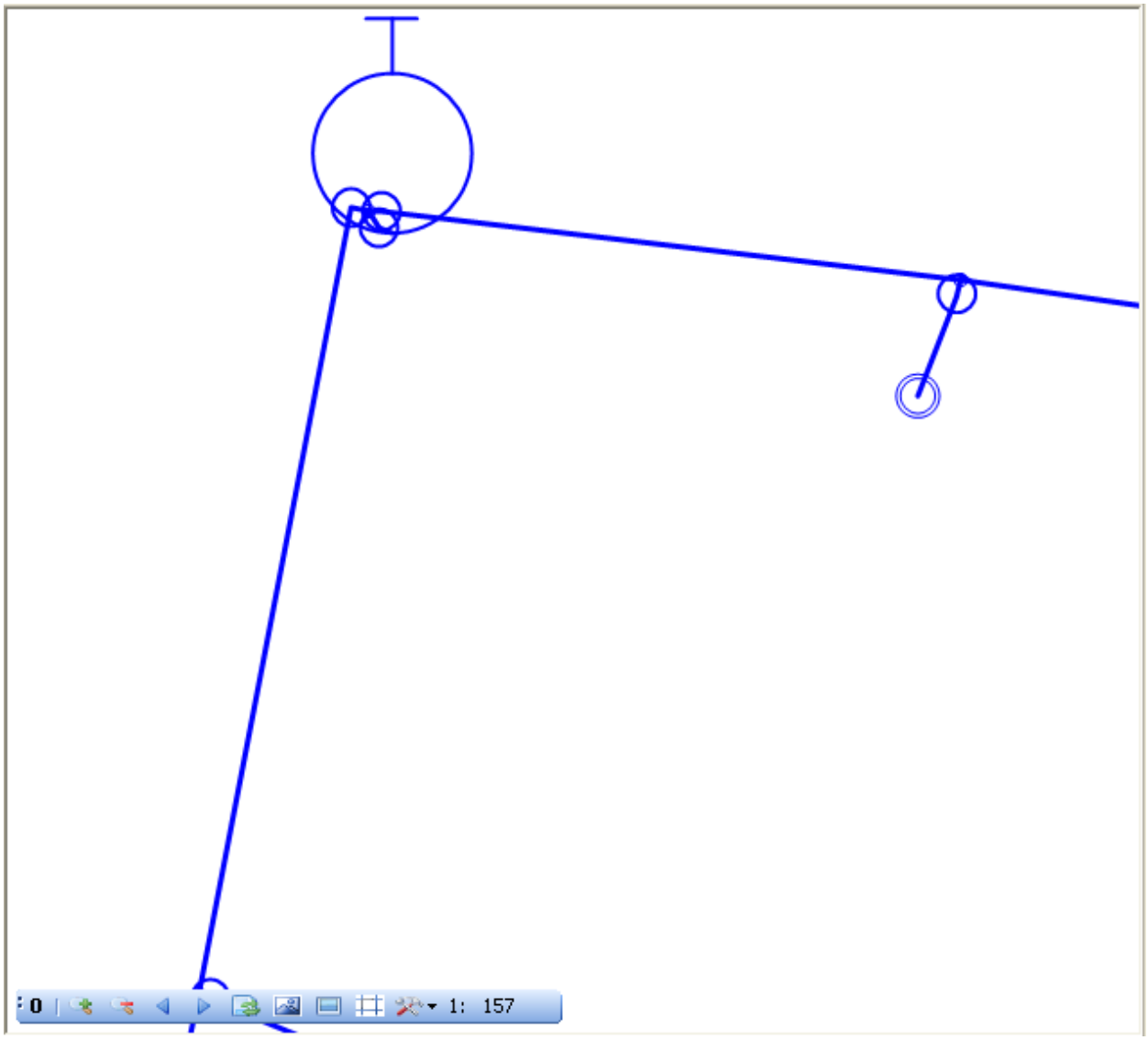
S připojenou knihovnou buněk můžeme zobrazit potřebné buňky. Vzhledem k tomu, že ESRI shape soubory neznají prvek typu buňka, jsou v tuto chvíli všechny prvky v databázovém datovém skladu typu bod. Tyto body teď budeme zobrazovat jako konkrétní buňky. Ve vlastnostech formálních vrstev nás teď bude zajímat vlastnost *DBColumnsToClient*. Jde o vlastnost, kde z databázové entity načítáme i hodnoty ve sloupcích dané databázové tabulky. Kromě toho jsou v prostředí MarushkaDesignu definovány tzv. pseudosloupce, kterých teď využijeme. Více se o pseudosloupcích `SET_PARS_` můžeme dočíst v manuálu pro MarushkaDesign. Pro zobrazení buňky z bodového objektu využijeme pseudosloupce `SET_PARS_POINT_FROM_CORG`, který nám blíže definuje bod, který bude sloužit jako vztažný bod pro příslušnou buňku. Zapsání tohoto sloupce můžeme provést ručně nebo pomocí výběru pseudosloupce, který je součástí tohoto dialogového okna. Hodnota v tomto pseudosloupci bude ‚0 0 7'. Dále je potřeba definovat název buňky, která se bude pro prvky z této databázové tabulky používat. Využijeme druhý pseudosloupec, tentokrát `SET_PARS_CELLNAME` a do jeho hodnoty vepíšeme název potřebné buňky z naší knihovny buněk. Pro tabulku *V_HYDRANT* bude zápis těchto dvou pseudosloupců vypadat takto:



Samostatné cvičení:

Pro zobrazení hydrantu jsme použili buňku *V_HN* z knihovny buněk, která má ve své vlastnosti `Cellname` vyplněnu hodnotu ‘1‘ a tato hodnota je řídící hodnotou pro databázové vyhodnocování jména buňky. Samostatně a bez podrobného návodu teď budeme postup opakovat pro všechny tři zbylé formální vrstvy s tím, že k vrstvě *V_UPRIP* přiřadíme buňku *V_UPRI*, k vrstvě *V_UZAVER* buňku *V_UZAV* a k vrstvě *V_UZEL* buňku *V_UZEL*.

Správnost našeho cvičení si můžeme okamžitě ověřit stažením dat do mapového okna MarushkaDesignu. Provedeme akci *Smaž vše* a následně do mapového okna načteme data z jednotlivých formálních vrstev. Postupně tedy načteme vrstvy *V_HYDRANT*, *V_TRASA*, *V_UPRIP*, *V_UZAVER* a *V_UZEL* a výsledek našeho snažení můžeme vidět například v této podobě (výřez je z levé horní části mapového okna):



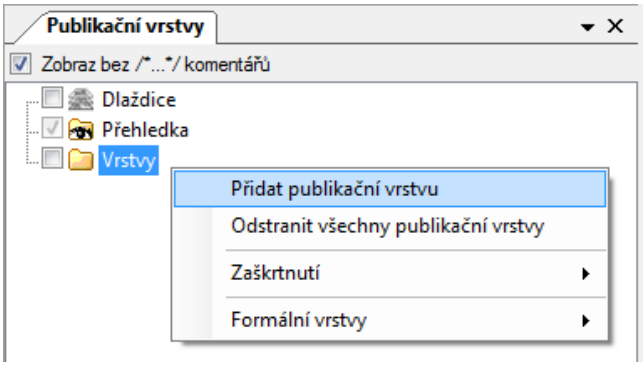
Na obrázku vidíme všechny elementy vodovodní sítě – trasu řadu, přípojek, fiktivních uzlů, uzávěrů, hydrantů a ukončení přípojek.

5 Zobrazení dat v lokálním WEB serveru

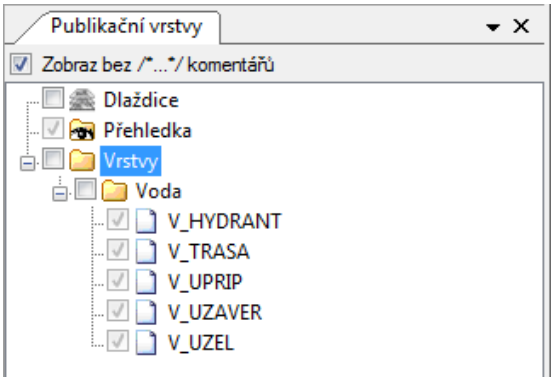
Naše mapová kompozice je v tuto chvíli sice ještě poněkud statická, a ne ještě úplně podle našich představ, ale i přesto už stojí za to ukázat si, jakým způsobem si výsledek zobrazení budeme moct prohlédnout v **lokálním WEB serveru**. Tak vytvořené prostředí už budeme moct přenášet na jakýkoliv Marushka server, a tak bude vypadat výsledná mapová publikace ve webovém prohlížeči.

5.1 Vytvoření publikační vrstvy

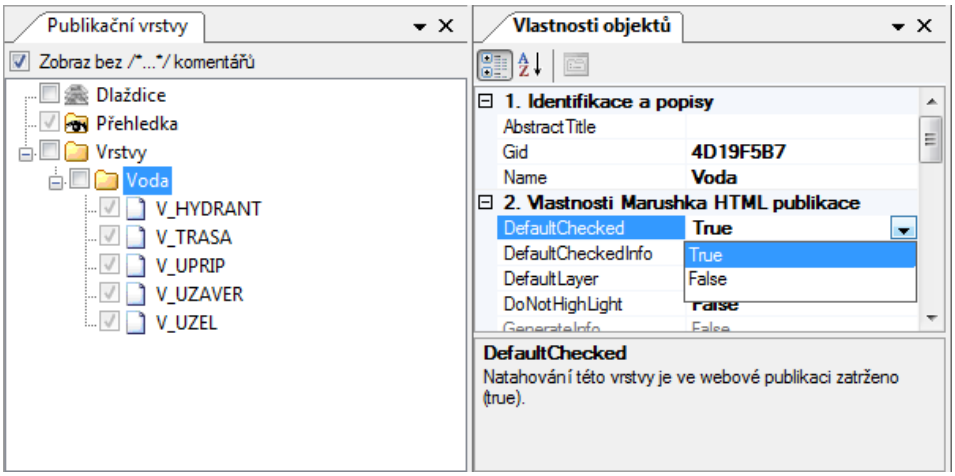
Prvním krokem k zobrazení našich dat ve webovém prostředí je vytvoření publikační vrstvy. Otevřeme si okno publikačních vrstev (pokud ještě nemáme) z panelu nástrojů *Vrstvy a grafická data*. Pro vytvoření nové publikační vrstvy si vyvoláme klikem pravého tlačítka kontextové menu na položce *Vrstvy* a zvolíme možnost *Přidat publikační vrstvu*. Pro jednoduchost vrstvu pojmenujeme ***Voda***:



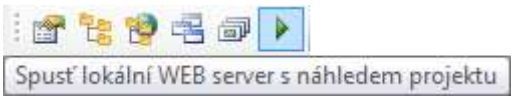
Publikační vrstva je vrstva, která se bude zobrazovat ve webové publikaci a může obsahovat jednu nebo více formálních vrstev. U jednotlivých formálních vrstev můžeme definovat jejich měřítkové rozsahy, které nám umožní v rámci jedné publikační vrstvy zobrazovat různá data v různých měřítcích (např. je zbytečné zobrazovat buňky fiktivních uzlů v měřítcích menších než 1: 1 000 apod.). Do publikační vrstvy Voda přesuneme postupně (nebo pomocí multiselectu) všechny formální vrstvy z databázového datového skladu SQLite a tím budeme mít zajištěno, že v jedné Publikační vrstvě Voda uvidíme všechny prvky této kategorie. Výsledek potom bude vypadat takto:



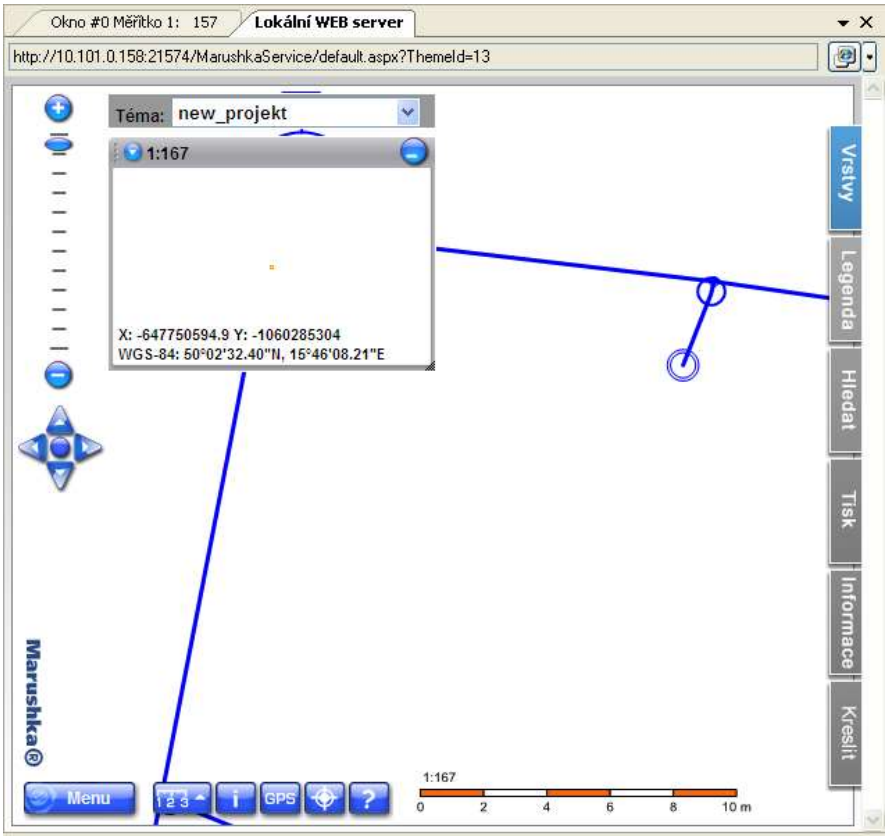
Abychom po startu lokálního WEB serveru rovnou viděli výsledek, tak ve vlastnostech Publikační vrstvy *Voda* změníme vlastnost *DefaultChecked* z hodnoty *false* na *true*. To nám zajistí, že po startu lokálního WEB serveru bude tato vrstva zobrazena:



Projekt si můžeme uložit (tentokrát ale bude uložen automaticky) a konečně spustíme lokální WEB server z panelu nástrojů *Vrstvy a grafická data*:



Po chvíli čekání se nám objeví v pozici původního mapového okna záložka s oknem interního lokálního webového prohlížeče v pozici výřezu z původního mapového okna:



Mapová kompozice je v tomto okamžiku ještě dost strohá a statická, ale už v tuto chvíli můžeme vidět naše data ve tvaru, v jakém se můžou zobrazovat kdekoliv na webu, pokud budeme mít správně konfigurovaný webový server. Konfigurace webových serverů je ale samostatnou kapitolou, která je nad rámec našeho tutoriálu. Dále se však budeme věnovat pouze práci v tomto lokálním prostředí. Naši publikační vrstvu můžeme vidět v záložce *Vrstvy*, kde si ji můžeme vypnout a zase zapnout. To je zatím jediná funkčnost našeho vytvořeného projektu.

6 Změna symbologie zobrazovaných dat – dokončení

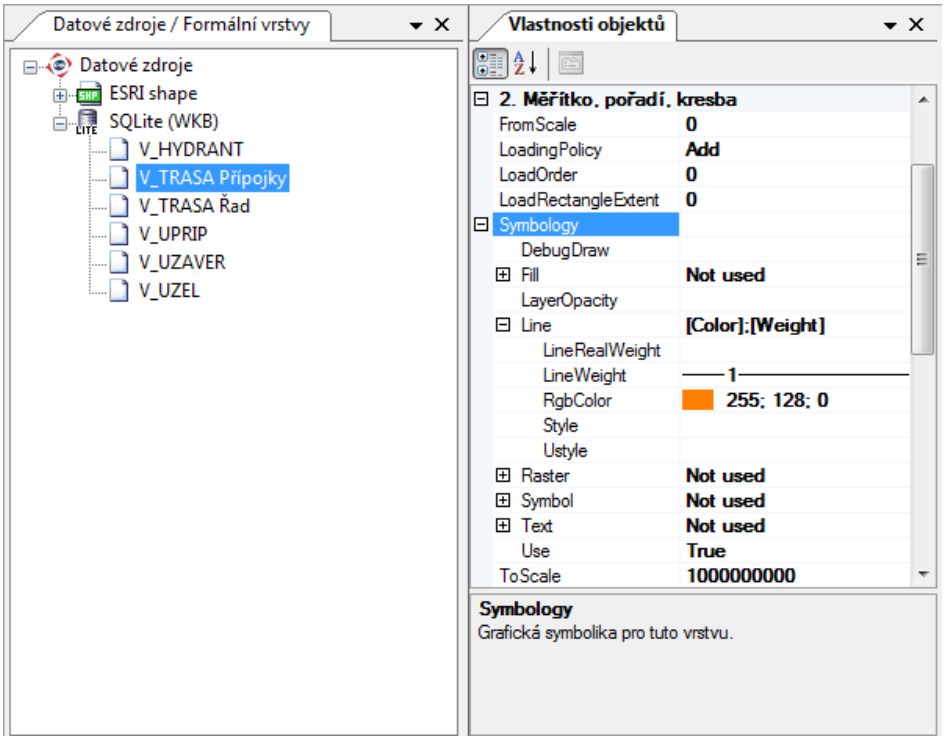
Představme si teď situaci, že přijde někdo a řekne, že se mu v publikaci **nelíbí buňky uzávěrů**, které nezpřehledňují webovou publikaci a chce mít **oranžové přípojky** a **světle modré ukončení přípojek**. Ještě před tím, než přejdeme k řešení těchto požadavků, jsme si sami všimli buněk hydrantů, které bychom chtěli vidět pod úhlem, ve kterém byly v původním datovém skladu.

Ještě před tím bychom chtěli změnit úhel otočení hydrantů. V databázi máme uloženu informaci o úhlu ve sloupci *angle*. Této informace využijeme a hodnotu v něm uloženou přidáme do pseudosloupců formální vrstvy *V_HYDRANT*. Nativní úhlovou jednotkou jsou ale radiány a tak je nutná typová konverze říkající, že hodnota uložená ve sloupci *angle* je ve stupních. Řetězec uložený ve vlastnosti *DBColumnsToClient* u formální vrstvy *V_HYDRANT* bude vypadat takto: „`'0 0 7' SET_PARS_POINT_FROM_CORG, '1' SET_PARS_CELLNAME, '(DEG)'||angle SET_PARS_ROTANGLE`“. Změnu si můžeme okamžitě prohlédnout opět stažením nových dat.

Další změny si ukážeme v pořadí podle složitosti. Začneme tedy odstraněním buněk uzávěrů. Dosáhneme toho tak, že v publikační vrstvě Voda odebereme (pravým tlačítkem myši) vrstvu *V_UZAVER*. Od této chvíle se v lokálním webovém serveru už tato vrstva zobrazovat nebude.

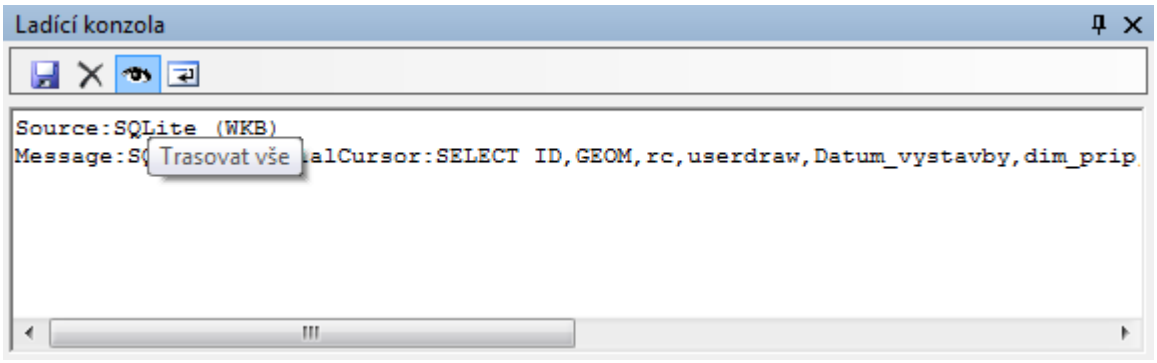
Přebarvení ukončení přípojky je také jednoduchým krokem a opět využijeme další z pseudosloupců, se kterými Marushka umí pracovat, tentokrát to bude `SET_PARS_REPLACE_COLOR`, který dokáže nahradit jednu barvu za jinou. Do vlastnosti *DBColumnsToClient* formální vrstvy *V_UPRIP* doplníme řetězec: `'C 255 0 0 255 255 0 128 192' SET_PARS_REPLACE_COLOR`, který nám zajistí náhradu původní modré barvy za novou světle modrou.

Posledním požadavkem bylo přebarvení přípojek na oranžovou barvu. Spojnice řadů a přípojek se nachází v jedné formální vrstvě, ale můžeme využít databázového atributu RC a tyto dvě skupiny (řady a přípojky) budeme zobrazovat zvlášť. Nejprve si **zklonujeme** formální vrstvu (kontextové menu pravým tlačítkem myši na formální vrstvě *V_TRASA – Formální vrstva – Klonuj*). Výsledkem bude nová formální vrstva s vlastností `SymbName=copy`. Pro lepší orientaci vlastnost SymbName u nové vrstvy změníme na „Přípojky“ a u původní vrstvy V_TRASA vyplníme vlastnost SymbName na „Řad“. V obou vrstvách teď musíme definovat databázovou podmínku tak, aby se nám zobrazovala pouze data řadů v jedné vrstvě a pouze data přípojek ve vrstvě druhé. V klonu pro Řady vyplníme vlastnost *DBWhereClause* takto: „`rc='Trasa vodovodního řadu'`“. V klonu pro Přípojky vyplníme tuto vlastnost takto: „`rc='Trasa vodovodní přípojky'`“. Teď už můžeme změnit symbologii tohoto klonu formální vrstvy dle definovaných požadavků.



6.1 Ladící konzola

V předchozím kroku jsme využili databázové podmínky a pro ověření správnosti databázových příkazů nám nejednou může posloužit **Ladící konzola**, která umožňuje „zachytávání“ veškerých příkazů, které probíhají na pozadí MarushkaDesignu. Ladící konzolu najdeme v menu *Nástroje – Ladící konzola*. Její základ tvoří dialogové okno, ve kterém se nám můžou zobrazovat průběžně výsledky jednotlivých dotazů. Zapnutí Ladící konzoly je vhodné pouze ve fázi ladění dotazů, protože jinak zbytečně snižuje výkon a rychlost MarushkaDesignu. Toto zapnutí se provádí ikonou zvýrazněnou v následujícím obrázku:



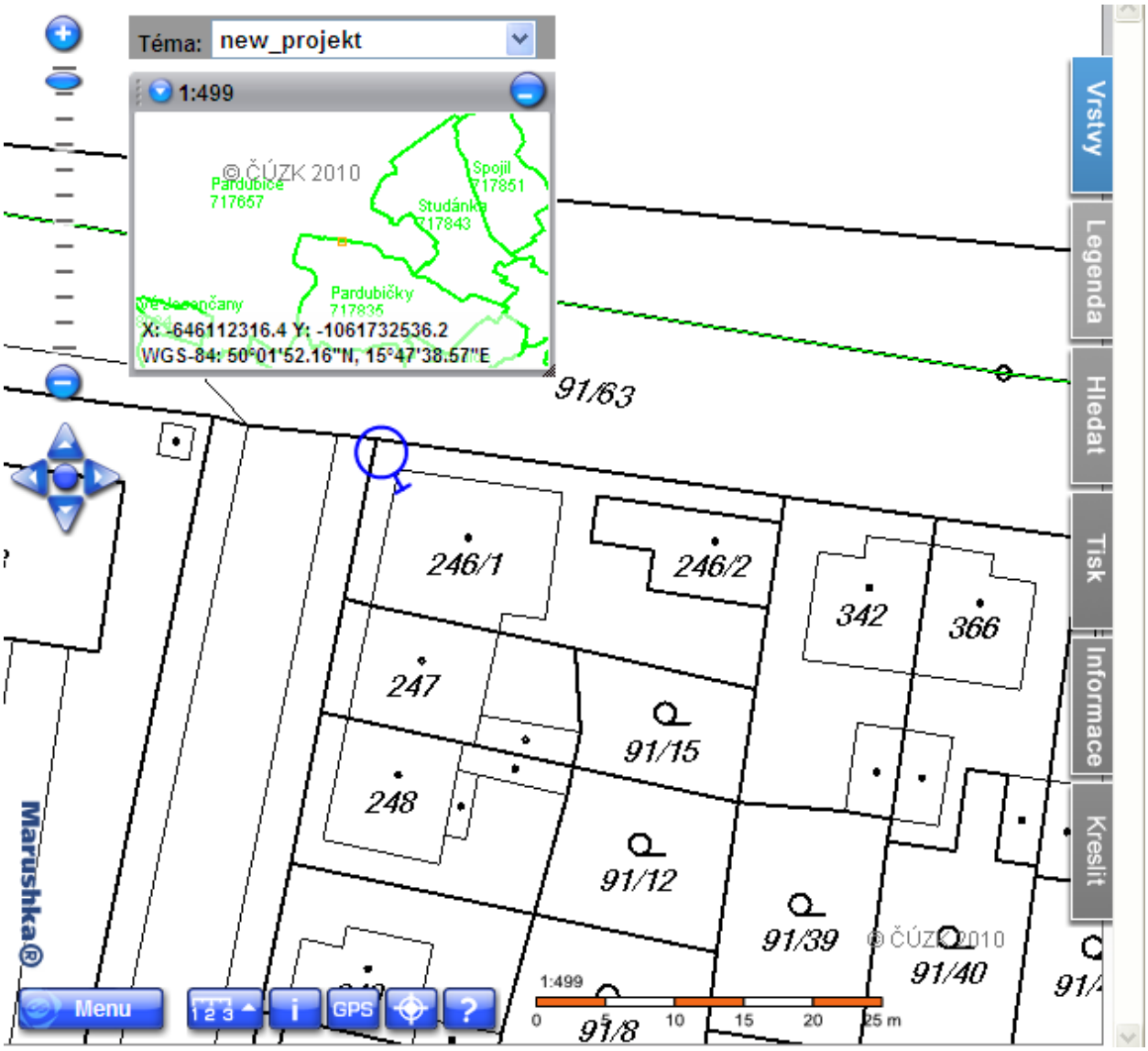
Výsledkem může být například odchycený databázový select, jehož správnost si můžeme následně vyzkoušet v SQL manageru a díky tomu pak můžeme odhalit chyby typu překlepů apod. Ladící konzola může být tedy často vhodným pomocníkem při jakýchkoliv potížích se zobrazením požadovaných dat z různých (nejenom databázových) datových zdrojů.

6.2 Poslední úpravy v základním zobrazení dat

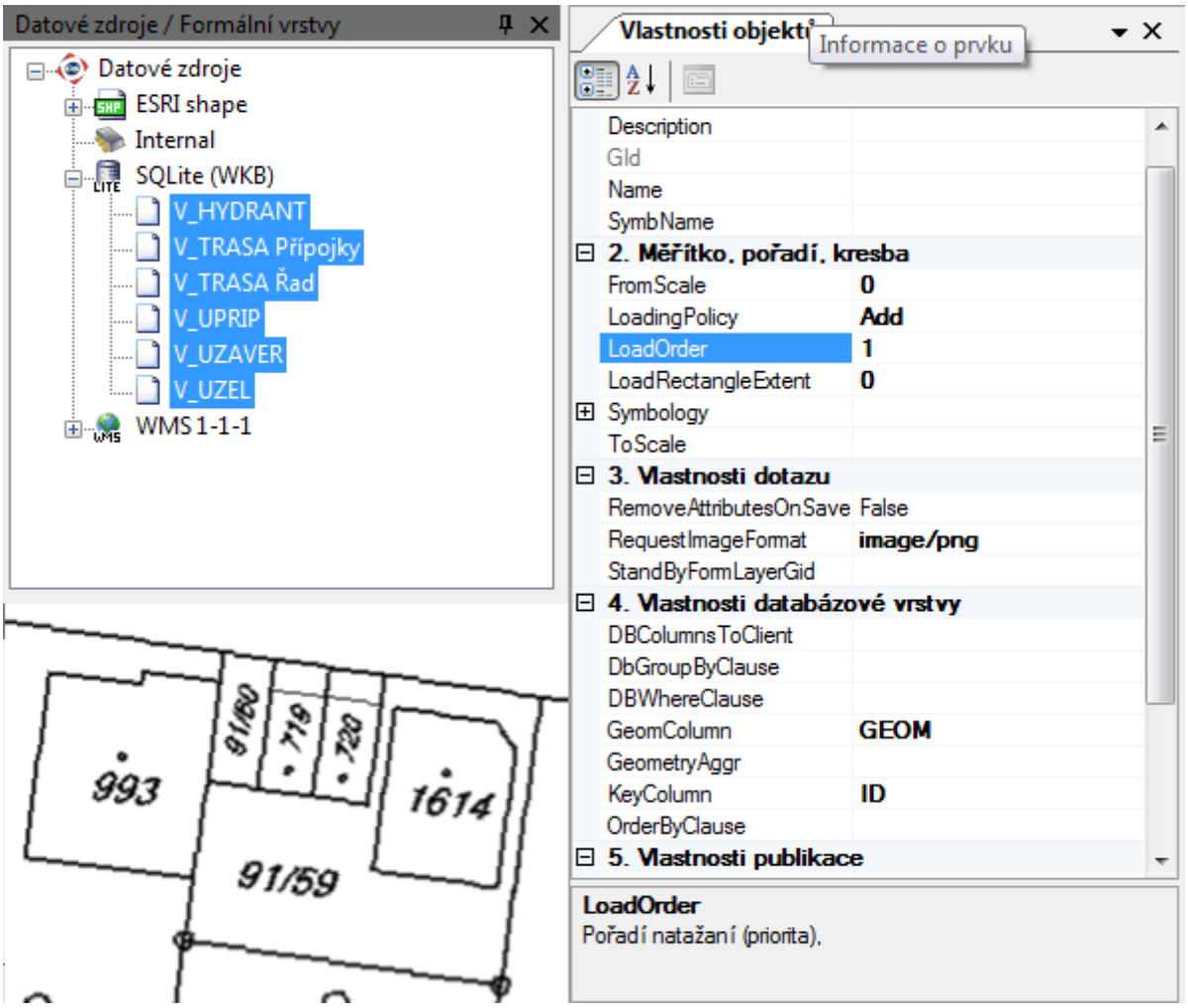
Poslední drobností, která nám ještě schází v publikaci našeho projektu, je zobrazení mapy, která by nám přiblížila průběh vodovodní sítě. K tomuto účelu využijeme reálnou veřejně dostupnou a bezplatnou wms službu Českého úřadu zeměměřičského a katastrálního. Do našeho projektu ji připojíme jako dílčí předpřipravený subprojekt.

V datových zdrojích v kontextovém menu zvolíme *Projekt – Přidej projekt* a ze složky `MarushkaExamples\Tutorial\Soubory\` vybereme soubor `NewWMS.xml`. Můžeme použít průvodce, který nám ukáže přehled datových zdrojů, formálních a publikačních vrstev, které jsou k dispozici v tomto projektu. V našem případě nebudeme provádět žádné změny, ale v jiných případech si můžeme sami zvolit, které části z tohoto subprojektu vybereme pro náš hlavní projekt. Po úspěšném importu do našeho projektu přibude datový zdroj WMS s jednou formální vrstvou, která obsahuje veškeré dílčí vrstvy potřebné pro zobrazení celé katastrální mapy. Rovněž nám do publikačních vrstev přibyla jedna publikační vrstva, které v jejich vlastnostech nastavíme *DefaultChecked* na *true*, aby se nám katastrální mapa zobrazovala při spuštění WEB serveru.

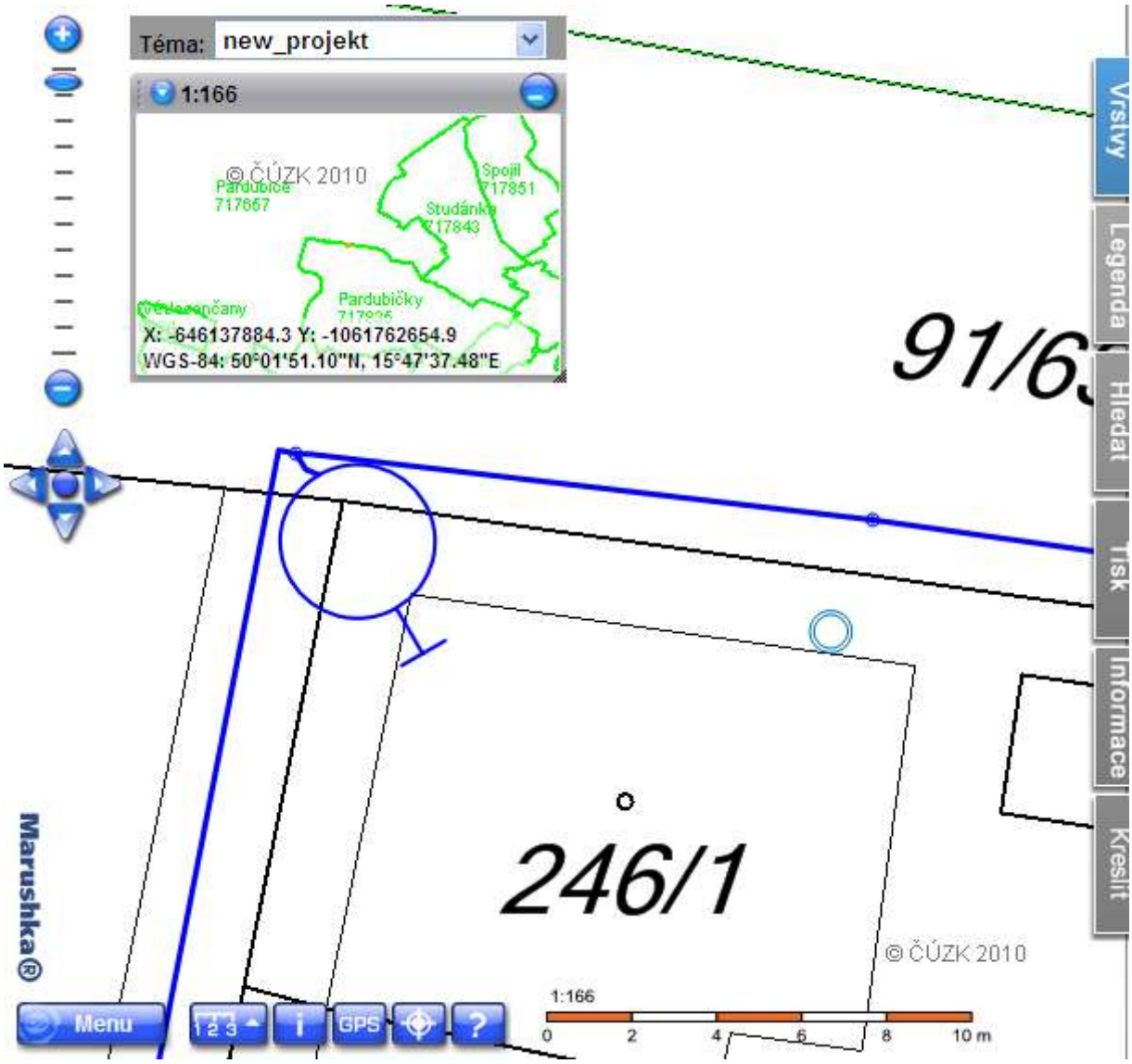
Formální vrstvu *CelaKN* z této WMS služby přetáhneme ještě do kategorie *Přehledka* ve formálních vrstvách. V tomto okamžiku budou data poskytovaná WMS službou zobrazována i v malé přehledce objevující se v levém horním rohu WEB serveru. Výsledek našeho snažení si můžeme opět prohlédnout v lokálním WEB serveru. Pokud jsme postupovali přesně podle pokynů, tak dostaneme pravděpodobně takový výsledek, který se podobá stavu na dalším obrázku:



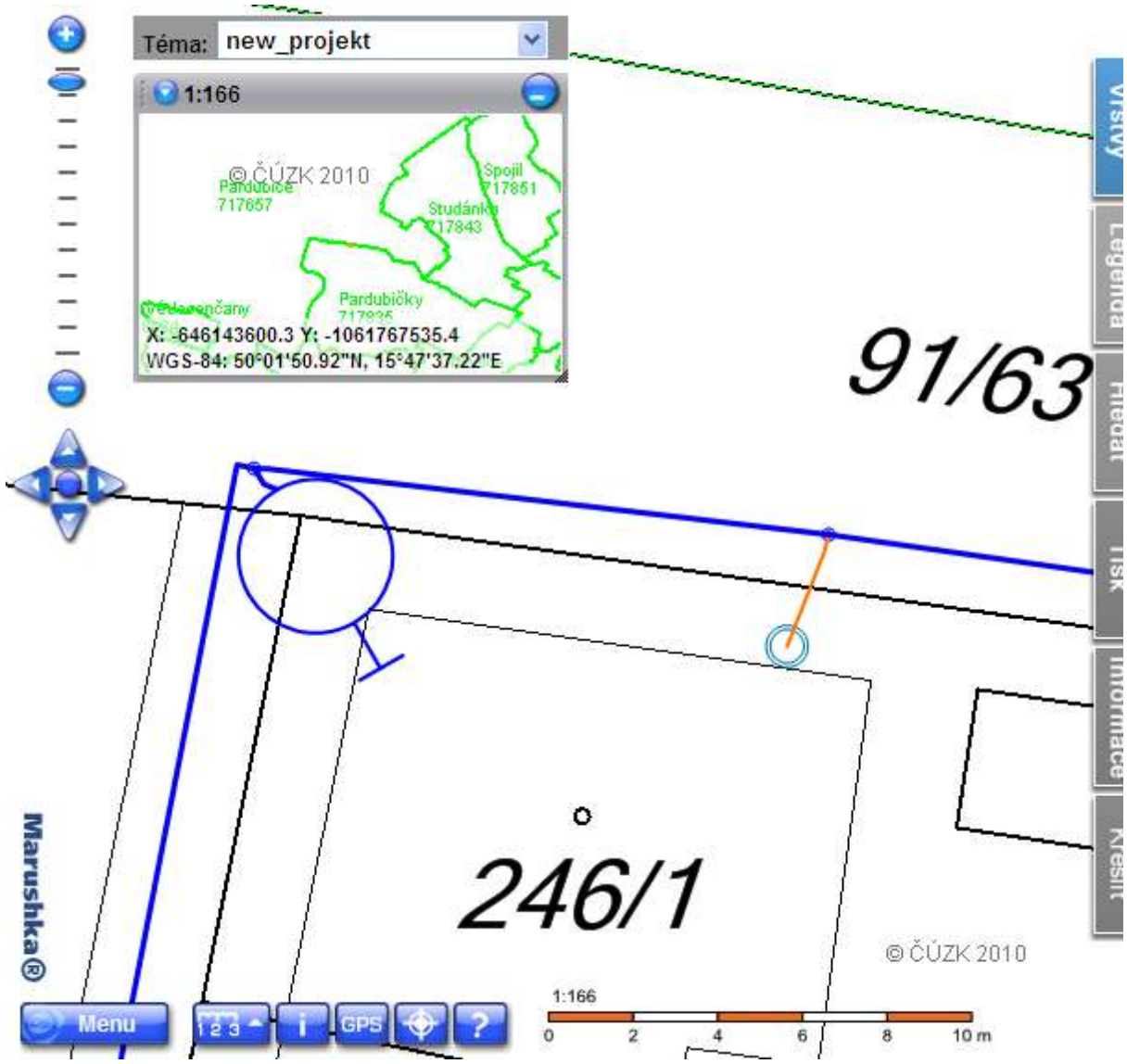
K našemu překvapení jsme zjistili, že se nám ztratila naše vodovodní síť. Jaká je příčina? Na obrázku zůstala viditelná pouze buňka hydrantu a i to je jev pouze náhodný. V prostředí MarushkaDesignu si totiž definujeme i pořadí vrstev, ve kterém budou vykreslovány. Tuto hodnotu jsme nijak neměnili a tak mají všechny vrstvy nastavenu hodnotu „0“ (ve vlastnostech formálních vrstev je to vlastnost *LoadOrder*) a všechny naše vrstvy se tak zobrazují přes sebe v náhodném pořadí. Tuto hodnotu musíme teď upravit kvůli korektnímu zobrazování našich dat. V našem případě bude dostatečným řešením, pokud všem formálním vrstvám z datového zdroje SQLite nastavíme hodnotu vlastnosti `LoadOrder=1`. Ve složitějších mapových projektech je k dispozici *Průzkumník mapové kompozice* v menu *Nástroje*, který umožňuje přehlednou správu jednak pořadí a také měřítkových rozsahů zobrazovaných vrstev.



Po této změně znovu spustíme lokální WEB server:



Výsledek našeho snažení je teď jasně patrný a už vidíme naši vodovodní síť až na jednu drobnost. V publikaci na WEB serveru teď nevidíme spojnice přípojek. Jak k tomu došlo? Při klonování formálních vrstev se nový klon automaticky nepřidává i do publikačních vrstev a nová formální vrstva přípojek tak nemůže být zobrazena. Vrátíme se tedy zpátky do našeho projektu a ze seznamu formálních vrstev „přetáhneme“ formální vrstvu *V_TRASA Přípojky* do publikační vrstvy *Voda*. Dalším opětovným spuštěním lokálního WEB serveru konečně získáváme korektní data ve tvaru, ve kterém jsme je chtěli vidět:



Živým testováním funkčnosti projektu jsme tedy odhalili několik nedostatků a chyb, kterých jsme se při konfiguraci projektu dopustili a vidíme, že pomocné mapové okno (přehledka) i okno lokálního WEB serveru, jsou při naší práci nepostradatelnými pomocníky. V tuto chvíli jsme již prozkoumali základní funkce Marushky pro jednoduché zobrazení našich vektorových dat v kombinaci s rastrovým podkladem ze zdroje WMS. Možnosti jsou prakticky neomezené – můžeme mít libovolné množství publikačních vrstev, podporovaných formátů, které pro naši publikaci můžeme využít je velké množství a tvorbu velkých projektů bychom mohli dále zdokonalovat. Pro představení základních funkcí je to v tuto chvíli už dostatečné a můžeme Marushku začít učit pokročilejším činnostem.

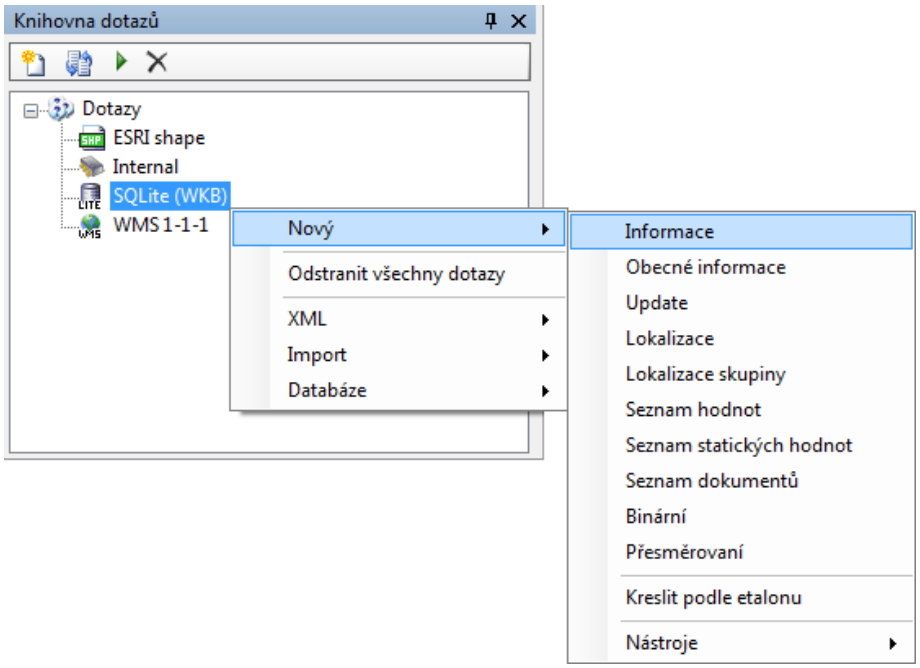
7 Interaktivní Marushka

Do této chvíle jsme pracovali s poněkud statickou mapou – pouze jsme zobrazovali naše data. V tuto chvíli můžeme mít ještě pocit, že Marushka nám zobrazuje jenom obrázek naší vodovodní sítě. A tento pocit se teď budeme snažit změnit. Naučíme Marushku zobrazovat i databázové údaje, které nejsou přímou součástí grafické prezentace dat a jsou uloženy v databázi u jednotlivých prvků. Základní nosnou kategorii všech databázových informací tvoří informační dotazy.

Začneme s informačním dotazem liniového objektu trasy. O daném prvku budeme chtít zobrazit následující údaje: *ID* (identifikátor databázového prvku), *rc* (popis blíže specifikující příslušný prvek), *Datum_vystavby* (atribut typu datum – u elementů není vyplněn – pouze pro ukázku práce s datem), *dim_prip* (dimenze potrubí – hodnoty naplníme pro testovací účely později), *zakazka* (číslo zakázky – pouze pro naše testování).

7.1 Vytvoření prvního informačního dotazu

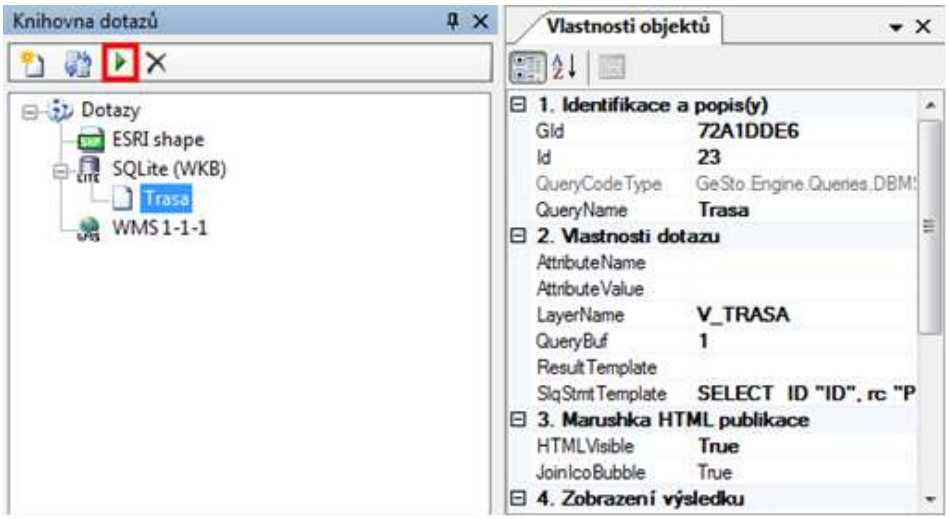
Otevřeme si knihovnu dotazů (z menu *Nástroje – Knihovny – Dotazy*). Každý vytvářený dotaz je vázán na konkrétní datový zdroj, takže vytvoříme v datovém zdroji SQLite nový **informační dotaz**:



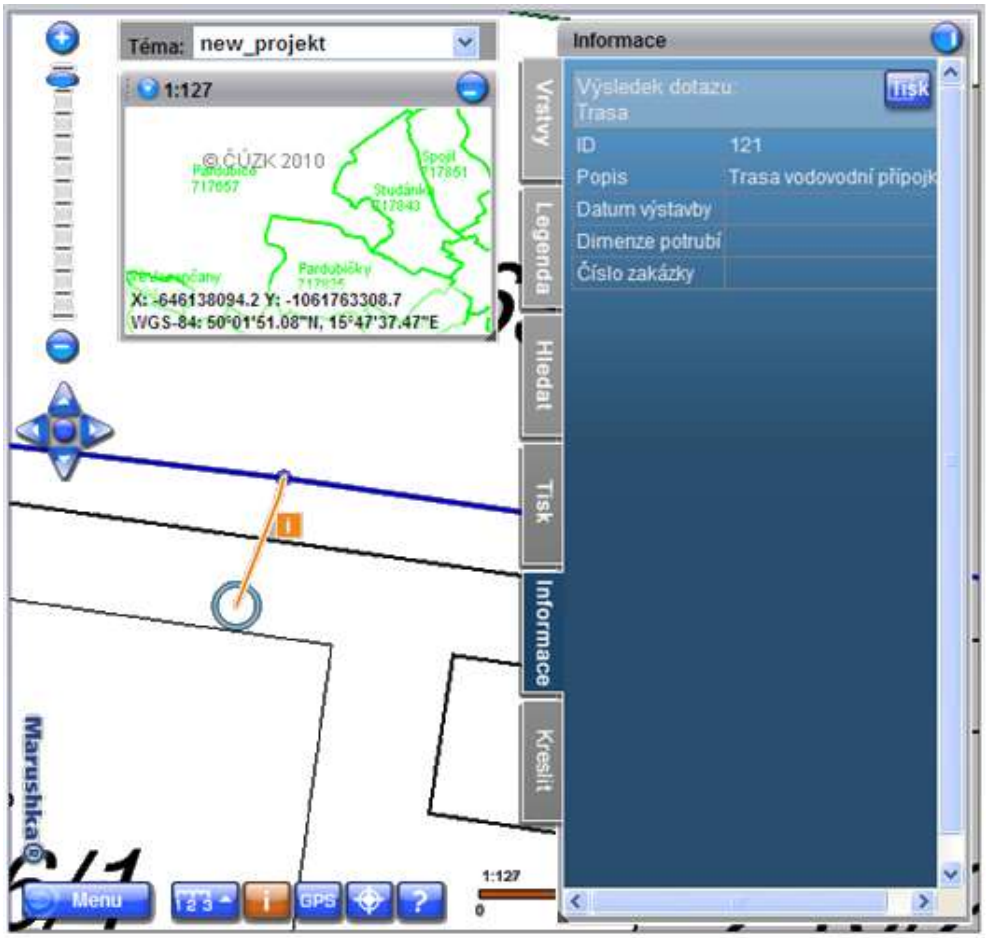
Tento dotaz si můžeme libovolně pojmenovat. Do vlastnosti tohoto dotazu *QueryName* vepíšeme například: „Trasa“ – toto pojmenování dotazu bude viditelné i ve webové publikaci. Do vlastnosti *LayerName* vepíšeme jméno fyzické vrstvy, ve které bude proveditelný tento dotaz (v našem případě databázové tabulky *V_TRASA*). U databázových datových zdrojů si musíme dávat pozor na to, že hodnota obsažená v tomto poli se nemusí shodovat se jménem formální vrstvy (obzvlášť při používání databázových pohledů, které slučují několik databázových tabulek dohromady). Posledním nutným předpokladem pro správnou funkčnost našeho prvního informačního dotazu je ještě vlastnost *SqlStmtTemplate*, do které vepíšeme samotný databázový select, který nám bude vracet konkrétní informace k danému prvku. V našem případě bude vypadat takto: `SELECT ID "ID", rc "Popis", Datum_vystavby "Datum výstavby", dim_prip "Dimenze potrubí", zakazka "Číslo zakázky" FROM V_TRASA WHERE ID=~(long)ID~`

Tento jednoduchý select nám tedy vrátí hodnoty v definovaných sloupcích, syntaxe s použitím vlnovky, která se objevuje ve *where klauzuli* propojí podmínku s prvkem, který byl vybrán v grafice. Přetypování **(long)** používáme z bezpečnostních důvodů jako prevenci proti případným „útokům zvenčí“ v podobě tzv. **SQL injection**.

Funkčnost konkrétního dotazu si můžeme otestovat ikonou pro testování dotazů (na obrázku zvýrazněna). V pravé části obrázku můžeme vidět část dialogového okna vlastností, které jsme právě změnili.



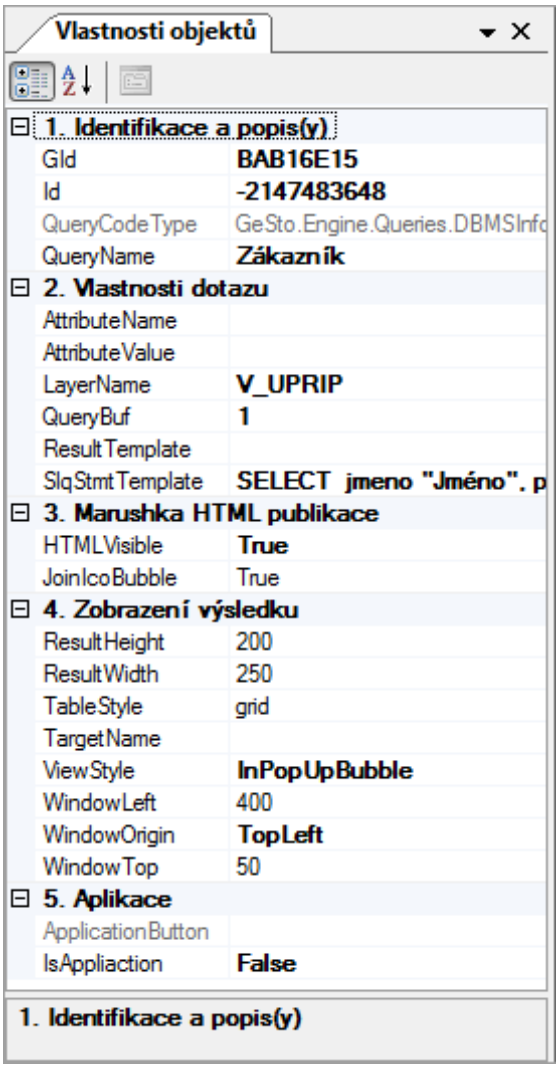
Náš první informační dotaz si teď vyzkoušíme i v lokálním WEB serveru. V lokálním WEB serveru se přepneme do Info režimu tlačítkem v dolní části serveru a při kliknutí levým tlačítkem myši na jakýkoliv prvek trasy aktivujeme funkci „*Preselect*“, která nám zvýrazní nejbližší prvek trasy, pravým tlačítkem pak můžeme překlikávat na další elementy. Při aktivní funkci *Preselect* se vedle zvýrazněného prvku objevuje Info ikona, jejíž obsah můžeme měnit a může se místo ní zobrazovat jakýkoliv rastrový obrázek, ale do těchto detailů v tomto Tutoriálu zacházet nebudeme. Po kliknutí na informační ikonu se v pravé části lokálního WEB serveru objeví výsledek informačního dotazu, který je svázán se zvýrazněným prvkem. V případě, kdy bychom měli u jedné fyzické vrstvy definováno více dotazů, tak po prvním kliknutí na informační ikonu by se nám prvně zobrazil jejich obsah, ze kterého bychom vybírali požadovaný dotaz.



Pro zobrazení textu, který se objeví v informační ikoně (místo standardního textu „i“), můžeme zobrazit i jiný text. Pomůže nám k tomu další pseudosloupec, jehož hodnotu definujeme v příslušné formální vrstvě. Do vlastnosti *DBColumsToClient* u vrstev *V_TRASA Řad* a *V_TRASA Přípojky* tedy doplníme řetězec: `'ID: '||id SET_INFO_ICON_TEXT`. To nám zajistí, že místo popisu *V_TRASA*, se nám bude zobrazovat *ID* příslušného prvku v plovoucí nápovědě.

7.2 Informační dotaz u buňky

Trochu odlišného přístupu využijeme u neliniových elementů. Vytvoříme si informační dotaz na ukončení přípojky. Začneme tedy opět vytvořením informačního dotazu stejným postupem, jaký jsme zvolili v minulém případě. Výsledkem bude informační dotaz *Zákazník* s těmito vlastnostmi:

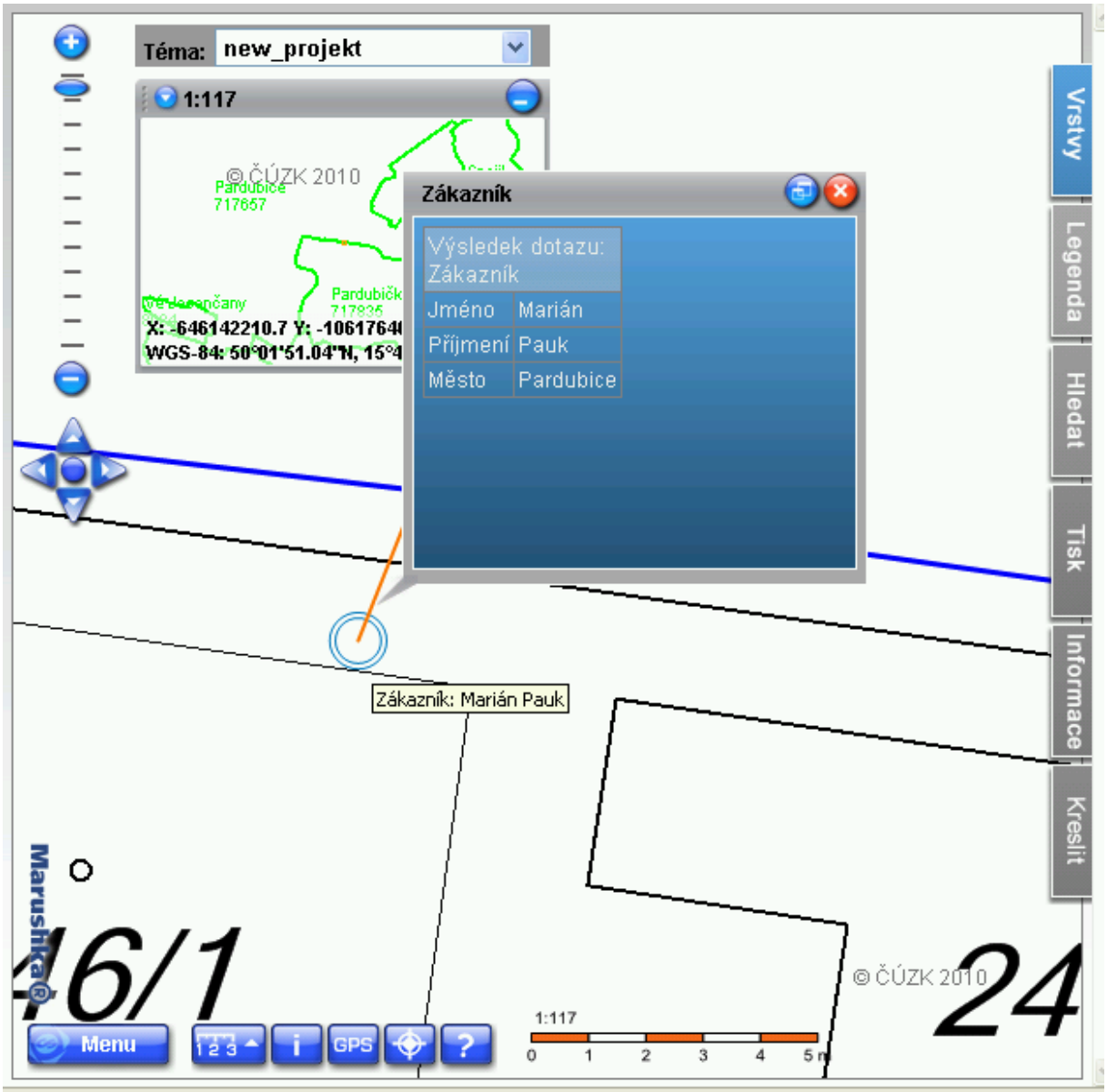


Na tomto příkladu si ještě ukážeme možnosti zobrazení výsledku informačních dotazů. Můžeme si vybrat, jestli ponecháme ve vlastnostech dotazu `ViewStyle=InPanel` (výsledek bude zobrazen na kartě informace v pravé části WEB serveru), *InPopUpBubble* (výsledek bude zobrazen v bublině nad příslušným prvkem) nebo *InNewWindow* (výsledek bude zobrazen v novém okně prohlížeče). V našem případě zvolíme prostřední možnost *InPopUpBubble* a výsledek informačního dotazu se nám tak bude zobrazovat ve „vyskakovací bublině“ nad prvkem.

Dále provedeme změnu textu zobrazovaného v informační ikoně pomocí už známého pseudosloupce *SET_INFO_ICON_TEXT*. V informační ikoně budeme chtít zobrazit jméno a příjmení zákazníka. Provedeme to zřetězením dvou sloupců a definice tohoto pseudosloupce bude vypadat takto: `ifnull(jmeno,'')||' '||ifnull(prijmeni,'') SET_INFO_ICON_TEXT`. Funkce *„ifnull“* je v tomto případě v databázovém prostředí SQLite téměř nutností, protože zřetězení nedokáže pracovat s prázdnými hodnotami. Dále můžeme doplnit ještě téměř stejně pseudosloupec *SET_INFO_ICON_LABEL*, který zobrazuje plovoucí nápovědu nad příslušným elementem. Posledním

důležitým pseudosloupcem, který je u buněk vhodný je ***SET_INFO_ICON_COVER***. Tento pseudosloupec zajišťuje zobrazení průhledné ikony nad vektorovou buňkou a může tak z buňky vytvořit aktivní prvek. Výsledkem pak bude ve vlastnosti *DBColumnsToClient* u formální vrstvy *V_UPRIP* následující řetězec (na pořadí definování sloupců nezáleží): `'true' SET_INFO_ICON_COVER, 'Zákazník: '|| ifnull(jmeno,'')||' '||ifnull(prijmeni,'') SET_INFO_ICON_LABEL, ifnull(jmeno,'')||' '||ifnull(prijmeni,'') SET_INFO_ICON_TEXT, '0 0 7' SET_PARS_POINT_FROM_CORG, '2' SET_PARS_CELLNAME, 'C 255 0 0 255 255 0 128 192' SET_PARS_REPLACE_COLOR.`

Vzhledem k tomu, že hodnota ‚true' u pseudosloupce *SET_INFO_ICON_COVER* zobrazuje informační ikony (i když neviditelné), tak si je ještě zobrazíme. Zobrazení informačních ikon je vlastností formální vrstvy. Zůstaneme tedy ve vlastnostech vrstvy *V_UPRIP* a vlastnost *GenerateInfo* nastavíme na *true*. Ve vlastnostech publikační vrstvy změníme nastavení vlastnosti *DefaultCheckedInfo* na hodnotu *true*, aby se nám tyto informační ikony zobrazovaly i po spuštění lokálního WEB serveru. Výsledek pak můžeme vidět v lokálním web serveru, kdy se nám po kliknutí levým tlačítkem myši na ukončení přípojky zobrazí následující informace:



8 Marushka upravuje data

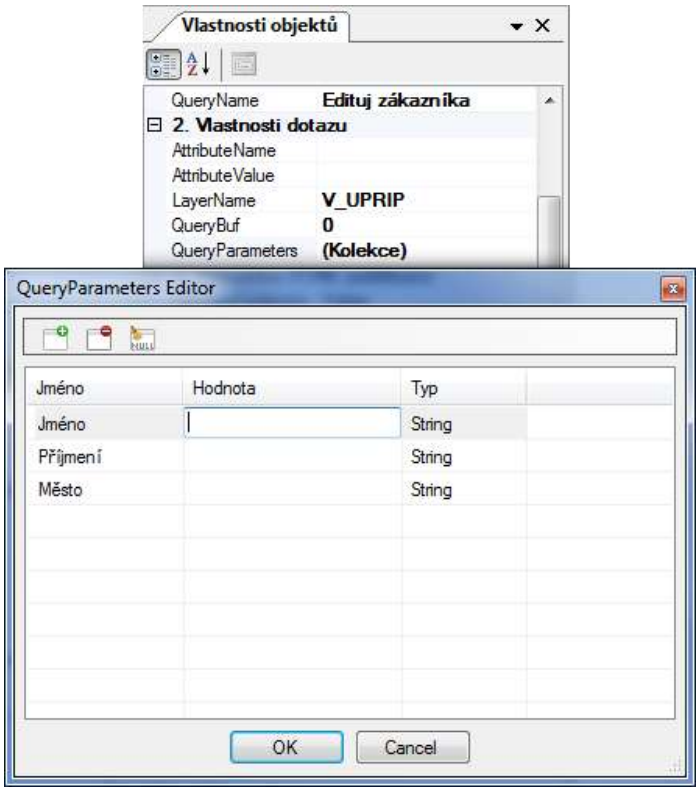
Marushku jsme už naučili zobrazovat grafická data, zobrazovat databázové údaje ke grafickým datům a teď se naučíme databázové údaje editovat. Vytvoříme si dvojici editačních dotazů. V prvním jednodušším případě půjde o prostou editaci textových údajů zákazníka.

8.1 Editace zákazníka

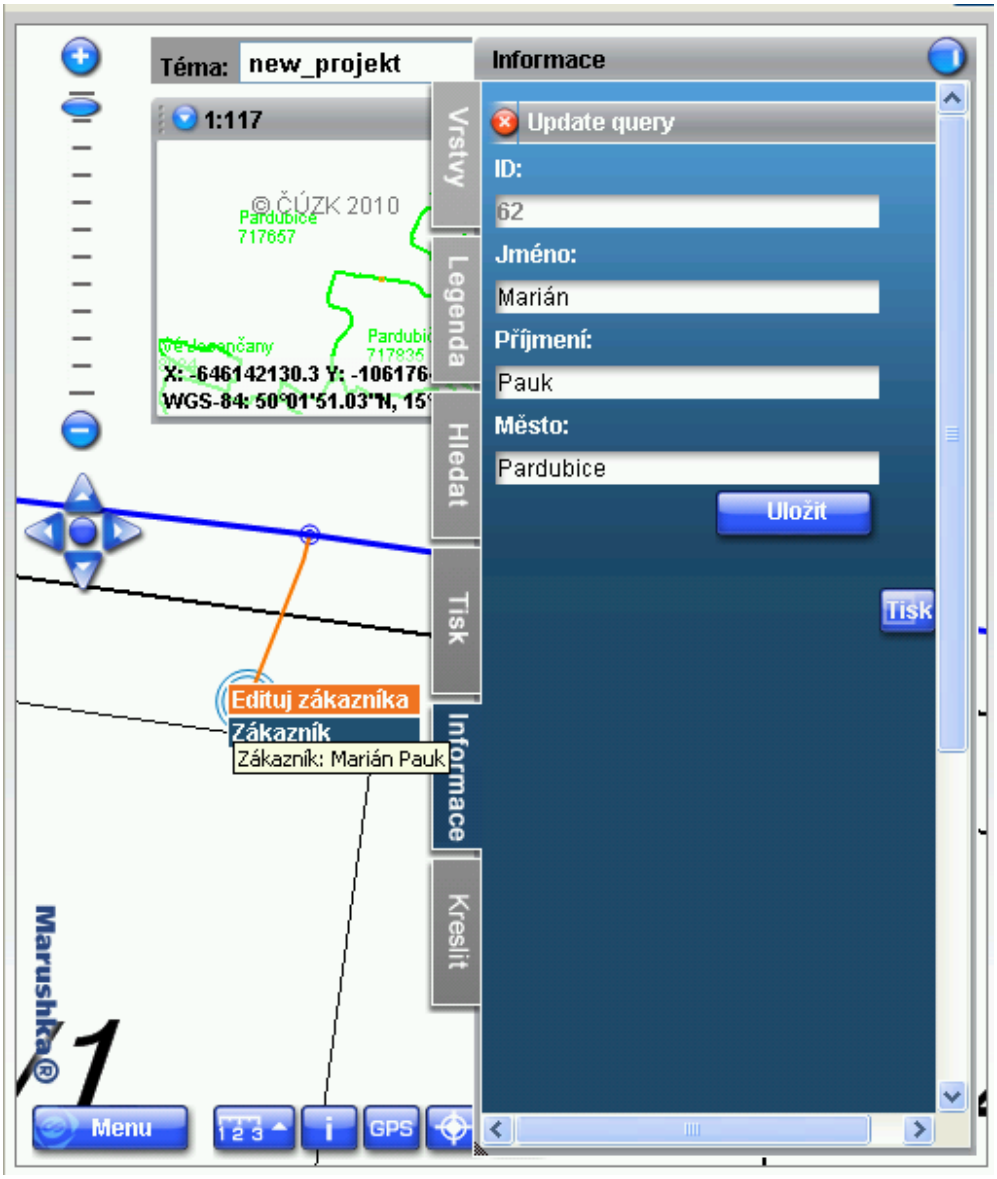
V knihovně dotazů si pomocí kontextového menu na datovém zdroje SQLite vytvoříme nový dotaz typu *Update*. Pojmenujeme si jej např. *Edituj zákazníka*. Ve vlastnosti *LayerName* musí být opět uvedena fyzická vrstva (tabulka) datového zdroje – v našem případě je to *V_UPRIP*. Ostatní vlastnosti ponecháme v tuto chvíli beze změny. Zastavíme se až u dvojice vlastností *InitSqlStmtTemplate* a *UpdSqlStmtTemplate*. První z nich je selectem sloupců, které chceme u příslušného editačního dotazu zobrazovat. Zobrazit a následně i editovat budeme chtít atributy: *Jméno*, *Příjmení* a *Město*. U tohoto selectu platí specifická pravidla jeho zápisu. Sloupce, které budou editačním dotazem následně editovány, musí být pojmenovány číslem a toto číslo musí být ve vzestupné řadě. Množství editovatelných sloupců pak musí odpovídat množství parametrů definovaných v editačním dotazu (vlastnost *QueryParameters*). Sloupce, které nebudou mít číselné indexované pojmenování, se budou považovat za sloupce pouze pro čtení a jejich hodnoty nebude možné měnit. Náš select bude tedy vypadat takto: `SELECT id "ID", jmeno "1", prijmeni "2", mesto "3" from V_UPRIP where id=~(long)ID~`.

Druhým SQL příkazem je fráze pro samotnou aktualizaci databázových údajů. Jednotlivé údaje se aktualizují podle pořadí parametru, které musí být shodné jako je pořadí v předchozím selectu. Důležité je v tomto případě opět přetypování jednotlivých parametrů jako ochrana proti SQL injekcím. Konstrukce našeho Updatu bude tedy vypadat takto: `UPDATE V_UPRIP set jmeno=~(string)1~, prijmeni=~(string)2~, mesto=~(string)3~ where id=~(long)ID~`.

Nesmíme ještě zapomenout na ony zmíněné parametry, které musíme ještě definovat ve vlastnosti *QueryParameters*. Pořadí parametrů musí odpovídat pořadí sloupců (parametrů) definovaných v příslušném selectu. Jejich názvy se budou zobrazovat u jednotlivých editovatelných polí ve webové publikaci. Editace parametrů je v otevřeném dialogovém okně intuitivní a není tak třeba dlouhých popisů. Přidání nového parametru se provede první ikonou, editace jména parametru pak dvojklikem myši. Definice parametrů bude v našem případě taková:



Ve webové publikaci (na našem lokálním WEB serveru) můžeme vidět výsledek naší snahy. U ukončení přípojky vidíme, že máme na výběr ze dvou dotazů, které jsou setříděny podle abecedy. Rovněž vidíme i plovoucí nápovědu, kterou jsme si vytvářeli v posledním informačním dotazu. Po aktivaci dotazu *Edituj zákazníka* se v pravém panelu v záložce Informace objeví výsledek našeho vytvořeného editačního dotazu. Do příslušných polí se nám načtou informace o vybraném prvku, z nichž pouze **ID nemůžeme editovat**. Ostatní informace můžeme editovat a aktualizovaná data uložit do databáze.



8.2 Editace přípojky (linie)

Na trase přípojky si ukážeme další možnosti editačních dotazů. Naučíme se editovat data pomocí statického číselníku, ukážeme si práci s daty typu datum. Ukážeme si, jak tento dotaz nabízet pouze u přípojek a nikoliv u řadů, když všechny prvky trasy leží ve stejné fyzické vrstvě.

Začneme vytvořením dalšího editačního dotazu stejně jako v minulém případě. Dotaz si pojmenujeme jako „*Edituj přípojku*“ a bude vztažený k fyzické vrstvě *V_TRASA*. Budeme chtít editovat pole *Datum výstavby*, *Dimenze přípojky* a *Číslo zakázky*. Navíc ještě zobrazíme *ID* a atribut *rc*. Výsledný základní select pro náš editační dotaz bude mít následující tvar: `SELECT id "ID", rc "Popis", Datum_vystavby "1", dim_prip "2", zakazka "3" from V_TRASA where id=~(long)ID~`. Všechny editovatelné parametry rovnou doplníme do *QueryParameters* s tím, že *Datum výstavby* bude typu *DateTime*, *Dimenze přípojky* a *Číslo zakázky* budou typu *Int*.

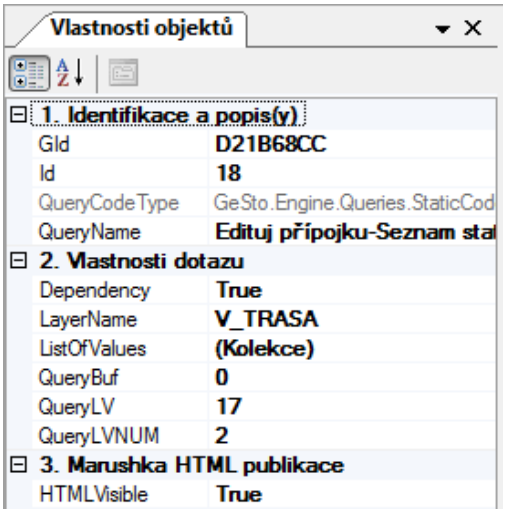
Vrátíme se do update fráze a tam vepíšeme následující příkaz: `UPDATE V_TRASA set Datum_vystavby=~(datetime)1~, dim_prip=~(int)2~, zakazka=~(int)3~ where`

`id=~(long)ID~`. V tomto příkladu si můžeme všimnout jiných datových typů, než které jsme využili doposud.

Takto připravený dotaz už by byl plně funkční a editace by se nám úspěšně povedla. To si ostatně můžeme sami vyzkoušet buď testováním dotazu v Knihovně dotazů, nebo v lokálním WEB serveru. My v tuto chvíli ale ještě náš editační dotaz doděláme do konce.

8.2.1 Vytvoření statického číselníku

Chtěli jsme, aby náš editační dotaz uměl pracovat se statickým číselníkem. V případě přípojky budeme brát v úvahu, že může mít dimenzi 0 (v případě neznámé dimenze), 32 nebo 40. Vytvoříme si tedy pomocný dotaz, který s naším editačním dotazem bude spolupracovat. V kontextovém menu si tedy zvolíme *Nový* dotaz – *Seznam statických hodnot*. Pro lepší orientaci v knihovně dotazů si tento dotaz pojmenujeme například: *Edituj přípojku-Seznam statických hodnot*. Ve vlastnostech tohoto dotazu vyplníme opět název fyzické vrstvy, ke které se dotaz vztahuje (*V_TRASA*), dále definujeme seznam statických hodnot (*ListOfValues*), který naplníme hodnotami: 0, 32 a 40 (jako oddělovač jednotlivých položek nám poslouží klávesa ENTER). Nakonec nás zajímají ještě vlastnosti *QueryLV* – zde vyplníme ID rodičovského dotazu, které si zkopírujeme z vlastnosti dotazu *Edituj přípojku* a poslední je pořadí parametru v rodičovském dotazu (v našem případě to bude hodnota 2, protože parametr pro dimenzi přípojky je v rodičovském dotazu druhý v pořadí. Výsledná podoba vlastností seznamu statických hodnot bude taková (pouze ID v QueryLV se bude lišit podle ID rodičovského dotazu):



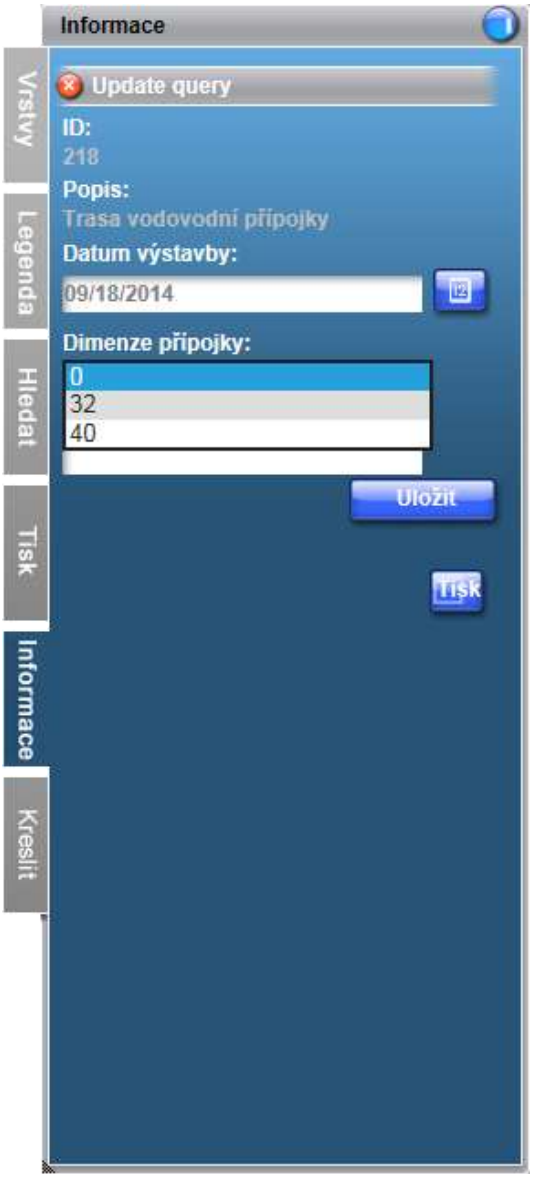
8.2.2 Omezení editačního dotazu na podmnožinu prvků

Dotaz se nám už zobrazuje podle našich představ (po vyzkoušení v lokálním WEB serveru), ještě bychom si ale přáli omezit dotaz jenom na přípojky. Využijeme atributu *rc*, který blíže specifikuje příslušný prvek. Do Vlastnosti *DBColumnsToClient* ve formální vrstvě *V_TRASA Přípojky* jej doplníme, abychom s ním mohli dále pracovat.

Přejdeme zpět do knihovny dotazů na náš dotaz "*Edituj přípojku*". Do vlastnosti *AttributeName* vepíšeme řetězec: *rc*. Do vlastnosti *AttributeValue* vepíšeme hodnotu: "*Trasa vodovodní přípojky"*. Tato podmínka nám zajistí spouštění dotazu pouze pro fyzickou vrstvu, která bude navíc splňovat tuto atributovou podmínku. Podmínka není definována přímo v databázovém prostředí, proto je nutné atribut, se kterým podmínka pracuje, uvádět i ve stahovaných atributech formální vrstvy.

8.2.3 Otestování výsledného editačního dotazu

Pokud jsme pracovali správně, tak editační dotaz v tuto chvíli nebude dostupný pro trasu řadů a bude dostupný pouze pro přípojky. Na obrázku vidíme výsledek našeho snažení. V náhledu vidíme možnosti editace prvku s ID=218. Parametr ID je stejně jako parametr Popis „*Trasa vodovodní přípojky*" needitovatelný. Dále vidíme u *Data výstavby* nový aktivní prvek vpravo od textového pole. Je to ikona kalendáře, díky které můžeme vybírat konkrétní datum. V dimenzi přípojky vidíme možnosti, které jsme si definovali jako statické hodnoty pro tento konkrétní dotaz. **U několika přípojek si zkusíme tuto hodnotu vyplnit** (tak, aby hodnoty byly u různých přípojek různé), tato data budeme potřebovat v některém z dalších kroků tvorby našeho projektu. V náhledu není vidět atribut *Číslo zakázky*, který ale není nijak podstatný a součástí projektu je jenom z důvodu testování různých datových typů.


Informace
Vrstvy
Legenda
Hledat
Tisk
Informace
Kreslit
Update query
ID:
218
Popis:
Trasa vodovodní přípojky
Datum výstavby:
09/18/2014
Dimenze přípojky:
0
32
40
Uložit
Tisk

9 Učíme Marushku kreslit

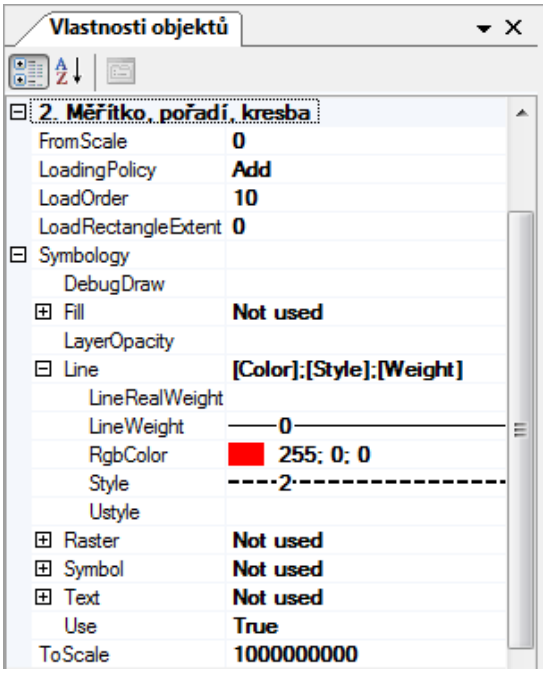
Po tom, co jsme Marushku naučili aktualizovat původní data, je na čase, aby se Marushka naučila nová data zakreslovat. Vytvoříme si dvojici kreslící dotazů, kterými budeme umět ve webovém prostředí kreslit nová data, která se následně zapíší i do naší SQLite databáze.

9.1 Vytvoření kreslícího dotazu – liniový objekt

Základem pro kreslení, jehož výsledek bude následně uložen do databáze, je opět dotaz v Knihovně dotazů. Využijeme opět kontextové menu v knihovně dotazů na datovém zdroji SQLite. Zde vybereme možnost *Kresli podle etalonu*. Dotaz si nazveme: *Kresli přípojku*. Ve vlastnostech dotazu budeme měnit pouze dvě věci. Základní select, který nám umožní nastavení atributů, které s novým prvkem budou vstupovat do databáze (mohou to být i uživatelsky definované atributy, to si ale ukážeme až na druhém příkladu). V některých databázových tabulkách máme atribut „*userdraw*“, který nám zajistí odlišení prvků, které v databázi vznikly ve webovém prostředí. V příslušném selectu tedy nastavíme hodnotu 1 a vyplníme rovněž i atribut RC, který nám definuje, že jde o přípojku. Select pak bude vypadat následovně: `SELECT '1' "userdraw", 'Trasa vodovodní přípojky' "rc"`. Druhou věcí, kterou musíme ve vlastnostech dotazu doplnit, je položka etalonu, kterou bude kreslící dotaz potřebovat. A tu si právě teď vytvoříme.

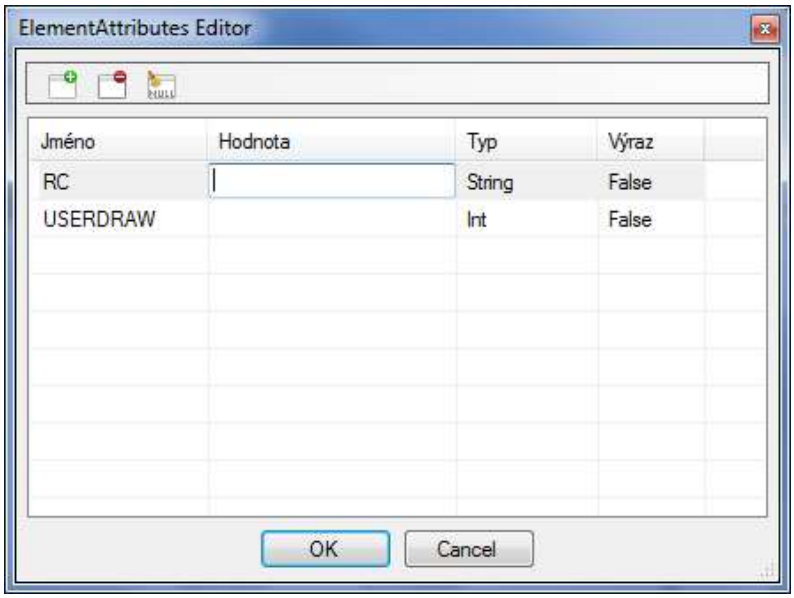
9.2 Vytvoření položky v Etalonu

Etalon nám v tomto případě poslouží jako definice symbologie pro nově zakreslovaný element včetně určení databázové tabulky, do které má být po zakreslení zapsán. Otevřeme si Knihovnu etalonů (z menu *Nástroje – Knihovny – Etalon*). V kontextovém menu datového zdroje SQLite vybereme možnost *Přidej položku*. Ve vlastnosti *Description* si ji pojmenujeme jako *Přípojka - linie*, *TableName* bude v našem případě *V_TRASA*. Přípojku zakreslenou na webovém serveru budeme chtít odlišit barevně od přípojek, které už jsou v datovém skladu. Zvolíme tedy vhodnou uživatelskou symbologii ve vlastnosti *Symbology* podle následujícího obrázku:



Dále víme, že budeme chtít kreslit liniový element, proto do vlastnosti *GeomType* nastavíme položku *WKBLineString*.

Nakonec musíme ještě v této položce Etalonu definovat parametry, které bude náš kreslící dotaz využívat. Tyto atributy definujeme ve vlastnosti *ElmAttribs* (viz obrázek):



Takto vytvořenou položku Etalonu už můžeme zapsat i do vlastnosti *EtalonItem* našeho kreslícího dotazu.

9.3 Testování kreslícího dotazu a odhalení nedostatků

Výsledek našeho kreslícího dotazu si můžeme prohlédnout v prostředí lokálního WEB serveru. Najdeme jej na kartě ***Kreslit***. Po zakreslení liniového objektu a po kliknutí na tlačítko *Uložit* rovnou vidíme výsledek. Pokud jsme postupovali přesně podle postupu, tak zjistíme, že není žádný vizuální rozdíl mezi přípojkou, která je už v databázi a kterou jsme právě zakreslili – všechny přípojky jsou oranžové. Je to způsobeno tím, že všechny přípojky jsou v jedné formální vrstvě, která je zobrazovaná v uživatelské (oranžové) symbologii.

9.3.1 Oprava chyby

Výsledku, který bychom si přáli vidět, dosáhneme tak, že **zklonujeme** formální vrstvu *V_TRASA Přípojky*. Klon nazveme *V_TRASA Přípojky – uživatelské*. Tuto novou formální vrstvu rovnou přidáme i do publikační vrstvy *Voda*. Takto vytvořené dvě vrstvy by nám v tuto chvíli vracely stejná data, proto musíme ještě upravit podmínku tak, abychom dosáhli požadovaného výsledku. U formální vrstvy *V_TRASA – Přípojky* bude *DBWhereClause* vypadat takto: *rc='Trasa vodovodní přípojky' and userdraw is null*. Využili jsme atributu *userdraw*, o kterém jsme si mohli v databázi zjistit, že není u prvků, které vznikly mimo prostředí Marushky, vyplněn. Ve zklonované vrstvě *V_TRASA Přípojky – uživatelské* změníme *DBWhereClause* naopak na: *rc='Trasa vodovodní přípojky' and userdraw=1*, čímž zajistíme, že v této vrstvě se budou naopak zobrazovat pouze přípojky, které jsme nakreslili v prostředí Marushky. U tohoto klonu změníme ještě vlastnost *Symbology*, aby vůbec tento klon měl požadovaný smysl. V této vlastnosti vypneme použití uživatelské symbologie. Po dalším otestování v prostředí lokálního WEB serveru už dostaneme korektní výsledek a naše přípojka bude zobrazena tak, jak byla uložena do datového skladu (v našem případě červenou přerušovanou čarou).

9.4 Vytvoření kreslícího dotazu s atributy

Úplně stejným způsobem si vytvoříme kreslící dotaz na *Ukončení přípojky*. Jediným rozdílem, na který je potřeba upozornit, je typ elementu v položce etalonu. Tady zvolíme hodnotu *WKBPoint*. Jinak platí postup z minulého příkladu.

Po vytvoření dvojice *Kreslícího dotazu* a *Položky etalonu* pokračujme v tom, co je potřeba doplnit a v čem se nový kreslící dotaz liší od toho předchozího.

9.4.1 Kreslící dotaz

Vlastnost *SqlStmtTemplate* bude mít v sobě navíc atributy, které může uživatel naplnit ve chvíli pořízení daného záznamu – databázový prvek tak rovnou dostane své vlastnosti. V našem případě budou součástí selectu tři atributy: *Jméno*, *Příjmení* a *Město* (všechny typu string). Musí být tedy

definovány ve vlastnosti *QueryParameters*. Vzhledem k tomu, že se jedná o buňku, tak musíme doplnit ještě název buňky. Výsledný select pak bude vypadat takto: `SELECT '1' "userdraw", 'Ukončení přípojky' "rc", ~(string)1~ jmeno, ~(string)2~ prijmeni, ~(string)3~ mesto, 'V_UPRI2' CELLNAME.`

Atribut *userdraw* v našem případě opět odlišuje data vzniklá v prostředí Marushky.

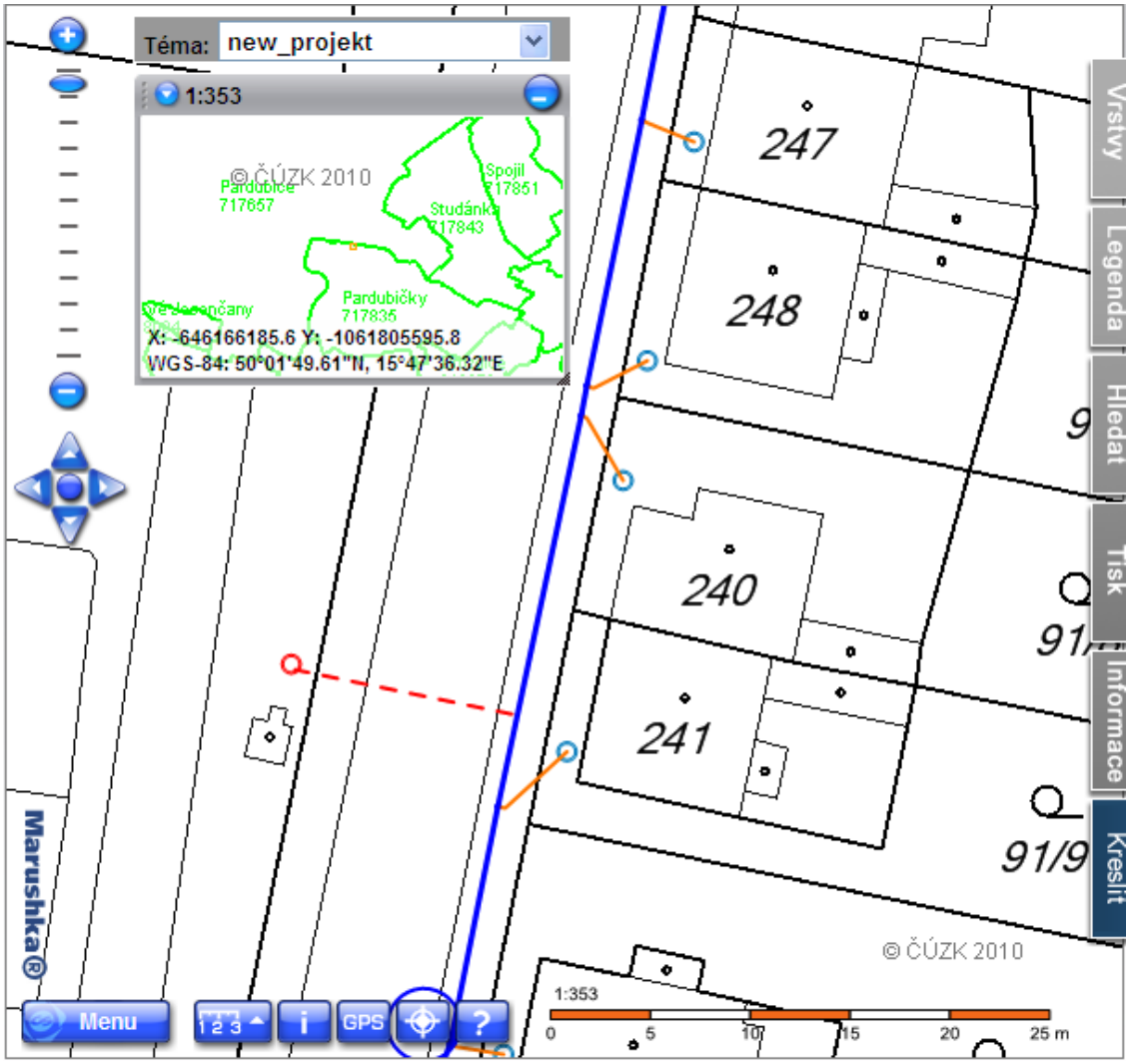
9.4.2 Položka etalonu

V položce etalonu musíme definovat všechny atributy – jak ty, které budeme zadávat při kreslení dotazu, tak i ty, které budou vyplněny automaticky při vložení prvku do databáze. V tomto případě jsou ale názvy parametrů pojmenovány podle sloupců v dané tabulce.

9.4.3 Formální vrstva

V klonu formální vrstvy tentokrát nebudeme upravovat vlastnost *Symbology*, ale ve vlastnosti *DBColumnsToClient* odstraníme dva pseudosloupce: *SET_PARS_CELLNAME* a *SET_PARS_REPLACE_COLOR* a to z toho důvodu, že tentokrát jsme pro zobrazení přímo využili buňku, která je už i v knihovně buněk červená a tak není důvod ji zobrazovat jinak. Části podmínky ve vlastnosti *DBWhereClause* zůstávají stejné, jako v předchozím kreslícím dotazu.

Pokud jsme pracovali správně, tak výsledek našeho tvoření bude vypadat asi nějak takto:



Na obrázku vidíme oranžové databázové přípojky se světle modrým ukončením – to jsou přípojky, které už v databázi byly původně a červenou přerušovanou čarou s červeným ukončením přípojky vidíme přípojku, kterou jsme vložili do databáze v prostředí Marushky. A právě tento stav byl cílem této kapitoly.

10 Mazaná Marushka

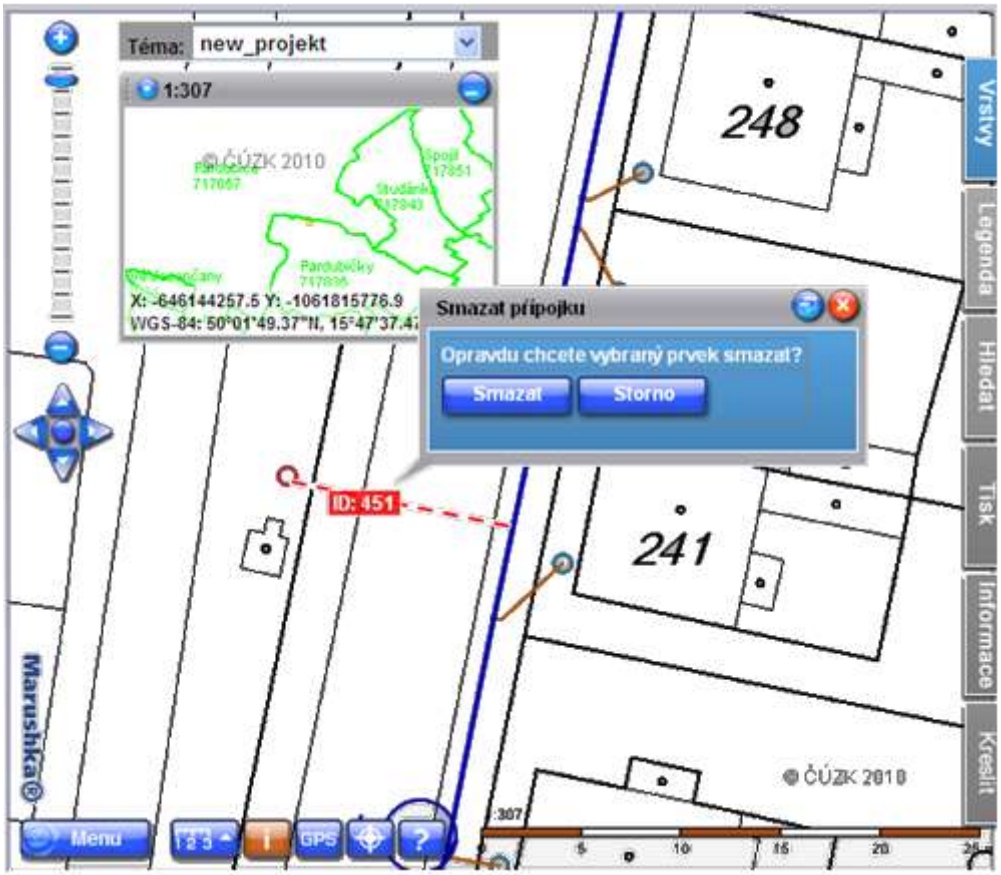
V této kapitole budeme chtít, aby Marushka dokázala smazat z databáze elementy, které jsme předtím sami v prostředí Marushky vytvořili. Nechceme, aby se data, která už jsou uložená v databázi, dala odstraňovat, a tak k uživatelským datům a původním datům budeme zase přistupovat dvojím způsobem. Opět využijeme atributu: *userdraw*.

Tento atribut v první řadě doplníme do vlastnosti *DBColumnsToClient* ve formálních vrstvách, ve kterých jej budeme potřebovat. To znamená, že tuto vlastnost upravíme u formálních vrstev *V_TRASA Přípojky – uživatelské* a *V_UPRIP uživatelské*.

10.1 Vytvoření dotazu pro smazání prvku

Dotaz pro smazání prvku je velmi jednoduchý. Vytvoříme si dvojici těchto dotazů – jak pro smazání ukončení přípojky, kterou jsme zakreslili, tak i jejího ukončení. Oba dva dotazy budou vypadat stejně – lišit se budou pouze v názvu fyzické vrstvy (databázové tabulky). Popíšeme si tedy pouze jeden tento dotaz, druhý si vytvoříme sami bez podrobného popisu.

V kontextovém menu Knihovny dotazů v datovém zdroji SQLite vybereme *Nový – Nástroje – Smazat prvek*. Dotaz přejmenujeme na: *Smazat přípojku*. Do vlastnosti *LayerName* vepíšeme jméno fyzické vrstvy *V_TRASA*. *SqlStmtTemplate* bude obsahovat jednoduchou frázi pro smazání aktuálního elementu: `DELETE FROM V_TRASA WHERE ID = ~(long)ID~`. Abychom dosáhli toho, že se nám dotaz pro mazání elementu nabízel **pouze u prvků, které vznikly v prostředí Marushky**, pak je nutné doplnit ještě vlastnosti dotazu *AttributeName* a *AttributeValue*. Do vlastnosti *AttributeName* vložíme text: *userdraw* a do vlastnosti *AttributeValue* hodnotu 1. Této hodnoty příslušný atribut nabývá pouze v případě, kdy byl prvek zapsán do databáze v prostředí Marushky. Poslední vlastností tohoto dotazu, kterou budeme měnit, je *ViewStyle*. V tomto případě je podstatně efektivnější zobrazení otázky, jestli si opravdu element přejeme smazat, do „bubliny“. Vlastnost *ViewStyle* tedy nastavíme na hodnotu *InPopUpBubble*. Stejným způsobem pak vytvoříme dotaz pro smazání ukončení přípojky. Pokud jsme pracovali správně, pak se nám v info režimu WEB serveru bude u přípojek zakreslených v prostředí Marushky objevovat dotaz pro smazání prvku a po jeho výběru se nám zobrazí dialogové okno (bublina) s otázkou pro potvrzení smazání prvku.



11 Marushka hledá a najde

Často potřebujeme najít určitý prvek na základě nějakých konkrétních databázových vlastností, příp. potřebujeme najít zeměpisné souřadnice. Pro lokalizaci pomocí GPS souřadnic má Marushka k dispozici již vytvořený dotaz skrývající se na WEB serveru pod ikonou GPS. Tento dotaz nebudeme nijak popisovat, ale můžeme si jej vyzkoušet.

My se budeme věnovat vytvoření vlastního lokalizačního dotazu. Využijeme databázových vlastností prvků, které jsou již v datovém skladu, a budeme chtít najít přípojku náležející konkrétnímu zákazníkovi (podle jeho jména). Chceme, aby lokalizační dotaz využil podle zadaných kritérií výběr z dynamického seznamu. Vytvoříme tedy dvojici dotazů – hlavní lokalizační dotaz a pomocný seznam hodnot.

11.1 Vytvoření lokalizačního dotazu

V knihovně dotazů pomocí kontextového menu datového skladu SQLite vybereme položku *Nový – Lokalizace*. Dotaz si pojmenujeme jako *Lokalizace **zákazníka***. Informace o zákazníkovi jsou uloženy v tabulce *V_UPRIP*, jejíž název zadáme do vlastnosti *LayerName*. Vlastnost *QueryBuf* můžeme ponechat beze změny. Záleží na tom, jak budeme tento dotaz využívat. Pokud budeme vyhledávat pouze podle příjmení, tak je vhodné, aby byla hodnota zvýšena, a dotaz nám bude vracet více možností k výsledné lokalizaci. Ve vlastnosti *QueryParameters* vytvoříme jeden záznam „jméno“, který bude typu string. Základní *SqlStmtTemplate* bude mít následující tvar:

```
select xmin-5000 XMIN, ymin-5000 YMIN, xmax+5000 XMAX, ymax+5000 YMAX, id
ID, ifnull(jmeno,'')||' '||ifnull(prijmeni,'') LABEL FROM V_UPRIP WHERE
ifnull(jmeno,'')||' '||ifnull(prijmeni,'')=~(string)1~
```

V tomto selectu vidíme, že výsledkem bude ukončení přípojky ve výřezu mapového okna, které je definováno rohy *XMIN*, *YMIN*, *XMAX*, *YMAX* – tyto souřadnice se přebírají přímo z vybraného databázového elementu a jsou upraveny tak, aby zobrazená buňka nezaujímala celé mapové okno. V atributu *LABEL*, který bude popisem příslušného prvku je zřetězeno jméno a příjmení zákazníka, pro lepší přehlednost je jméno od příjmení oddělené mezerou. Konstrukce ***ifnull*** je nutná pro případ, kdy by jedno z polí (jméno nebo příjmení) bylo prázdné a v databázovém prostředí SQLite je tento zápis nutný pro korektní chování selectu.

Dále změníme také symbologii výsledku lokalizace. Ve vlastnosti *Symbology* si nastavíme barevnou interpretaci podle našeho vlastního uvážení. Předdefinována je světle modrá barva se světle modrou výplní, takže i bez ručního zásahu do symbologie bude ve většině případů výsledek lokalizace viditelný.

Poslední vlastností, kterou v tomto dotazu změníme je vlastnost *DynamicCodeList*. Tuto vlastnost změníme na *True* a zapneme tím funkci našeptávač, kdy podle psaných znaků budeme dostávat interaktivní nabídku existujících záznamů definovaných v dynamickém seznamu hodnot.

11.2 Vytvoření seznamu hodnot

V knihovně dotazů pomocí kontextového menu datového skladu SQLite vybereme položku *Nový – Seznam hodnot*. Nově vytvořený dotaz si přejmenujeme na: *Lokalizace zákazníka – Seznam hodnot*.

Ve vlastnosti *LayerName* opět vyplníme jméno fyzické vrstvy, ke které je tento dotaz vázán (*V_UPRIP*), vyplníme *ID rodičovského dotazu* (*QueryLV*), které najdeme ve vlastnostech vytvořeného lokalizačního dotazu. Lokalizace zákazníka a pořadí parametru v rodičovském dotazu (*QueryLVNUM*). V našem případě máme parametr pouze jeden jediný, tak do této vlastnosti vepíšeme jedničku.

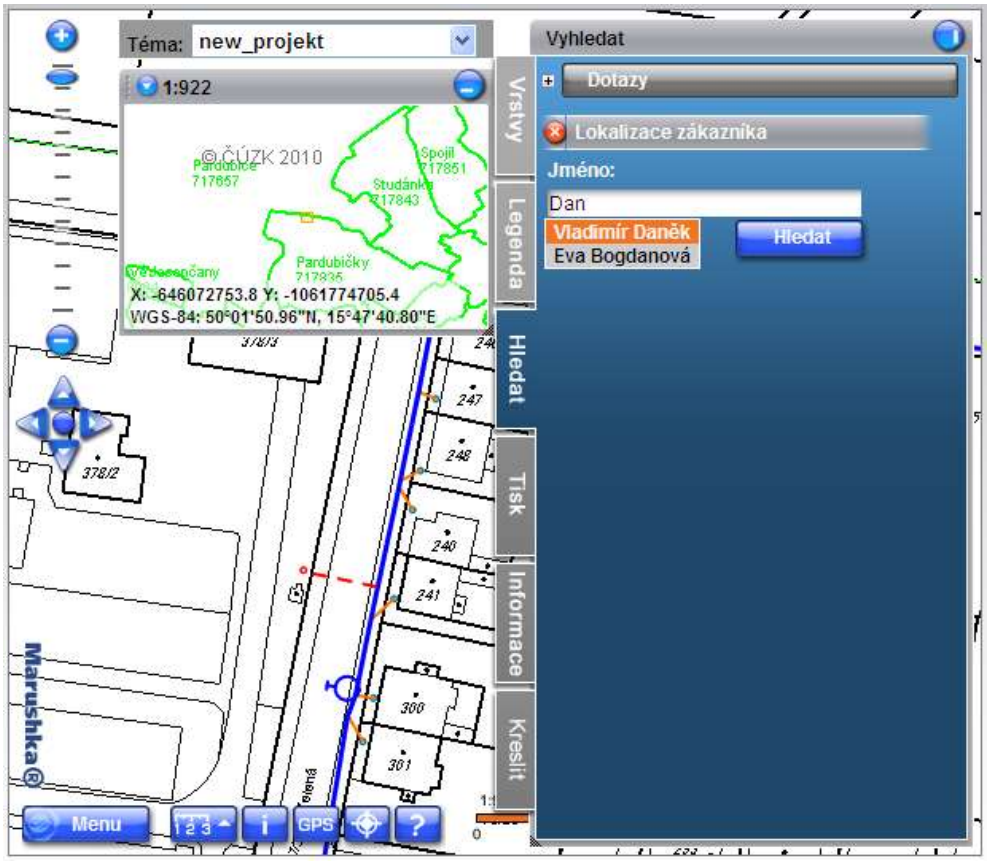
Ve vlastnosti *SqlStmtTepmlate* bude fráze selectu následujícího znění:

```
SELECT ifnull(jmeno,'')||' '||ifnull(prijmeni,'') FROM V_UPRIP WHERE
prijmeni like '%~1~%'.
```

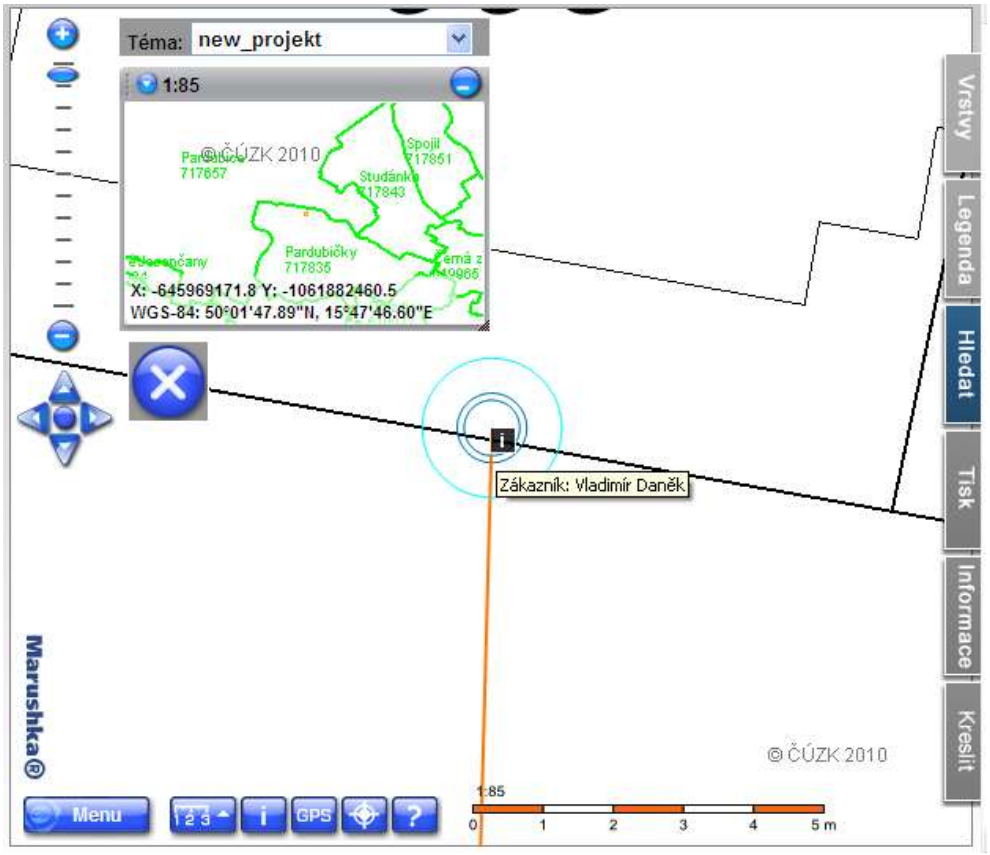
Výsledkem selectu bude seznam hodnot splňující definovanou podmínku. Takovou definicí dosáhneme toho, že v seznamu hodnot se nám budou nabízet položky, kde dosavadní zápis v textovém poli lokalizačního dotazu bude obsahovat libovolnou část z hodnoty ve sloupci *„prijmení“*. Pokud bychom odstranili symbol „%“ před samotným parametrem, pak by se nám zobrazovala pouze příjmení, která by tímto řetězcem začínala.

11.3 Testování lokalizačního dotazu

V prostředí lokálního WEB serveru přejdeme na záložku Hledat, kde vidíme jediný lokalizační dotaz, který jsme právě vytvořili. Tento dotaz vybereme a objeví se nám dialogové okno pro zápis jména (resp. příjmení) zákazníka, kterého chceme najít. Ve chvíli, kdy budeme postupně vepisovat text, tak se nám bude zobrazovat seznam hodnot splňující kritérium, které jsme si popsali výše. Příklad jednoho takového pokusu můžeme vidět na následujícím obrázku:



Vidíme, že po zapsání řetězce „**Dan**“ se nám objevují dvě možnosti. Zákazník Vladimír **Dan**ěk i zákaznice Eva Bog**dan**ová. Obě jména obsahují řetězec „**dan**“ (bez ohledu na velikost písma). Po vybrání položky, kterou jsme opravdu požadovali a po potvrzení tlačítkem *Hledat*, se nám konečně objeví výsledek lokalizace:



Na obrázku vidíme přípojku, která je vycentrovaná a omezující obdélník mapového okna je přizpůsoben většímu okraji, který jsme definovali v naší Select frázi lokalizačního dotazu. V příkladu byla upravena symbologie výsledku bez výplně, abychom viděli i ukončení přípojky. Po najetí myši nad informační ikonu se nám objeví bližší informace o zákazníkovi a vidíme, že výsledek našeho hledání byl korektní.

12 Marushka – dokumentaristka

Marushka dokáže komplexně pracovat s veškerými dokumenty, které jsou přidruženy k jednotlivým databázovým elementům. Dokumenty dokáže zobrazovat, vkládat je do databázového datového skladu a taky je z něj odstraňovat. V této kapitole si vytvoříme dvojici dokumentačních dotazů, které budou umožňovat kompletní správu dokumentů. Dokumentační dotaz se bude vztahovat k ukončení přípojky – může jít o jakoukoliv dokumentaci, která se k přípojkám může vztahovat.

12.1 Vytvoření binárního dotazu

Pro samotnou práci s dokumentem si musíme vytvořit binární dotaz, díky kterému budeme příslušný dokument otvírat. Tento dotaz ještě není nijak provázán s grafickou databázovou tabulkou.

V knihovně dotazů pomocí kontextového menu datového skladu SQLite vybereme položku *Nový – Binární*. Vytvořený dotaz přejmenujeme na: *Dokumentace přípojky – Binární*. V jeho vlastnostech budeme editovat pouze frázi databázového selectu – *SqlStmtTemplate*. Bude mít následující tvar.

```
select DOC DOCUMENT, BLOB_TYPE EXTENSION, BIRTH_DATE BIRTH_DATE FROM demo_doc WHERE id=~(long)ID~
```

SQL fráze obsahuje výčet tří povinných sloupců, které jsou obsaženy v naší databázové tabulce, kterou jsme si vytvořili v úvodní části tohoto projektu.

12.2 Vytvoření dotazu pro správu dokumentů

V knihovně dotazů pomocí kontextového menu datového skladu SQLite vybereme položku *Nový – Nástroje – Prohlížeč souborů*. Vytvořený dotaz si pro přehlednost přejmenujeme na *Dokumentace přípojky*. Tento dotaz již bude přímo vázaný ke grafické databázové vrstvě a SQL fráze definované v dalších vlastnostech se tak budou vztahovat přímo ke grafickému elementu, ke kterému budou jednotlivé dokumenty přidruženy. Ve vlastnosti *LayerName* tedy doplníme název fyzické vrstvy pro ukončení přípojky (*V_UPRIP*). Pro lepší přehlednost ještě změníme vlastnost *ViewStyle* na *NewWindow*. Další vlastnosti ponecháme beze změny, přejdeme až k poslední trojici vlastností – k SQL frázím, které nám umožní dokumenty nahrávat, prohlížet a odstraňovat. Začneme seznamem dokumentů k příslušnému databázovému prvku. Konstrukce vlastnosti *SqlDocListTemplate* bude vypadat takto:

```
SELECT id ID, BLOB_TYPE EXTENSION, LABEL LABEL, BIRTH_DATE BIRTH_DATE, '*ID Binárního dotazu*' BINARYQUERY FROM demo_doc WHERE ID_GRAPHICS_ELEMENT=~(long)ID~
```

V tomto selectu musíme **upravit ID Binárního dotazu** – sem do apostrofů vložíme skutečné ID dotazu, který jsme si vytvořili v minulém kroku.

Dále vytvoříme dotaz, který nám umožní vložit příslušný dokument, kterému se vytvoří vazba na vybraný grafický element. Konstrukce vlastnosti *SqlInsertDocumentTemplate* bude tedy vypadat takto:

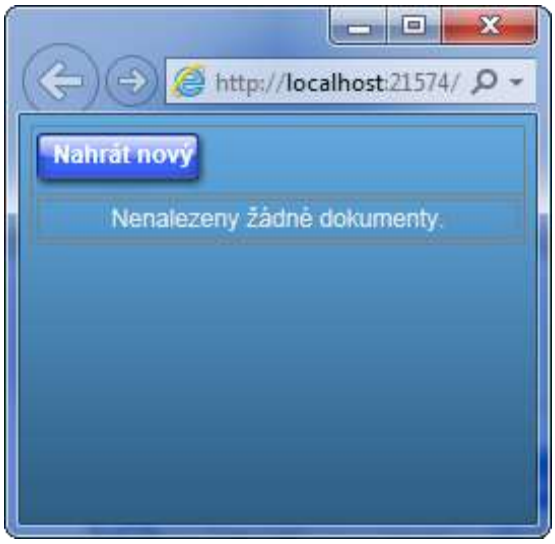
```
INSERT INTO demo_doc (BLOB_TYPE,LABEL,DOC,ID_GRAPHICS_ELEMENT) VALUES (~BLOB_TYPE~,~FILE_NAME~,~DOCUMENT~,~ELEMENTID~)
```

Uvedené parametry jsou v této frázi povinné a uvádí typ dokumentu, jeho název, samotný dokument a vazbu na grafický prvek.

Posledním krokem k vytvoření dokumentačního dotazu pro kompletní správu souborů je fráze pro smazání příslušného dokumentu. *SqlDeleteDocumentTempate*, která je z uvedených tří vlastností tou nejjednodušší:

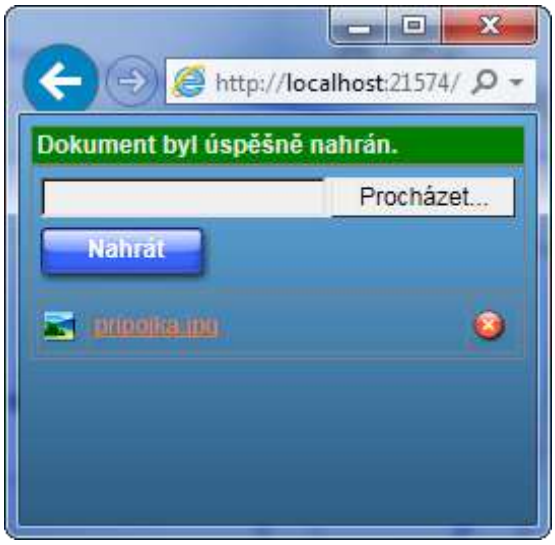
```
DELETE FROM demo_doc WHERE ID=~(long)ID~
```

Předchozími kroky jsme splnili všechny požadavky k funkčnímu dokumentačnímu dotazu a můžeme si rovnou vyzkoušet jeho funkčnost. V lokálním WEB serveru si vybereme libovolnou buňku ukončení přípojky, na které tento dotaz budeme testovat. Z dotazů, které máme k této buňce k dispozici, si vybereme položku *Dokumentace přípojky*. Poté se nám otevře nové okno prohlížeče dokumentů:



V tuto chvíli je seznam dokumentů prázdný. Teď si můžeme vyzkoušet práci s dokumenty. Klikneme na tlačítko *Nahrát nový*. Objeví se nám řádek pro název dokumentu. Po kliknutí na tlačítko *Procházet* se nám otevře dialogové okno pro otevření souboru. Ve složce `MarushkaExamples\Tutorial\Soubory` najdeme obrázkový soubor „`pripojka`“. Ten otevřeme a v textovém poli se nám objeví text s cestou k našemu souboru. Po kliknutí na tlačítko *Nahrát* dojde k databázovému insertu příslušného dokumentu.

Dialogové okno se seznamem souborů se nám okamžitě změní a získáme v něm i dočasnou informaci o úspěšném vložení souboru:



Vložený obrázek si pak můžeme prohlédnout prostým kliknutím na řádek s jeho názvem. Obrázek se následně otevře v novém okně webového prohlížeče. Dokument můžeme z databáze i odstranit ikonou s křížkem v pravé části dialogového okna se seznamem dokumentů. Tím dojde samozřejmě pouze k odstranění dokumentu v databázi, nikoliv ke smazání souboru z disku, odkud jsme jej vložili.

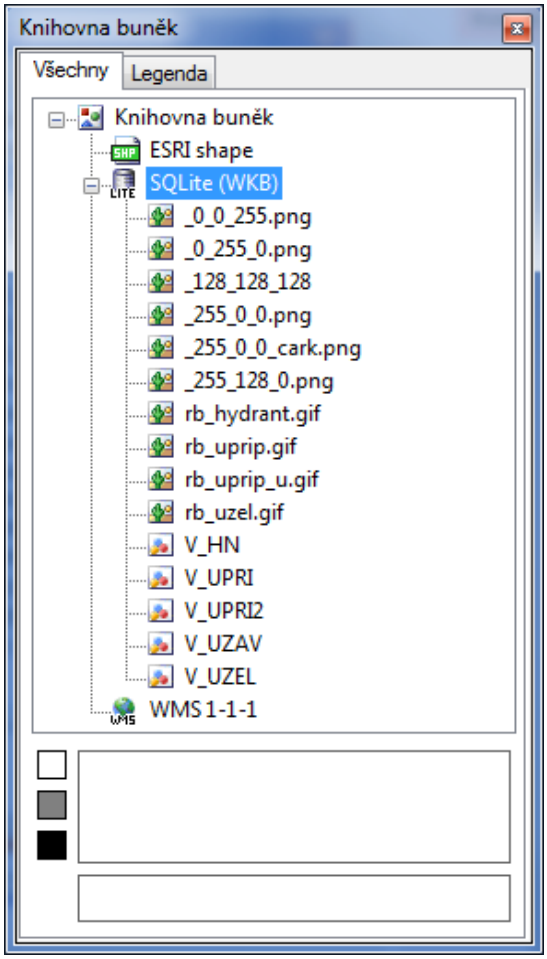
13 Legenda v Marushce

Pro snazší orientaci v jednotlivých prvcích, které jsou zobrazeny v mapové publikaci, nám poslouží legenda, kterou si ale musíme vytvořit. Základním předpokladem je existence rastrových buněk, které se pak budou zobrazovat jako jednotlivé položky legendy. Legendu můžeme mít v Marushce dvojího typu. ***Statickou***, kdy jednotlivé položky legendy budou zobrazeny vždy, kdy bude v daném výřezu mapového okna zobrazen alespoň jeden element formální vrstvy, ke které se položka legendy vztahuje. A druhou možností je vytvoření legendy ***dynamické***, kdy budou zobrazeny pouze relevantní položky legendy. To znamená, že položka legendy bude zobrazovaná pouze v případě, kdy se v zobrazeném výřezu mapového okna bude vyskytovat alespoň jeden prvek splňující příslušná kritéria.

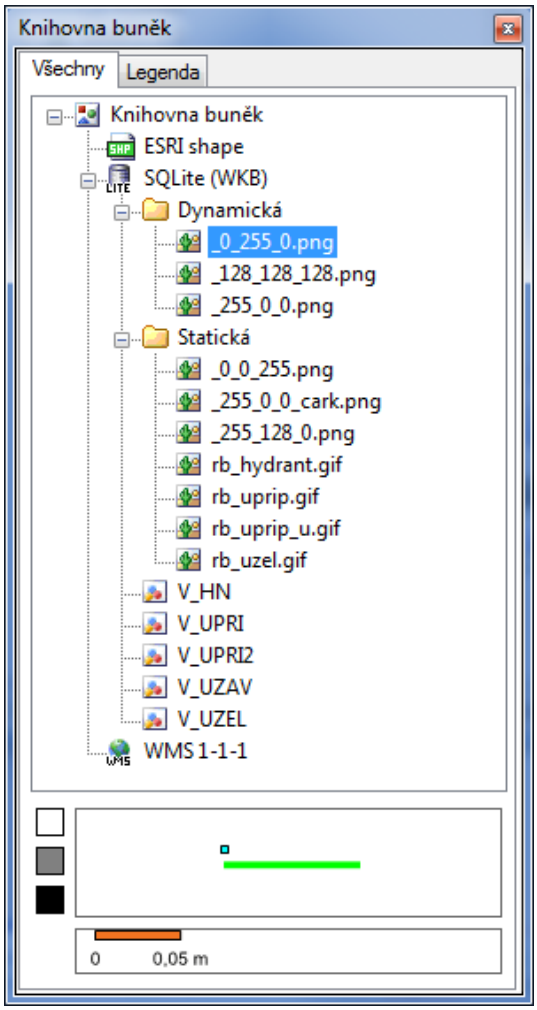
13.1 Příprava knihovny buněk

Obě varianty zobrazení legendy mají jednu společnou vlastnost a tou je nutnost existence rastrových buněk, které budou sloužit jako jednotlivá položka legendy. V našem tutoriálu máme připraveny malé obrázky, ze kterých si vytvoříme jednotlivé buňky. V knihovně buněk si tedy rovnou připravíme buňky pro statickou i pro dynamickou legendu.

V knihovně buněk otevřeme v kontextovém menu datového zdroje SQLite položku *Nová buňka – Rastrová buňka – Nová buňka ze souboru*. V dialogovém okně pro otevření souboru vybereme první buňku, kterou najdeme v souborech našeho tutoriálu (`C:\MarushkaExamplex\Tutorial\Soubory\rb_hydrant.gif`). Stejný postup zopakujeme pro další rastrové buňky pro statickou legendu (`rb_uprip.gif`, `rb_uprip_u.gif`, `rb_uzel.gif`, `_0_0_255.png`, `_255_0_0_cark.png` a `_255_128_0.png`) a pro tři rastrové buňky, které nám poslouží při tvorbě dynamické legendy (`_128_128_128.png`, `_255_0_0.png` a `_0_255_0.png`). Vytvoření takových obrázků si v našem tutoriálu nebudeme demonstrovat, protože přímo nesouvisí s prací v MarushkaDesignu a tyto obrázky můžeme vytvořit v libovolném grafickém editoru. Poté, co vytvoříme poslední rastrovou buňku, bude naše knihovna buněk vypadat takto:



Z obrázku je patrné, že nově vytvořené rastrové buňky nejsou moc přehledně uspořádány, tak v první fázi provedeme úpravu v jejich uspořádání. U buněk *_128_128_128.png, _255_0_0.png* a *_0_255_0.png* změníme vlastnost *Description* tak, abychom dostali řetězec následujícího tvaru: „*Legenda~Dynamická~původní název*“. U zbylých nově vytvořených rastrových buněk vložíme do vlastnosti *Description* text: „*Legenda~Statická~původní název*“. Po této úpravě bude naše dialogové okno s knihovnou buněk vypadat takto:



Uspořádání buněk už je přehlednější, použití vlnovkové notace je efektivním způsobem, jak si můžeme v našich datech udělat pořádek. V dalším kroku můžeme změnit vlastnost *Caption*, což je text, který se bude zobrazovat u položky legendy. V dolní části vidíme náhled buňky, podle které dokážeme odhadnout (až u statické legendy), k čemu byla konkrétní buňka vytvořena. O položkách, které přísluší dynamické legendě, se zmíníme v další části této kapitoly, kde pochopíme hlouběji jejich význam. Když budeme editovat položky podle pořadí, ve kterém jsou zobrazeny na obrázku výše, tak do vlastnosti *Caption* doplníme postupně tyto popisy:

Dynamická legenda:

_0_255_0.png – Přípojka dimenze 40

_128_128_128.png – Neurčená přípojka

_255_0_0.png – Přípojka dimenze 32

Statická legenda:

_0_0_255.png – Trasa vodovodního řadu

_255_0_0_cark_png – Trasa vodovodní přípojky – kreslená uživatelem

_255_128_0.png – Trasa vodovodní přípojky

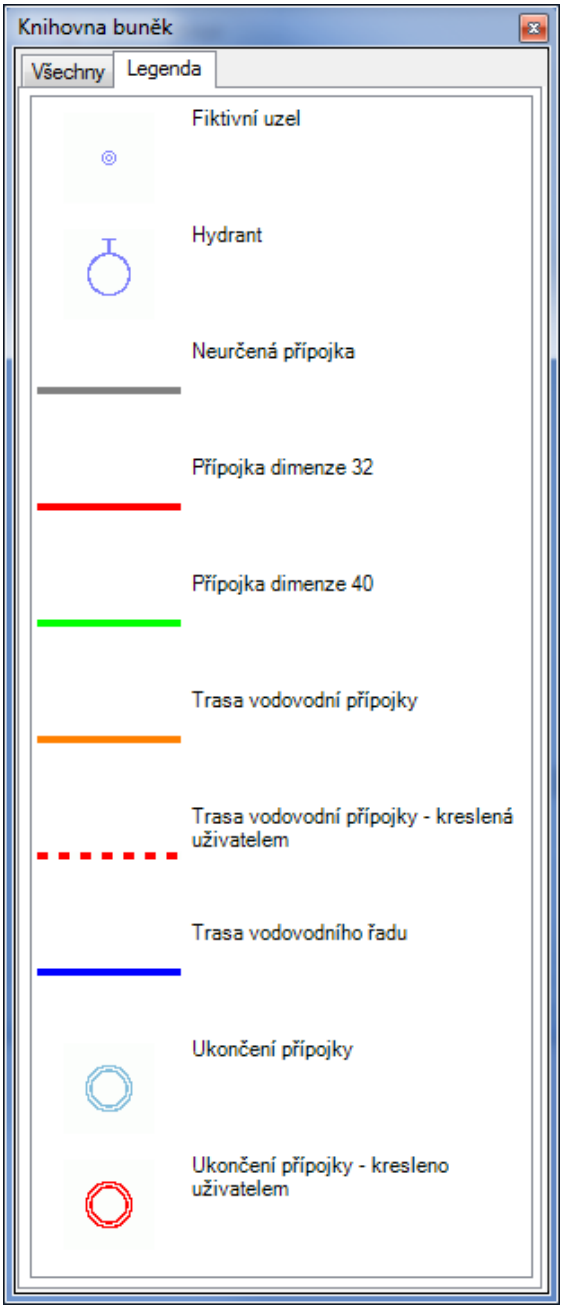
rb_hydrant.gif – Hydrant

rb_uprip.gif – Ukončení přípojky

rb_uprip_u.gif – Ukončení přípojky – kresleno uživatelem

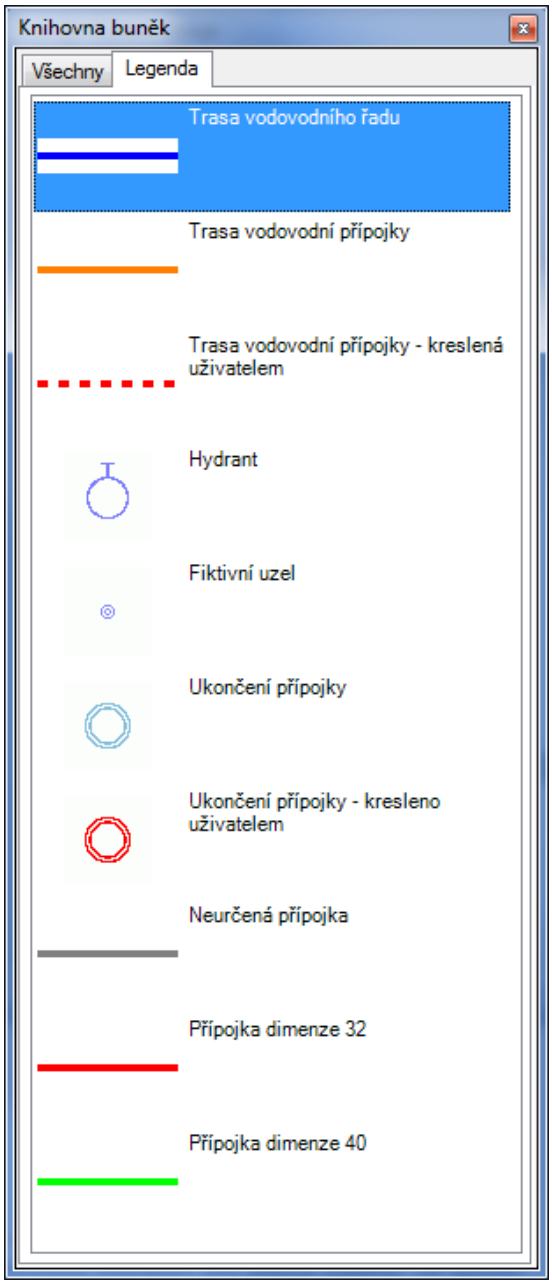
rb_uzel.gif – Fiktivní uzel

Ve vlastnosti *CellName* odebereme pro zjednodušení u všech buněk koncovku .png. Po této změně na kartě *Legenda* uvidíme tento stav:



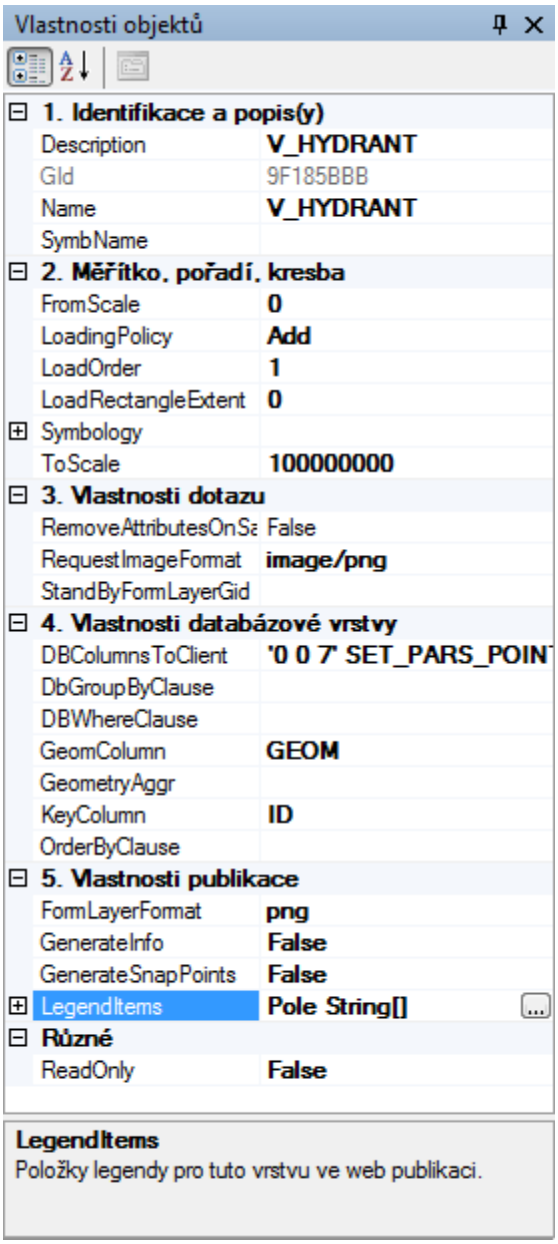
Z obrázku je patrné, že třídění legendy nám nemusí vyhovovat. Vlastnost třídění legendy je vlastností celého projektu, takže tuto vlastnost najdeme na kartě vlastnosti pro *Datové zdroje*. Úvodní nastavení je třídění podle parametru *Caption*. Tuto vlastnost ale nemusíme nijak měnit a Legendu si můžeme pomocí přetahování myší setřídit podle našich požadavků. Při prvním pokusu o přetažení položky legendy se nám objeví dialogové okno s informací, že aktuálně není dovoleno měnit řazení legendy. Pokud tuto otázku potvrdíme tlačítkem ano, tak můžeme pokračovat v ručním seřazování. Při ručním třídění můžeme opět využít multiselectu (pomocí klávesy CTRL nebo SHIFT a levého tlačítka myši) a třídit tak celou skupinu buněk. Naši legendu si setřídíme například takovýmto způsobem:

V horní části jsou položky statické legendy s prioritou zobrazení pro liniové objekty, ve spodní části vidíme položky dynamické legendy, kterou si vysvětlíme v další části této kapitoly.

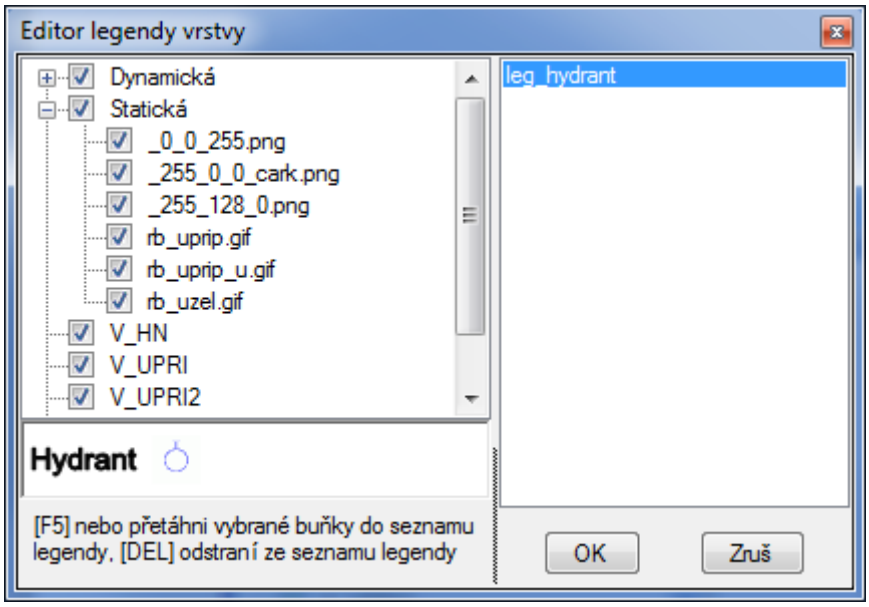


13.2 Vytvoření statické legendy

V první fázi si vytvoříme statickou legendu. Opustíme naši knihovnu buněk a přejdeme k vlastnostem formálních vrstev, ke kterým jsou jednotlivé položky legendy vázány. Ve vlastnostech formální vrstvy *V_HYDRANT* si najdeme vlastnost *LegendItems*:

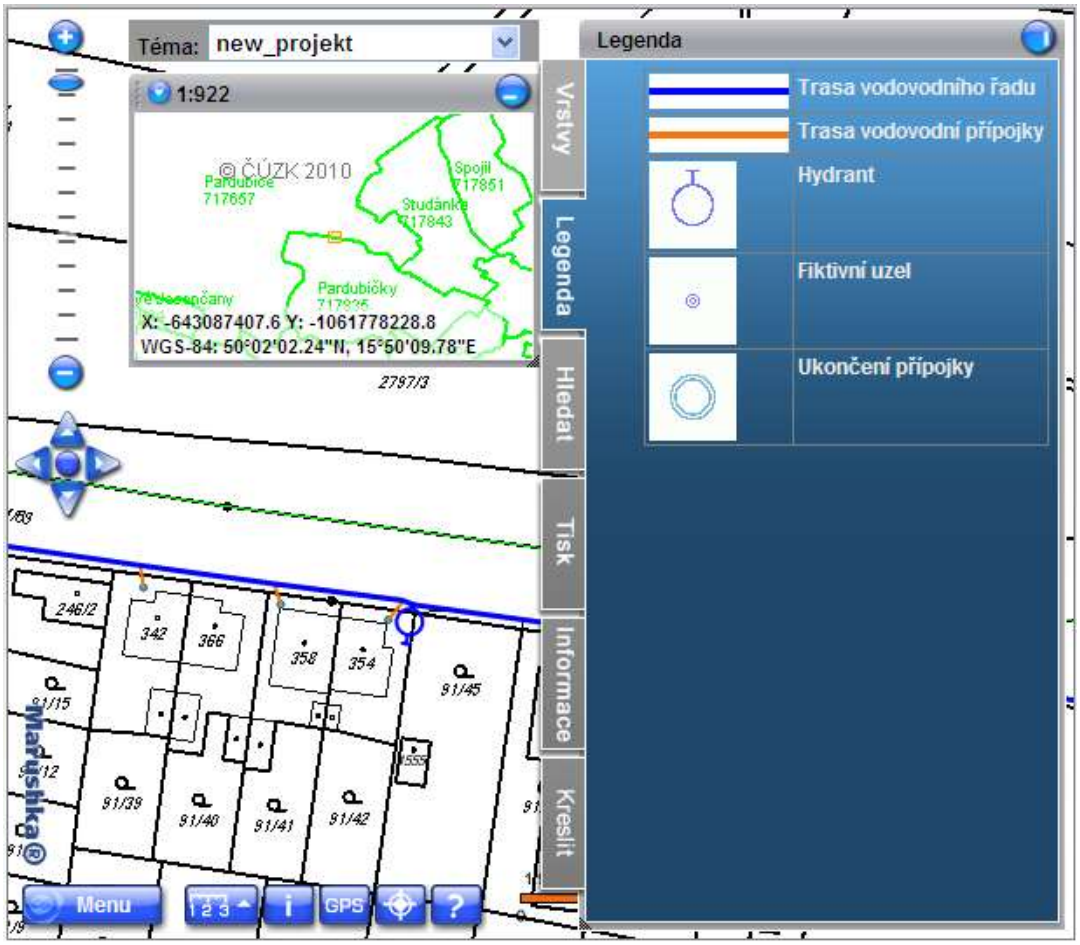


Po kliknutí na tlačítko se třemi tečkami se nám objeví dialogové okno s editorem legendy formální vrstvy. Najdeme si tedy položku pro hydrant a buď klávesou F5 nebo tažením myši v kombinaci s levým tlačítkem přesuneme příslušnou položku do pravé části dialogového okna:



Tímto jsme tedy vytvořili položku legendy pro příslušnou formální vrstvu a tato položka bude v prostředí lokálního WEB serveru zobrazena vždy, pokud v aktuálním výřezu mapového okna bude existovat alespoň jeden prvek náležející této formální vrstvě. Celý postup aplikujeme postupně na všechny naše formální vrstvy (kromě vrstvy *V_UZAVER*), a každé z nich přiřadíme právě jednu položku legendy. Obecně ale platí, že jedna formální vrstva může mít neomezené množství položek legendy.

Zobrazení právě vytvořené legendy si nyní můžeme opět vyzkoušet v prostředí lokálního WEB serveru. Pokud budeme prostorově v místě, kde jsou prvky ze všech formálních vrstev, tak by se nám zobrazila kompletní legenda. V opačném případě uvidíme jenom její část:



13.3 Dynamická legenda a tematizace

Pro použití dynamické legendy se přímo nabízí **tematizace**. Tematizací rozumíme zobrazení mapové kompozice jinou grafickou symbologií na základě databázových vlastností jednotlivých grafických elementů. V našem případě budeme chtít různou barvou zobrazit přípojky podle jejich dimenzí a k této tematizaci budeme chtít zobrazit příslušnou legendu, která nám umožní snadnější „přečtení“ odlišného grafického zobrazení.

13.3.1 Vytvoření tematizace

Pro vytvoření naší tematizace budeme potřebovat novou formální vrstvu. K tomu využijeme stávající formální vrstvu „*V_TRASA Přípojky*“, ze které si vytvoříme klon tak, jak jsme si už vysvětlili v předchozích kapitolách. Vlastnost *SymbName* u nově vytvořeného klonu formální vrstvy si přejmenujeme na „*Přípojky thema*“. V tomto klonu budeme chtít mít zobrazeny jak původní databázové přípojky, tak přípojky nově vzniklé v prostředí Marushky. Změníme tedy vlastnost *DBWhereClause* z podmínky: „*rc='Trasa vodovodní přípojky' and userdraw is null*“ na „*rc='Trasa vodovodní přípojky'*“.

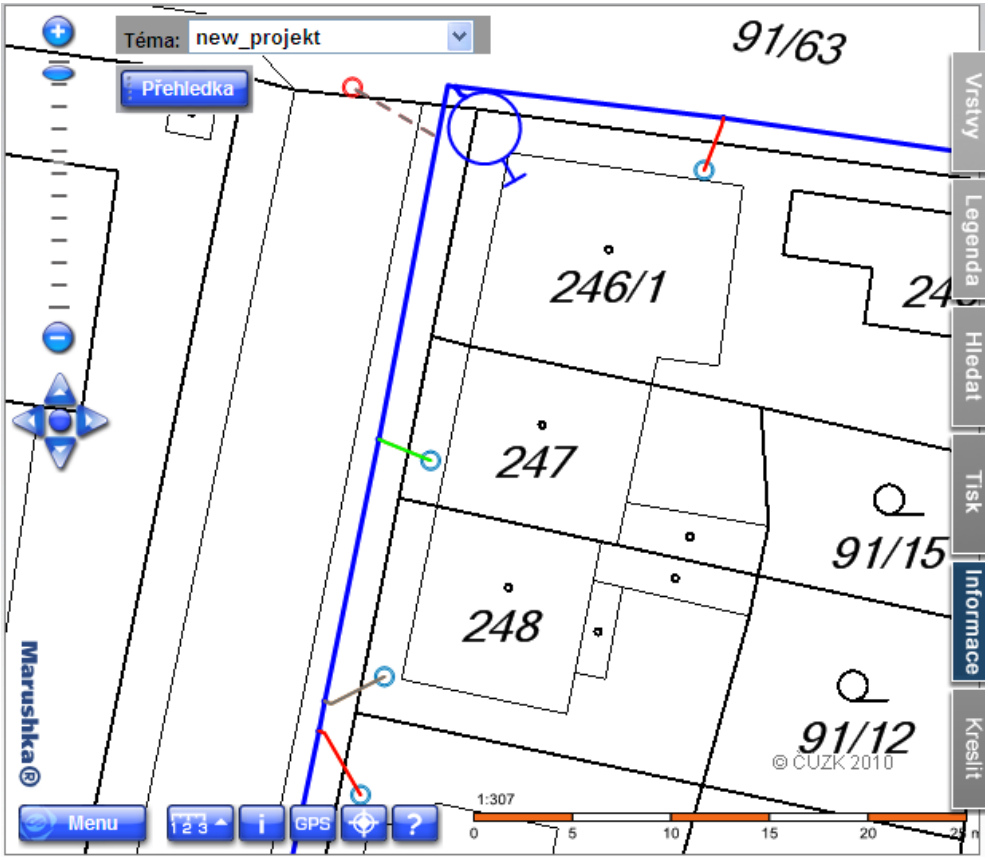
Ke změně barevné prezentace databázových grafických elementů využijeme pseudosloupců ve vlastnosti *DBColumnsToClient* v kombinaci s mírně pokročilou konstrukcí databázové fráze, ve které využijeme podmíněného větvení „case“. Celá tato vlastnost tak bude vypadat takto:

```
rc, 'ID: '||id SET_INFO_ICON_TEXT,case when dim_prip=32 then '255 255 0 0'
when dim_prip=40 then '255 0 255 0' else '255 128 128 128' end
SET_PARS_RGBCOLOR.
```

Abychom docílili toho, že se nám tematizované přípojky budou zobrazovat vždy nad databázovými prvky, tak ještě změníme vlastnost této formální vrstvy *LoadOrder* na hodnotu „2“. Tím bude zajištěno, že „přebarvená data“ budou zobrazena nad databázovými daty v původní symbologii.

Vytvoříme si novou publikační vrstvu s názvem „*Tematizace přípojek*“, do které náš nový klon formální vrstvy přetáhneme.

Pokud jsme postupovali poctivě při průchodu tímto tutoriálem a splnili úkol v kapitole Editace přípojky - Otestování výsledného editačního dotazu a opravdu vyplnili dimenze v několika přípojkách, pak můžeme otestovat i funkčnost naší tematizace. Výsledkem v lokálním WEB serveru (po ručním zapnutí nově vytvořené publikační vrstvy pro tematizaci) pak bude okno s takovou barevností:



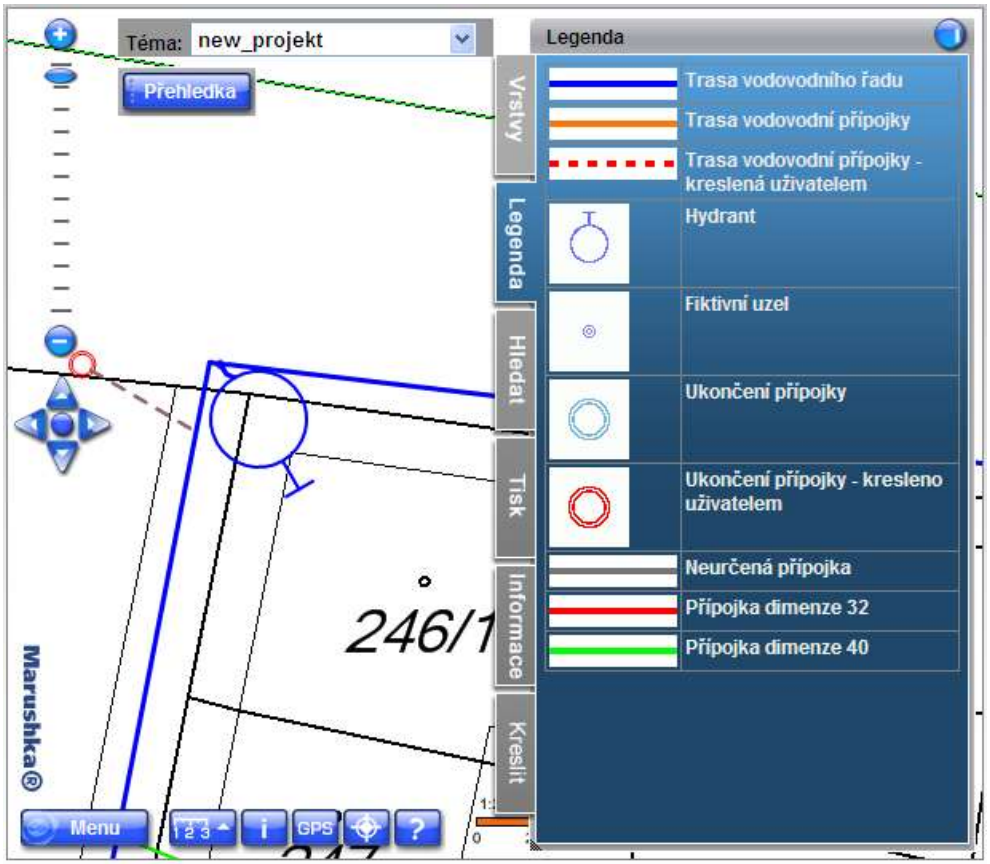
Na obrázku vidíme červené, šedé a zelené přípojky, z nichž styl čáry u přípojek, které byly zakresleny v prostředí Marushky, zůstává a takto zakreslené přípojky jsou tak i nadále vizuálně odlišitelné. V tomto okamžiku ale k tematizovaným přípojkám nemáme k dispozici legendu, proto je na čase, abychom si ji vytvořili.

13.3.2 Vytvoření dynamické legendy

Tento krok už je poměrně jednoduchý a je analogický k přebarvení přípojek pomocí pseudosloupce *SET_PARS_RGBCOLOR*. V tomto případě využijeme pseudosloupce *SET_LEG_ITEM*. Původní vlastnost *DBColumnsToClient* tedy rozšíříme do následujícího tvaru:

```
rc, 'ID: '||id SET_INFO_ICON_TEXT, case when dim_prip=32 then '255 255 0
0' when dim_prip=40 then '255 0 255 0' else '255 128 128 128' end
SET_PARS_RGBCOLOR, case when dim_prip=32 then '_255_0_0' when dim_prip=40
then '_0_255_0' else '_128_128_128' end SET_LEG_ITEM
```

V pseudosloupci *SET_PARS_RGBCOLOR* jsou jednotlivé parametry definovány ARGB kódem příslušných barev, v pseudosloupci *SET_LEG_ITEM* jde o vlastnost *CellName* jednotlivých buněk, které jsou definovány jako položky legendy. Proto bylo pro vlastnost *CellName* využito tohoto pojmenování, které je čitelné jako RGB kód dané barvy a jeho zápis je při dodržování této struktury poměrně intuitivní. Znovu tedy provedeme zkoušku zobrazení legendy v lokálním WEB serveru – opět po ručním zapnutí publikační vrstvy Tematizace přípojek:



Při pohybu v mapovém okně si pak můžeme ověřit, že i ve chvíli, kdy budeme mít zobrazenou formální vrstvu pro tematizaci přípojek, tak legenda bude proměnlivá podle toho, které typy atributů se nám v danou chvíli budou vyskytovat v mapovém okně.

14 Učíme Marushku zpívat a tancovat

V rámci tohoto návodu jsme si vyzkoušeli veškerou základní funkčnost pro tvorbu webových mapových kompozic v prostředí MarushkaDesignu. S Marushkou jsme si prošli její první kroky, naučili jsme ji číst, psát, kreslit a i jinak pracovat s daty. Zpívat a tancovat ji opravdu nenaučíme, ale je jenom na nás, jakým způsobem budeme pracovat s dostupnou datovou základnou a nakolik budeme schopní data využít. Možnosti jsou opravdu široké a celý tento dokument je jenom základním průvodcem funkcemi Marushky. Neukázali jsme si všechno, celý průvodce je zjednodušeným přehledem funkčnosti jednotlivých komponent. Každý projekt lze neustále rozvíjet a zdokonalovat, když budeme dále přemýšlet nad dalším rozšířením jednotlivých funkcí. Základní práci v MarushkaDesignu tedy máme úspěšně za sebou.

Výsledek našeho snažení pak můžeme přenést na oficiální webový server a naši mapovou kompozici pak můžeme zveřejnit (ať už interně či úplně veřejně). Principy tohoto zveřejňování už ale sahají za rámec této příručky, více se o této problematice můžeme dočíst v oficiálním manuálu, který je součástí MarushkaDesignu.

A kam dál? Marushka je stále živá a její možnosti jsou nekonečné, tak tedy závěrem ještě pár dalších námětů, co bychom si mohli v našem testovacím projektu sami vyzkoušet.

14.1 Náměty k samostatným cvičením

- Samostatná úprava kreslícího dotazu pro kreslení přípojky, kdy rovnou při kreslení budeme chtít vkládat dimenzi přípojky.
- Vrátíme se k buňkám uzávěrů, které jsme vypustili z publikace. Zkusíme je tam vrátit s tím, že budeme chtít zvýraznit zavřené uzávěry. Databázovou tabulku V_UZAVER rozšíříme o sloupec STAV. Tento sloupec budeme umět editovat v prostředí WEB serveru (vytvoření editačního dotazu se statickým číselníkem). Následně vytvoříme jednoduchou tematizaci, případně formální vrstvu, která bude mít speciální grafickou symbologii pro zavřené uzávěry. Výsledkem bude zobrazení vypnutých uzávěrů.
- Ve spolupráci s oficiálním manuálem si můžeme vytvořit sestavu (obecný dotaz), který nám zobrazí html stránku např. s přehledem zákazníků v konkrétním městě. V tomto případě už si tu jenom naznačíme cestu, jakou bychom postupovali, protože takový dotaz už je poměrně uživatelsky náročný. Kromě znalosti databázové prostředí vyžaduje i obecné znalosti HTML kódu, aby se výsledek dotazu zobrazoval korektně. V knihovně dotazů si vytvoříme nový dotaz – *Obecné informace*. Ve vlastnosti *SqlStmtTemplate* si vytvoříme dotaz, který nám vrátí potřebný seznam (jedná se o dotaz, který není propojen s žádným grafickým elementem) a ve vlastnosti *ResultTemplate* bychom si vytvořili HTML šablonu pro zobrazení výsledku tohoto dotazu. HTML šablonu můžeme využít i v jiných případech, kdy výsledek dotazu zobrazujeme v novém okně.

V tomto okamžiku si můžeme poblahopřát, že jsme na cestách s Marushkou dospěli až do této fáze a dokázali jsme si vytvořit celý projekt. Další možnosti záleží už jenom na naší fantazii.