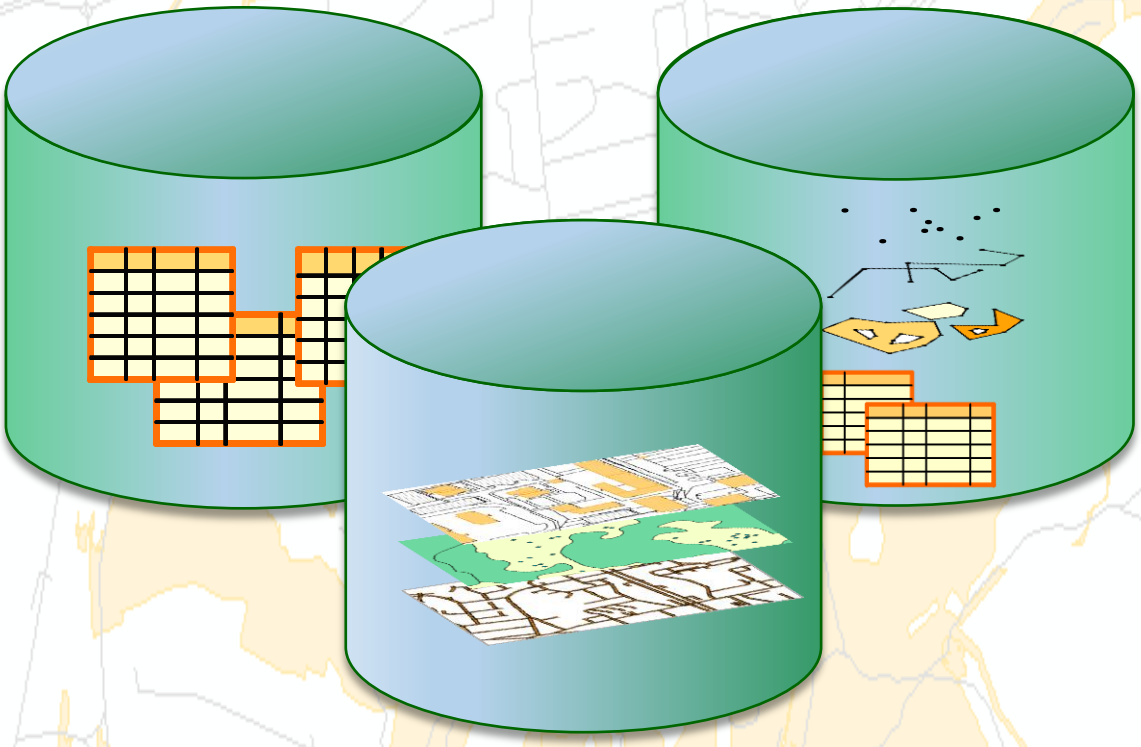
DATABÁZOVÉ SYSTÉMY V GIS

Návody na cvičenia

Renata Ďuračiová, Dušan Cibulka



STU
SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE

DATABÁZOVÉ SYSTÉMY V GIS

Návody na cvičenia

Renata Ďuračiová, Dušan Cibulka

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
2015

Recenzenti: Ing. Róbert Fencík, PhD.
Mgr. Ivan Škultéty

Ing. Renata Ďuračiová, PhD., Ing. Dušan Cibulka, PhD.

DATABÁZOVÉ SYSTÉMY V GIS
Návody na cvičenia

Vydala Slovenská technická univerzita v Bratislave v Nakladateľstve STU, Bratislava, Vazovova 5, v roku 2015.

Edícia skrípt

Rozsah 135 strán, 161 obrázkov, 7 tabuliek, 7,906 AH, 8,128 VH, 1. vydanie,
edičné číslo 5880, vydané v elektronickej forme;
umiestnenie na http://www.svf.stuba.sk

Schválila Edičná rada Stavebnej fakulty STU v Bratislave.

85 – 255 – 2015

ISBN 978-80-227-4502-4

Obsah

Zoznam použitých značiek a skratiek

Značka / skratka	**Charakteristika**
ACID	*Atomicity, Consistency, Isolation, Durability* (atomickosť, konzistencia, izolovanosť a trvácnosť)
ANSI	*American National Standards Institute* (Americký normalizačný úrad)
API	*Application Programming Interface* (aplikačné programové rozhranie)
BLOB	*Binary Large Object* (veľký binárny objekt)
BSD	*Berkeley Software Distribution*
CASE	*Computer Aided Software Engineering* (počítačovo podporované softvérové inžinierstvo)
CSW	*Catalog Service for the Web* (webová katalógová služba)
DBMS	*Database Management System* (systém riadenia databázy)
DCL	*Data Control Language* (jazyk na riadenie prístupu k dátam)
DDL	*Data Definition Language* (jazyk na definíciu dát)
DML	*Data Manipulation Language* (jazyk na manipuláciu s dátami)
E-R (diagram)	*Entity-Relationship (diagram)* (entitno-relačný (diagram))
EPL	*Eclipse Public Licence*
EPSG	*European Petroleum Survey Group*
ESRI	*Environmental Systems Research Institute*
FK	*Foreign Key* (cudzí kľúč)
GIS	*Geographic Information System* (geografický informačný systém)
GiST	*Generalized Search Tree*
GML	*Geography Markup Language* (geografický značkovací jazyk)
GNU	*GNU's Not Unix*

Značka / skratka	**Charakteristika**
GPL	*General Public License*
IEC	*International Electrotechnical Commission* (Medzinárodná elektrotechnická komisia)
IP	*Internet Protocol*
ISO	*International Organization for Standardization* (Medzinárodná organizácia pre normalizáciu)
JDBC	*Java Database Connectivity* (databázové rozhranie na platforme Java)
JPEG	*Joint Photographic Experts Group*
KN	Kataster nehnuteľností
MVCC	*Multi-Version Concurrency Control*
ODBC	*Open Database Connectivity* (otvorené databázové rozhranie)
ODbL	*Open Database Licence*
OGC	*Open Geospatial Consortium*
OLE DB	*Object Linking and Embedding Database*
OpenFTS	*Open Source Full Text Search engine*
PDA	*Personal Digital Assistant*
PDF	*Portable Document Format*
PK	*Primary Key* (primárny kľúč)
PL/SQL	*Procedural Language/Structured Query Language* (procedurálne rozšírenie SQL)
PNG	*Portable Network Graphics*
PSM	*Persistent Stored Module* (trvalo uložený modul)
SDE	*Spatial Database Engine*
SEQUEL	*Structured English Query Language* (štruktúrovaný anglický dopytovací

Značka / skratka	**Charakteristika**
	jazyk)
SFA	*Simple Feature Access* (jednoduchý prístup k objektom)
S–JTSK	Systém jednotnej trigonometrickej siete katastrálnej
SPIT	*Shapefile to PostGIS Import Tool*
STN	Slovenská technická norma
SQL	*Structured Query Language* (štruktúrovaný dopytovací jazyk)
SQL/MM	*SQL Multimedia* (rozšírenie SQL pre multimediálne dáta)
SQL/PSM	*SQL/Persistent Stored Modules* (rozšírenie SQL o prácu s trvalo uloženými modulmi (procedúrami))
SRDB	Systém riadenia databázy
SRID	*Spatial Reference System Identifier* (identifikátor referenčného súradnicového systému)
TCL	*Transaction Control Language* (jazyk na riadenie transakcií)
TIFF	*Tagged Image File Format*
TIN	*Triangulated Irregular Network* (nepravidelná trojuholníková sieť)
T-SQL	*Transact-SQL* (rozšírenie SQL o spracovanie transakcií)
URL	*Uniform Resource Locator* (jednotný lokalizátor zdroja)
XML	*eXtensible Markup Language* (rozšíriteľný značkovací jazyk)
WKB	*Well-Known Binary* (binárna reprezentácia objektov)
WKT	*Well-Known Text* (textová reprezentácia objektov)

Úvod

Elektronické skriptá Databázové systémy v GIS (Návody na cvičenia) sú určené predovšetkým študentom 2. stupňa študijného programu Geodézia a kartografia (GaK) na Stavebnej fakulte Slovenskej technickej univerzity v Bratislave, ale aj všetkým záujemcom o problematiku datatabázových systémov, správy priestorových informácií a geografických informačných systémov (GIS). Cieľ skrípt je poskytnúť čitateľom informácie z oblasti databázových systémov v súvislosti s ich využitím v GIS a tiež umožniť praktickú aplikáciu poznatkov získaných v rámci predmetu DSGIS pri riešení zadaných úloh (zadaní). Preto sú skriptá zamerané nielen na širokú problematiku databázových systémov, ale aj na jej rozšírenie o charakteristiku priestorových dát a ich praktické použitie v databázových systémoch a GIS.

Skriptá sú členené do piatich hlavných kapitol a záverečnej kapitoly, v ktorej sú uvedené výsledky riešených úloh. Každá hlavná kapitola obsahuje úvodnú teoretickú časť, ktorá je nazvaná „Teoretické minimum“ a praktickú časť, ktorú tvorí zadanie úloh a prípadne aj postup ich riešenia. Zaradenie podkapitoly „Teoretické minimum“ umožňuje samostatné použitie elektronických skrípt, pretože obsahuje najdôležitejšie informácie, ktoré sú potrebné pri riešení úloh zadaných v druhej časti každej kapitoly. „Teoretické minimum“ zahŕňa len stručné informácie, podrobnejšie je problematika opísaná v skriptách (Ďuračiová, 2014) alebo v prednáškach predmetu DSGIS.

V prvej kapitole skrípt sú opísané základné princípy a charakteristiky databázových systémov a prvé zadanie je zamerané na inštaláciu softvérového prostredia databázového systému, konkrétne systému PostgreSQL. Druhá kapitola obsahuje stručné vysvetlenie štruktúry relačných databáz, ich matematických základov a tiež charakteristiku funkcionality dopytovacieho jazyka SQL *(Structured Query Language)*, ktorý už dlhodobo predstavuje štandard v oblasti relačných a objektovo relačných databáz. Zadania z tejto tematickej oblasti sú zamerané na zvládnutie tvorby jednoduchých dopytov v jazyku SQL, najprv z hľadiska používania databázy a neskôr aj z pohľadu jej tvorby a modifikácie. Tretia kapitola obsahuje problematiku rozšírenia databázových systémov o možnosti spracovania a analýzy priestorových dát. Úlohy zadania sú orientované na zvládnutie tvorby priestorových dopytov do databázy s využitím rôznych funkcií voľne dostupného modulu PostGIS. Štvrtá kapitola dopĺňa problematiku databázových systémov a priestorových dát o možnosť importu

a exportu dát z a do databázy a tiež približuje problematiku vizualizácie priestorových dát v prostredí GIS, konkrétne v open source GIS softvéroch QGIS a uDig. Piata kapitola je venovaná navrhovaniu a modelovaniu databáz pomocou entitno-relačných diagramov alebo prostredníctvom štandardizovaného modelovacieho jazyka UML *(Unified Modeling Language)* a obsahuje aj základné princípy a postupy logického návrhu relačných databáz. Táto kapitola približuje problematiku databázových systémov z pohľadu návrhára alebo tvorcu databáz. Zadanie je zamerané na samostatný návrh a vytvorenie jednoduchej databázy z vybranej domény, ktorá bude spĺňať stanovené požiadavky.

Kapitoly skript na seba navzájom nadväzujú, ale zároveň sú vytvorené tak, aby zadané úlohy bolo možné riešiť aj samostatne, a tým sa oboznámiť napríklad len s inštaláciou databázového systému (kapitola **1**), základmi jazyka SQL (kapitola **2**), problematikou priestorových databáz (kapitola **3**), importom, exportom a vizualizáciou priestorových dát (kapitola **4)** alebo logickým návrhom databáz (kapitola **5**). Výhodou skript je, že riešenie úloh zo zadaní si nevyžaduje inštaláciu žiadneho proprietárneho softvéru a môže byť realizované aj prostredníctvom dostupných open source softvérov. Príklady a úlohy v skriptách sú pre lepšiu názornosť v oblasti geodézie a kartografie tematicky zamerané na problematiku dát katastra nehnuteľností (KN) alebo vektorových priestorových dát s možnosťou ich vizualizácie v prostredí GIS. Dopyty v jazyku SQL sú kvôli možnosti rýchleho osvojenia si základných princípov jazyka demonštrované na veľmi jednoduchých príkladoch, orientovaných len na konkrétny jav alebo funkciu, ktoré sú realizované vo výrazne zjednodušenej modelovej databáze. Takto postavené príklady umožňujú osvojenie si základných princípov databázových systémov a pravidiel jazyka SQL dostupnou formou aj v relatívne krátkom období (časová dotácia predmetu DSGIS je dve hodiny prednášok a dve hodiny cvičení týždenne počas jedného semestra). Modelové databázy a súbory priestorových dát, ktoré tvoria prílohy týchto skript, sú dostupné na: `ftp://147.175.19.15/dsgis/`.

Elektronické skriptá svojimi návrhmi obsahovo obohatili aj ich recenzenti, Ing. Róbert Fencík, PhD. a Mgr. Ivan Škultéty, za čo im úprimne ďakujeme. Za pomoc, podporu a vytvorenie podnentného prostredia ďakujeme svojim blízkym, kolegom a dobrým priateľom.

Bratislava, október 2014 autori

Konvencie a poznámky

V texte skrípt sú dodržané nasledujúce konvencie:

- novodefinované alebo prvýkrát použité pojmy sú vyznačené **tučným písmom**,
- anglické výrazy sú uvedené *šikmým písmom (kurzívou) v zátvorkách ( ) za konkrétnym pojmom*,
- názvy tabuliek, atribútov a ich konkrétnych hodnôt sú uvedené fontom `Courier New`,
- kľúčové slová jazyka SQL sú v dopytoch písané **`VEĽKÝM TUČNÝM PÍSMOM fontu Courier New`**,
- voliteľné povinné nastavenia v dopytoch sú označené zátvorkami { },
- voliteľné nepovinné nastavenia v dopytoch sú označené hranatými zátvorkami [ ].

Text skrípt bol vytvorený v softvérovom prostredí Microsoft Word a na zhotovovanie obrázkov, tabuliek, diagramov a schém uvedených v skriptách boli využité softvérové prostredia Microsoft Word, Microsoft PowerPoint, Microsoft Visio, Microsoft Access, Select Architect, StarUML, ArcGIS, QGIS (QuantumGIS) a uDig. Priestorové dáta použité v zadaniach úloh (v prílohe) a na niektorých obrázkoch pochádzajú z projektu OpenStretMap, ich autori sú „OpenStretMap a prispievatelia“ a sú poskytované pod licenciou ODbL *(Open Database Licence)*. V texte skrípt sú použité aj ďalšie názvy a obchodné známky spoločností Microsoft, Oracle Corporation, IBM, Sybase, ESRI, Sun Microsystems a iných.

Zoznam zadaní úloh

1 Databázový systém a inštalácia systému riadenia databázy

(1. cvičenie)

Teoretické minimum:

1.1 Dáta, databáza, databázový systém, systém riadenia databázy

Dáta[1] sú v STN 73 0401-3 definované ako „opakovateľné predstavenie informácie formalizovaným spôsobom vhodným na komunikáciu, interpretáciu alebo spracovanie“. Dáta sú nespracované fakty, spracovaním dostanú požadovanú štruktúru a význam a stávajú sa z nich **informácie**. Dáta, ktoré poskytujú informácie a poznatky vztiahnuté k určitým miestam v priestore, sa nazývajú **priestorové dáta** (kapitola **3.1**) (Ďuračiová, 2014).

Databáza je „organizovaná a integrovaná zbierka dát vzťahujúca sa na danú tému alebo oblasť, uložená v pamäti počítača a usporiadaná tak, aby sa mohla používať v požadovaných aplikáciách“ (STN 73 0401-3). Podľa (Delikát, 2006) je databáza zdieľaná integrovaná počítačová štruktúra, ktorá zahŕňa dáta a metadáta (dáta o dátach). Predstavuje množinu vzájomne súvisiacich dát uloženú na pamäťovom médiu, usporiadanú a organizovanú tak, aby podporila vykonávanie špecifických požiadaviek.

Databázový systém (Obr. 1.1a) je systém, ktorý pozostáva z databázy a zo **systému riadenia databázy** (SRDB) (angl. *Database Management System* (DBMS)).

SRDB je súbor (skupina) programov na spracovanie databázovej štruktúry a riadenie prístupu k databáze. SRDB poskytuje pohodlné a bezpečné multipoužívateľské prostredie na efektívnu manipuláciu aj s veľkým objemom dát. Je to softvér, ktorý komunikuje s databázou, databázovou aplikáciou a prostredníctvom nej aj s používateľom databázového systému[2] (Obr. 1.1b). SRDB podporuje dopytovací jazyk, ktorý obsahuje príslušné príkazy na zabezpečenie základnej funkcionality systému. Väčšinou ide o dopytovací jazyk SQL

[1] Výraz **dáta** nemá jednotné číslo, preto v jednotnom čísle používame slovo **údaj**. V niektorých zdrojoch literatúry sú preto výrazy údaje a dáta používané ako synonymá, aj keď výraz údaje má všeobecnejší význam ako výraz dáta, pretože údaje môžu byť aj v inej ako digitálnej forme (analógovej).

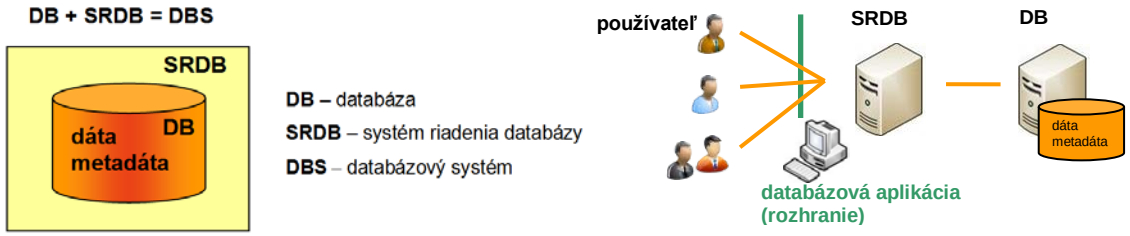
[2] V niektorých zdrojoch literatúry sú pojmy databázový systém a SRDB považované za synonymá.

(Structured Query Language) (kapitola **2.2**), ktorý predstavuje v súčasnosti štandard na tvorbu dopytov v relačných a objektovo relačných databázových systémoch.

SRDB umožňuje napr.:

- **vytvoriť** databázu na konkrétnom médiu,
- **definovať** dátové typy[3], štruktúry a konštrukcie pre dáta uložené v databáze,
- **manipulovať** s dátami (vytvárať a aktualizovať dáta, selektovať dáta na základe konkrétnych požiadaviek),
- **zdieľať** databázu medzi viacerými používateľmi (riadením prístupu k databáze prostredníctvom nastavenia prístupových práv alebo riadením vzájomných konfliktov pri aktualizácii databázy viacerými používateľmi súčasne),
- **chrániť** dáta proti neautorizovaným a chybným prístupom a haváriám systému (napr. zamedzením neoprávnených prístupov a modifikáciám databázy, zálohovaním databázy a pod.),
- **spravovať transakcie**, ktoré vykonajú v databáze niekoľko zmien súčasne v rámci jednej transakcie.

a) b)



Obr. 1.1. a) Databázový systém, b) komunikácia používateľov s databázovým systémom (Ďuračiová, 2014)

1.2 Prehľad najpoužívanejších systémov riadenia databázy

Softvérové prostredia databázových systémov (konkrétne SRDB) môžeme podľa dostupnosti ich zdrojového kódu rozdeliť na proprietárne (s uzavretým (nedostupným) zdrojovým kódom) a open source (s otvoreným zdrojovým kódom, väčšinou voľne dostupné).

[3] **Dátový typ** určuje v databáze typ hodnôt, ktoré môže atribút nadobúdať. Medzi najpoužívanejšie dátové typy v relačných databázach patria reťazcové, numerické, booleovské, dátumové a časové dátové typy, ale napr. aj dátový typ na ukladanie veľkých binárnych objektov BLOB (*Binary Large OBject*) alebo BINARY. Na prácu s priestorovými dátami sa používajú špeciálne priestorové dátové typy (kapitola **3.1**).

Medzi najznámejšie open source SRDB v súčasnosti patria **PostgreSQL** (kapitola **1.2.1**) a **MySQL** (kapitola **1.2.2**), medzi proprietárne SRDB najmä **Oracle** (kapitola **1.2.3**), **SQL Server** (kapitola **1.2.4**), ale aj napr. databázové systémy **Sybase** (vyvíjaný rovnomennou spoločnosťou), **DB2** a **Informix** (v súčasnosti obidva vyvíjané spoločnosťou IBM).

Menšie databázové aplikácie môžu byť realizované aj ako personálne (jednopoužívateľské) alebo desktopové databázy, napr. v softvérovom prostredí **Microsoft Access** alebo **Visual FoxPro**, aj keď v týchto prípadoch v skutočnosti nejde o kompletné plnohodnotné databázové systémy.

1.2.1 PostgreSQL

PostgreSQL (http://www.postgresql.org[4]) je objektovo relačný (kapitola **2.1**) open source databázový systém sprístupnený pod vlastnou licenciou PostgreSQL Licence (PostgreSQL) (http://opensource.org/licenses/postgresql). Databázový systém PostgreSQL bol pôvodne vyvinutý na univerzite Berkeley v Kalifornii a jeho pôvodný názov bol Postgres. (nadväzoval na pôvodný projekt Ingres). Je podporovaný najčastejšie používanými operačnými systémami ako Linux, UNIX a Windows. Je plne ACID[5] *(Atomicity, Consistency, Isolation, Durability)* kompatibilný, čo znamená, že vykonané transakcie spĺňajú všetky potrebné kritériá, ktorými sú: atomickosť, konzistencia, izolovanosť a trvácnosť.

PostgreSQL a jeho implementácia SQL spĺňa štandard SQL:2008. PostgreSQL plne podporuje cudzie kľúče, spájanie tabuliek, pohľady, triggre (kapitola **2.2.2**) a uložené procedúry (v rôznych jazykoch, napr. C, Perl, Python a pod.). Podporuje aj poddopyty,

[4] citované 26. 9. 2013

[5] **Kritériá ACID** definujú teoretické požiadavky na správanie sa systémov, v ktorých je dôležité bezpečné spracovanie dát. Tieto požiadavky musia byť vždy splnené napr. v prípade vykonávania transakcií. Jednotlivé písmená skratky ACID znamenajú:

A – *Atomicity* (atomicita) – t. j. príkazy v transakcii sa buď vykonajú všetky spolu alebo sa nevykoná žiadny z nich.

C – *Consistency* (konzistencia) – pred a po dokončení transakcie musia byť dáta konzistentné.

I – *Isolation* (izolovanosť) – transakcia je izolovaná od okolia, operácie vykonávané v rámci transakcie sú pre okolie systému neviditeľné (t. j. transakcia je nezávislá aj od iných transakcií).

D – *Durability* (trvácnosť) – ak bola transakcia potvrdená, všetky zmeny dát sú trvalé (t. j. ostávajú v platnosti aj v prípade ukončenia chodu systému alebo v prípade havárie databázového systému).

vrátane vnorených výberov v klauzule FROM (kapitola **2.2.1**), ako aj veľa dátových typov definovaných v štandarde SQL:2008, napr. INTEGER (celé číslo), NUMERIC (formátované číslo), BOOLEAN (logická pravdivostná hodnota), CHAR (znakový reťazec pevnej dĺžky), VARCHAR (znakový reťazec premenlivej dĺžky), DATE (dátum), INTERVAL (interval) a TIMESTAMP (časová značka). Zároveň umožňuje uloženie veľkých binárnych objektov ako sú obrázky, zvuky alebo video. Má natívne programové rozhrania pre programovacie jazyky a platformy C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC *(Open Database Connectivity)* a iné.

PostgreSQL podporuje sofistikované nástroje, ako sú napríklad MVCC *(Multi-Version Concurrency Control)*, ktorý umožňuje zvýšiť výkonnosť databázy v multipoužívateľskom prostredí. Medzi ďalšie vlastnosti patrí podpora bodov obnoviteľnosti, tabuľkových priestorov, asynchrónnej replikácie, vnorených transakcií, záloh alebo optimalizácie dopytov (http://www.postgresql.org/about/). Podporuje medzinárodné znakové sady, viacbajtové kódovanie znakov, prípadne aj kódovanie Unicode. Je vysoko škálovateľný čo sa týka množstva údajov, ale aj paralelných používateľov, ktorých vie obslúžiť. V súčasnosti existujú aktívne PostgreSQL systémy v produkčných prostrediach, ktoré spravujú viac ako 4 TB dát. Niektoré základné všeobecné limity (obmedzenia) PostgreSQL sú uvedené v tabuľke (Tabuľka. 1.1). Zatiaľ čo PostgreSQL tvorí plne relačný systémový katalóg, ktorý podporuje viac schém v rámci jednej databázy, jeho katalóg je dostupný aj prostredníctvom tzv. informačnej schémy[6] (tak, ako je to definované v SQL štandarde).

PostgreSQL podporuje aj pokročilý systém indexovania[7] GiST *(Generalized Search Tree)*, ktorý prináša širokú škálu rôznych triediacich a vyhľadávacích algoritmov ako napr. B-stromy *(B-tree)*, B^+-stromy *(B^+-tree)*, R-stromy *(R-tree)* a mnoho ďalších. Tiež poskytuje rozhranie, ktoré umožňuje jednak vytváranie vlastných dátových typov, a tiež rozšíriteľné metódy dopytovania, ktorými možno v nich vyhľadávať. GiST ponúka flexibilitu pri určovaní čo ukladať, ako to skladovať a tiež schopnosť definovať nové spôsoby vyhľadávania dát.

[6] **Schéma** relačnej databázy je špecifická množina dát, ktorá opisuje dátový model databázy alebo jej časti. Zahŕňa tabuľky a všetky ostatné konštrukcie patriace k tej istej databáze. Norma ISO/IEC 9075-11: Information and Definition Schemas (SQL/Schemata) špecifikuje dve schémy – informačnú a definičnú. Informačná schéma obsahuje mená databázových objektov (tabuliek, stĺpcov, pohľadov, procedúr a pod.) a môže ich poskytnúť aplikácii. V PostgreSQL tvorí informačná schéma automaticky súčasť každej databázy. Definičná schéma poskytuje dátový model na podporu informačnej schémy a jej pochopenie.

[7] **Index** je dátová štruktúra, ktorá umožňuje SRDB rýchlejšie lokalizovať konkrétne záznamy v súbore a tým zrýchliť odozvu na používateľské dopyty (Conolly a kol., 2009). Indexy sú tabuľkové štruktúry, ktoré v jednom stĺpci obsahujú hodnotu kľúča a v druhom ukazovateľ na fyzické umiestnenie riadku tabuľky s touto hodnotou kľúča.

GiST zároveň slúži ako základ pre mnoho verejných projektov, ktoré využívajú PostgreSQL. Patrí medzi ne napr. OpenFTS *(Open Source Full Text Search engine)*, ktorý poskytuje online indexovanie dát a hodnotenie relevantnosti pre databázové vyhľadávanie, alebo aj rozšírenie PostGIS, ktoré je určené na prácu s priestorovými dátami (kapitola **3.1**).

Tabuľka. 1.1. Parametre PostgreSQL

Limit	Hodnota
Maximálna veľkosť databázy	neobmedzená
Maximálna veľkosť tabuľky	32 TB
Maximálna veľkosť riadka	1.6 TB
Maximálna veľkosť poľa	1 GB
Počet riadkov na tabuľku	neobmedzený
Počet stĺpcov na tabuľku	250 – 1600 v závislosti od typu stĺpcov
Počet indexov na tabuľku	neobmedzený

V oblasti GIS má najväčší význam práve voľne dostupný modul PostGIS, ktorý je vyvíjaný od roku 2001. Jeho zdrojový kód je prístupný na použitie v nekomerčnej, ale aj v komerčnej sfére. PostGIS je projekt, ktorý pridáva k PostgreSQL podporu pre geografické objekty, vďaka čomu môže byť databáza v PostgreSQL použitá ako priestorová databáza pre GIS podobne ako napr. SDE *(Spatial Database Engine)* od spoločnosti ESRI *(Environmental Systems Research Institute)* alebo *Spatial extension*, priestorové rozšírenie od spoločnosti Oracle. PostGIS spĺňa príslušné štandardy konzorcia OGC *(Open Geospatial Consortium)*[8], konkrétne *OpenGIS Implementation Standard for Geographic information – Simple feature access*[9] a *Simple Features – SQL – Types and Functions 1.1*. Podporuje

[8] **OGC** (*Open Geospatial Consortium*) (www.opengeospatial.org) je medzinárodné neziskové priemyselné združenie komerčných spoločností, vládnych inštitúcií a univerzít, ktoré vyvíja štandardy a špecifikácie pre otvorené používateľské rozhranie, ktoré zefektívni prácu s priestorovými dátami najmä v prostredí webu, mobilných technológií, lokalizačných služieb a štandardných aplikácií GIS. Štandardy prijaté konzorciom OGC majú v produktoch a aplikáciách GIS široké zastúpenie a zároveň sa stávajú základom niektorých noriem série ISO 19 100 Geografické informácie.

[9] OPEN GEOSPATIAL CONSORTIUM Inc.: OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture, (OGC 06-103r4, 2011);

vektorové aj rastrové dáta, formáty WKT *(Well-Known Text)* a WKB *(Well-Known Binary)*[10], priestorové indexy na zrýchlenie dopytovania sa na priestorové objekty a obsahuje aj približne 900 funkcií na analýzu a spracovanie priestorových dát. Podľa geometrického modelu OGC (modelu geometrie OpenGIS) podporuje dátové typy ako napr. Point (bod), Line (línia) a Polygon (polygón). Medzi funkciami na spracovanie a analýzu priestorových dát nechýbajú funkcie typu ST_Area, ST_AsBinary, ST_AsGML, ST_AsGeoJSON, ST_AsKML, ST_AsText, ST_GeogFromText, ST_Buffer, ST_Intersects, ST_Touches, ST_Union, ST_Centroid, ST_Contains, ST_ConvexHull, ST_Overlaps, ST_Length, ST_Disjoint, ST_Distance a mnoho ďalších (kapitola **3.2**). PostGIS podporuje okrem iného aj transformáciu priestorových dát medzi rôznymi súradnicovými systémami, medzi ktorými môže byť nakonfigurovaný aj na Slovensku používaný súradnicový systém S-JTSK (Systém jednotnej trigonometrickej siete katastrálnej).

Prístup k priestorovej databáze v PostgreSQL je prostredníctvom modulu PostGIS podporovaný v mnohých softvéroch GIS, napr. aj v open source softvéroch GRASS a QGIS[11] (kapitola **4.2**) alebo v proprietárnom softvérovom prostredí ArcGIS. Priestorové dáta z rozšírenia PostGIS môžu byť priamo načítané nielen do desktopových GIS (napr. QGIS), ale môžu byť prepojené aj priamo s mapovými servermi, ako sú napr. GeoServer (www.geoserver.org) alebo MapServer (www.mapserver.org).

1.2.2 MySQL

MySQL (www.mysql.com[12]) je tiež veľmi populárny open source databázový systém, ktorý je v súčasnosti dostupný v rôznych edíciách, ako napr. MySQL Enterprise Edition, MySQL Standard Edition alebo MySQL Classic Edition. MySQL Community Edition je voľne dostupná verzia, ktorá je sprístupnená pod licenciou GNU GPL[13] *(GNU's Not Unix*

OPEN GEOSPATIAL CONSORTIUM Inc.: OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 2: SQL option, (OGC06-104r4, 2010).

[10] Formáty **WKT** a **WKB** poskytujú možnosť reprezentovať objekty jednotným spôsobom v textovom (WKT) alebo binárnom tvare (WKB) a tým ich napr. aj prenášať do iných databázových systémov. Uvedené formáty sú opísané v štandarde (OGC 06-103r4, 2011) alebo v (Koreň, 2009).

[11] pôvodne QuantumGIS

[12] citované 2. 10. 2013

[13] **Licencia GNU GPL** (alebo len GPL) (http://www.gnu.org/copyleft/gpl.html) je najrozšírenejšia licencia pre slobodný softvér, ktorá používateľom garantuje slobodu používania, študovania, zdieľania a modifikácie

General Public License). Systém bol podporovaný aktívnou komunitou open source vývojárov a bol pôvodne realizovaný na platforme Linux ako serverový databázový systém spoločnosťou MYSQL AB zakúpenou v roku 2008 spoločnosťou Sun Microsystems, ktorá od roku 2009 tvorí súčasť spoločnosti Oracle Corporation.

MySQL je napísaný v jazyku C a C++ a podporuje operačné systémy ako Linux, Windows, Mac OS X, Solaris a iné. Používa veľmi rýchle indexy (B-tree diskové tabuľky) s možnosťou kompresie indexov. Databáza môže obsahovať až približne 200 000 tabuliek, ktoré môžu zahŕňať až 5 miliárd riadkov. Podporuje 64 indexov na tabuľku, kde každý index môže pozostávať z 1 až 16 stĺpcov. Dostupné sú aj aplikačné programové rozhrania API *(Application programming interface)* pre jazyky C, C++, Eiffel, Java, Perl, PHP, Python, Ruby a Tcl alebo konektory pre databázové rozhrania ODBC *(Open Database Connectivity)*, JDBC (*Java Database Connectivity*) alebo aj .NET. Databázový systém MySQL má plnú podporu pre rôzne znakové sady, v ktorých sú údaje ukladané, a je aj ACID kompatibilný.

MySQL tiež podporuje priestorové rozšírenia, ktoré umožňujú vytváranie, uchovávanie a analýzu priestorových (geografických) dát. Pre priestorové atribúty (stĺpce) typu MyISAM *(Indexed Sequential Access Method)* podporuje priestorové indexy. MySQL podporuje aj dátové typy podľa geometrického modelu OGC a to konkrétne Point (bod), LineString (líniový reťazec), Polygon (polygón), GeometryCollection (zložené objekty), MultiPoint (objekt zložený z viacerých bodov), MultiLineString (zložený líniový reťazec, resp. objekt zložený z viacerých líniových reťazcov) a MultiPolygon (zložený polygón). Na reprezentáciu geometrických objektov v dopytoch sa používajú formáty WKT a WKB. Medzi príklady podporovaných funkcií patria napríklad funkcie AsBinary, AsWKB, AsText, AsWKT, GeomFromText, GeomFromWKB, Area, Centroid, GLength, Buffer, ConvexHull, Difference, Intersection, Union, ST_Contains, ST_Within, ST_Crosses, ST_Disjoint, ST_Equals, ST_Intersects, ST_Overlaps alebo ST_Touches (kapitola **3.2**).

1.2.3 Oracle

Oracle (www.oracle.com[14]) predstavuje v súčasnosti špičkové riešenie v oblasti databáz. Ide o proprietárny databázový systém, ktorý je známy svojou výkonnosťou, bezpečnosťou, dostupnosťou či schopnosťou spravovať veľké objemy dát. Je vyvíjaný

softvéru. Na splnenie slobody štúdovania a modifikácie softvéru je nevyhnutné poskytovanie otvoreného zdrojového kódu (*open source*).

[14] citované 3. 10. 2013

spoločnosťou Oracle Corporation a je dostupný v edíciách Standard Edition, Standard Edition One, Enterprise Edition alebo Express Edition, ktorá je voľne dostupná. Najnovšia verzia Oracle Database 12c je navrhovaná špeciálne pre klaudové technológie *(cloud technology)*[15], ktoré sa v súčasnosti často využívajú.

Aj databázový systém Oracle okrem iného podporuje spracovanie priestorových dát. Priestorové rozšírenie (nadstavba) sa nazýva Oracle Spatial and Graph a zahŕňa podporu na spravovanie a analýzu rôznych typov priestorových dát. Podporuje vektorové a rastrové dáta, 3D dátový model a 3D geometriu, ale napr. aj geokódovanie alebo vyhľadávanie trás. Oracle obsahuje súbor priestorových funkcií a operátorov, medzi ktoré patria napr. SDO_AGGR_CENTROID, SDO_AGGR_CONVEXHULL, SDO_AGGR_LRS_CONCAT, SDO_AGGR_MBR, SDO_AGGR_CONCAT_LINES, SDO_WITHIN_DISTANCE, SDO_ANYINTERACT, SDO_CONTAINS, SDO_COVEREDBY, SDO_COVERS, SDO_EQUAL, SDO_INSIDE, SDO_OVERLAPS, SDO_TOUCH, SDO_JOIN, SDO_POINTINPOLYGON, SDO_OVERLAPBDYDISJOINT, SDO_AGGR_SET_UNION, SDO_AGGR_UNION alebo SDO_OVERLAPBDYINTERSECT (podrobná charakteristika uvedených funkcií, ako aj ďalších funkcií podporovaných v ORACLE je uvedená napr. na stránke http://docs.oracle.com/cd/E11882_01/appdev.112/e11830/toc.htm). Oracle navyše podporuje aj OGC špecifikácie Web Map Service (WMS), Web Feature Service (WFS), Catalog Service for Web (CSW) a Location Services (OpenLS).

1.2.4 SQL Server

Databázový systém SQL Server (http://www.microsoft.com[16]) je produkt spoločnosti Microsoft (pôvodne bol vyvíjaný ako spoločný produkt spoločností Microsoft a Sybase). SQL je objektovo relačný (pôvodne relačný) databázový systém (kapitola **2.1**), ktorý je dostupný v edíciách ako Enterprise, Standard, Workgroup, Web, Compact, alebo Express. SQL Server ponúka širokú škálu integrovaných služieb, ktoré umožňujú vykonávať rôzne operácie s dátami, ako napríklad dopyty, vyhľadávanie, synchronizáciu, generovanie zostáv a vytváranie analýz. Ochrana dát je zabezpečená dynamickým šifrovaním v celej databáze,

[15] „***Cloud technology***“ predstavuje technológie poskytovania služieb alebo softvéru uložených na serveroch v prostredí webu, ku ktorým môžu používatelia pristupovať z ľubovoľných miest napr. pomocou webových prehliadačov alebo API konkrétnej aplikácie. Týmto spôsobom vytvorený systém neponúka softvér ako produkt, ale samotné spracovanie dát ako službu v prostredí webu (t. j. používatelia neplatia napr. za softvér, ale za jeho využívanie).

[16] citované 8. 10. 2013

dátových súboroch alebo súboroch protokolu bez nutnosti zmeny aplikácií. Vďaka možnostiam vizualizácie a integrácii so systémom Microsoft Office možno použiť nástroje *business intelligence* (tzv. podnikovej inteligencie), spravovať zostavy a analýzy ľubovoľnej veľkosti a úrovne podrobnosti a uľahčiť prácu používateľom.

SQL Server poskytuje od verzie 2008 aj komplexnú podporu priestorových dát, pričom podľa použitého súradnicového systému rozlišuje geometrické a geografické dátové typy (Lacko, 2011) (kapitola **3.1**). SQL Server umožňuje jednoducho spracovávať, používať a rozširovať priestorové dáta prostredníctvom aplikácií s priestorovými funkciami. Podporuje dátové typy ako napr. Point, LineString, CircularString (kruhový reťazec), CompoundCurve (zložená krivka), Polygon, CurvePolygon, MultiPoint, MultiLineString, MultiPolygon alebo GeometryCollection. Súčasťou je podpora priestorových indexov alebo priestorových funkcií (kapitola **3.2**) ako napr. STArea, STAsText, STBuffer, STContains, STConvexHull, STDifference, STDimension, STDisjoint, STDistance, STEndpoint, STEquals, STGeometryType, STIntersection, STIntersects, STIsClosed, STIsEmpty, STIsValid, STLength, STNumPoints, STOverlaps, STUnion, STWithin.

1.3 Inštalácia SRDB PostgreSQL

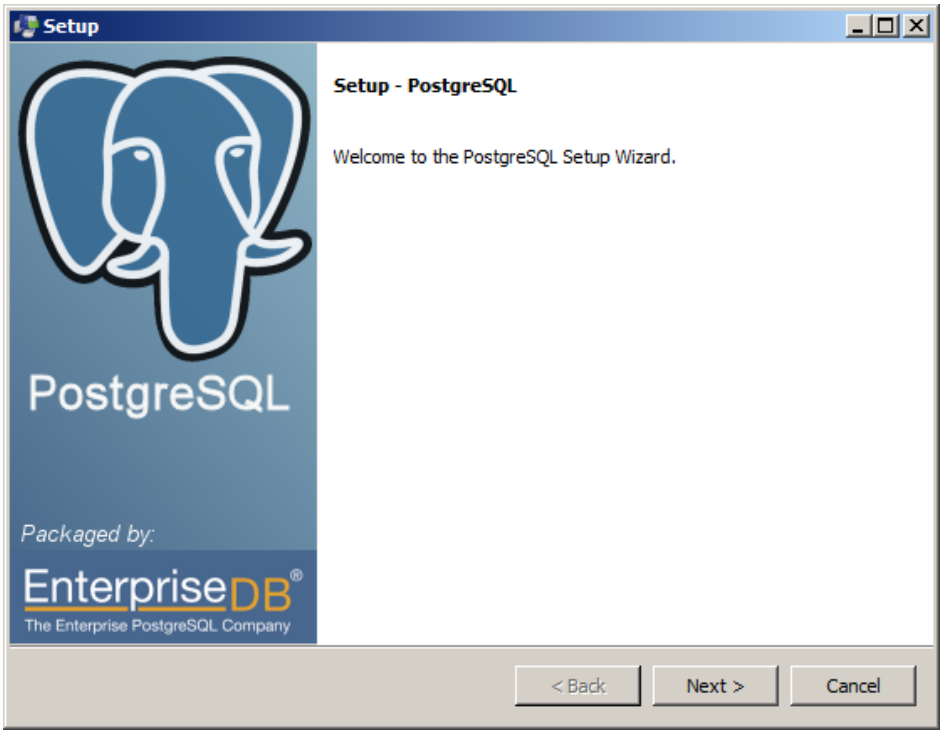
(1. cvičenie)

Zadanie č. 1:

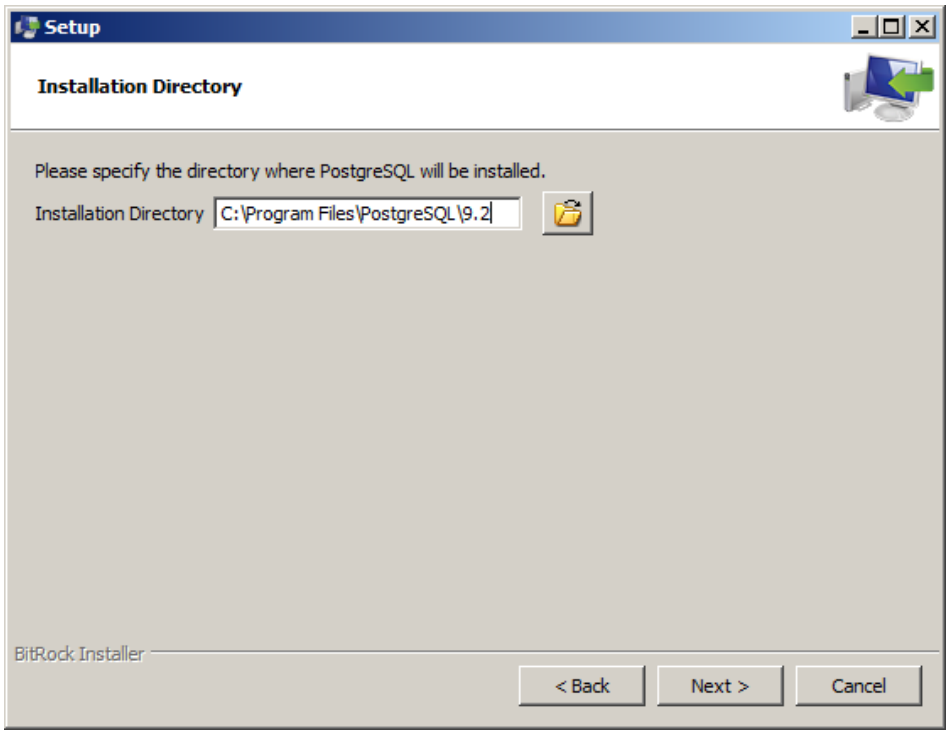
Nainštalujte SRDB PostgreSQL a jeho rozšírenie PostGIS do určeného počítača.

Postup riešenia:

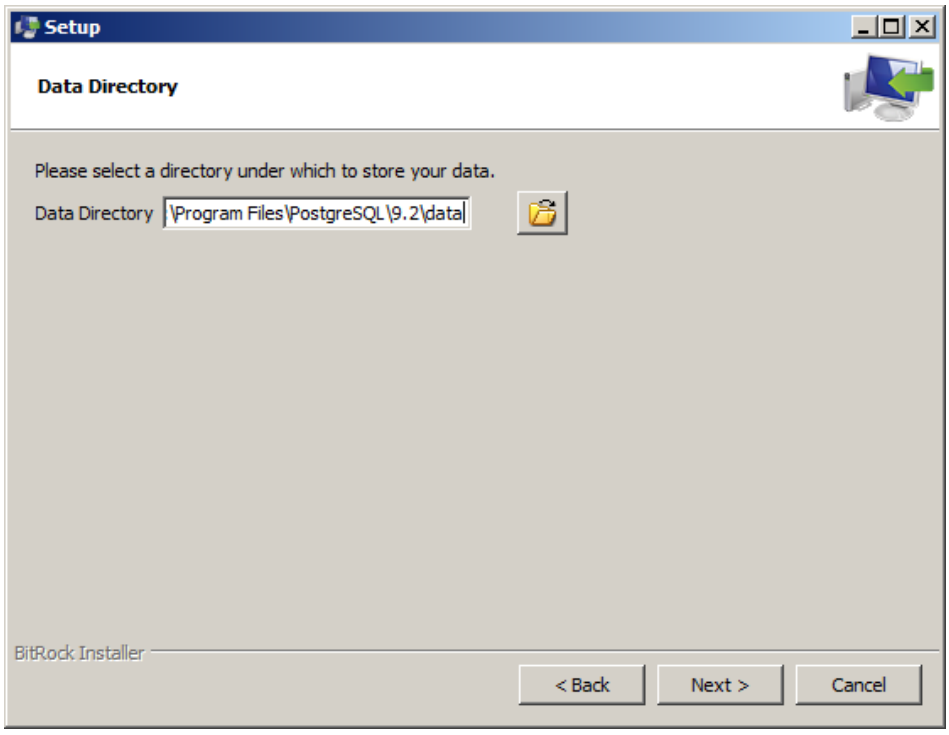
1. Inštalačný súbor môžete stiahnuť z internetu cez www.postgresql.org/download/. Postup inštalácie je opísaný pre operačný systém Windows. Pre 64 bitovú verziu systému Windows je to súbor postgresql-9.2.4-1-windows-x64.exe.
2. Inštaláciu spustíte dvojklikom na stiahnutý súbor. Po spustení sa zobrazí uvítacie okno (Obr. 1.2). V inštalácii sa pokračuje pomocou tlačidla „Next“.
3. V ďalšom kroku špecifikujte miesto pre inštaláciu databázového systému (Obr. 1.3).
4. Špecifikujte adresár do ktorého budú ukladané údaje databázového systému (Obr. 1.4)
5. Definujte heslo pre vytvoreného používateľa „postgres“ (Obr. 1.5).



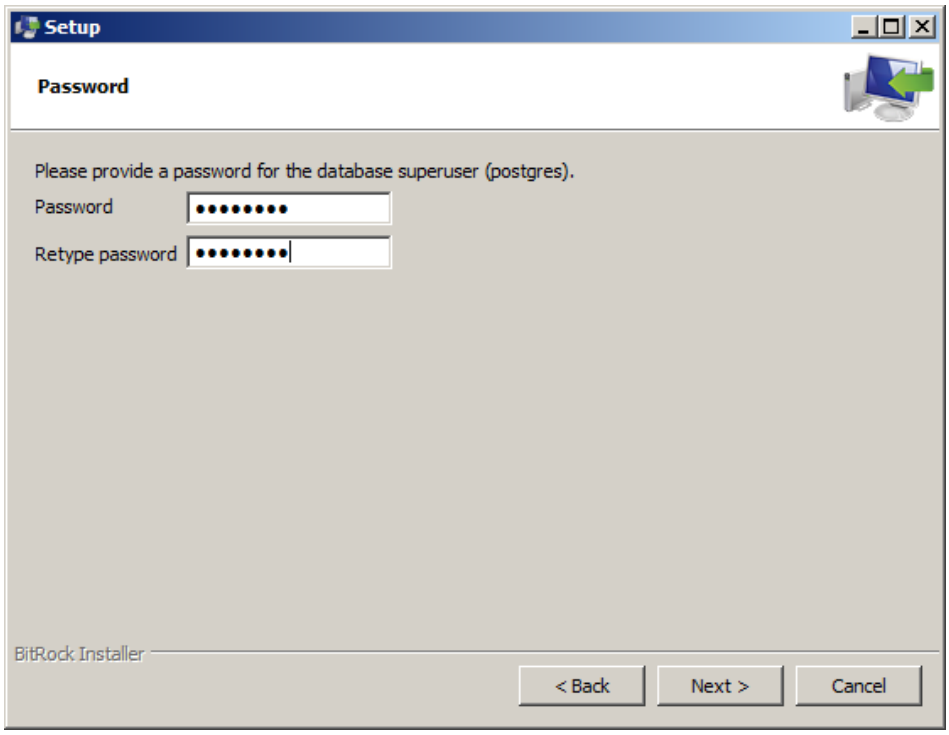
Obr. 1.2. Inštalácia PostgreSQL



Obr. 1.3. Špecifikácia miesta pre inštaláciu PostgreSQL



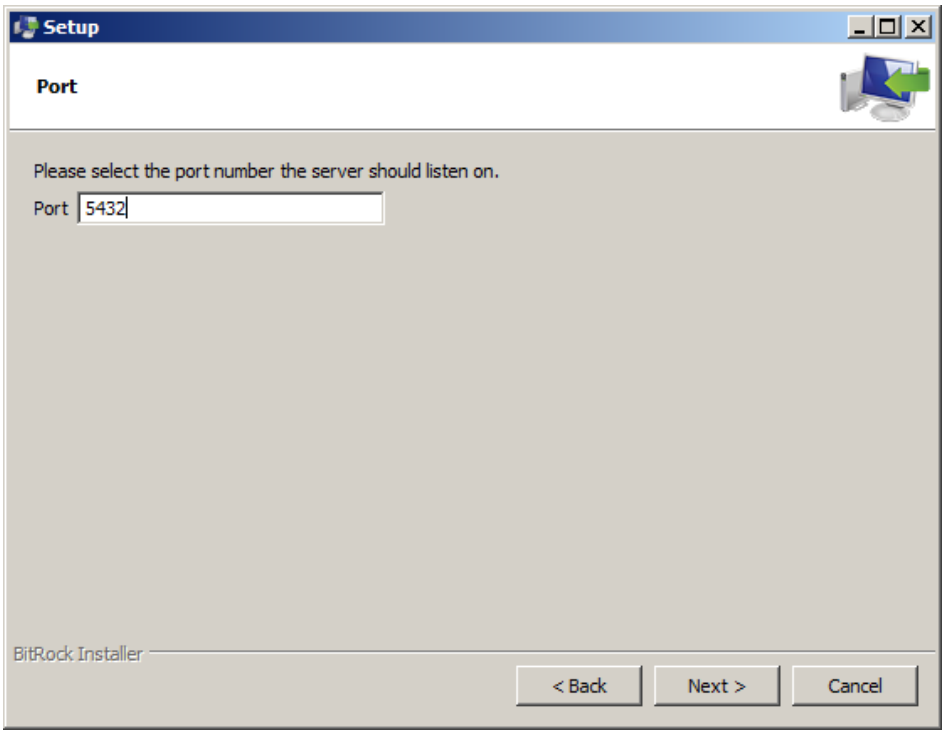
Obr. 1.4. Špecifikácia adresára na ukladanie údajov



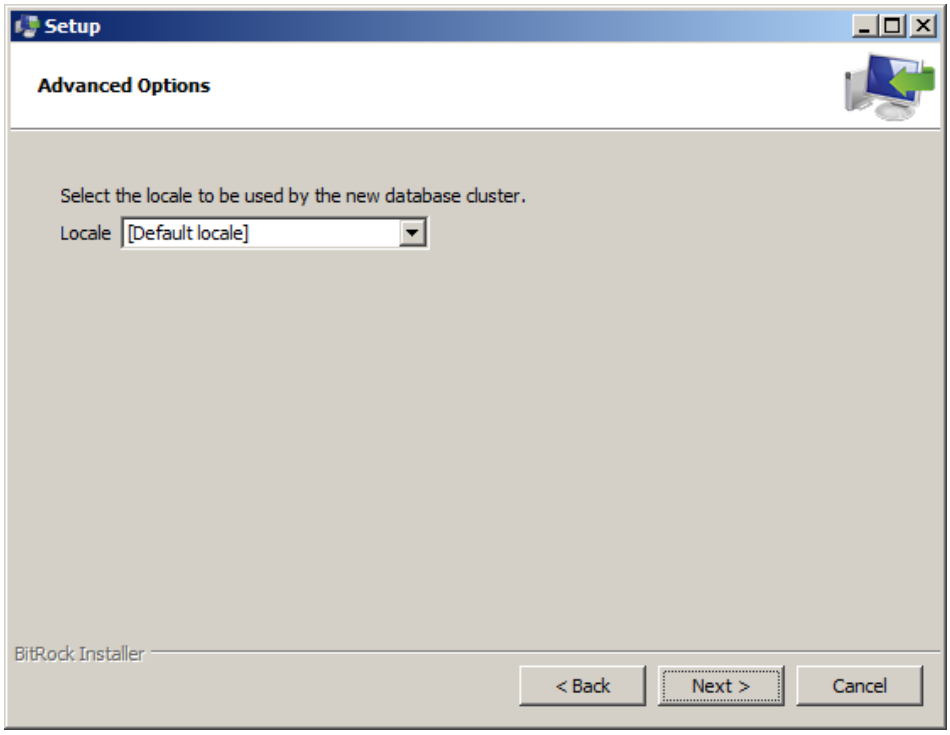
Obr. 1.5. Definovanie hesla pre používateľa „postgres“

6. Vyberte port, na ktorom bude databázový systém dostupný. Štandardne sa pre PostgreSQL používa port „5432“ (Obr. 1.6).
7. Zvoľte miestne nastavenia, ktoré budú použité pre databázový klaster *(cluster)*[17] (Obr. 1.7).
8. Po príprave inštalácie ju spustíte tlačidlom „Next“ (Obr. 1.8).
9. Priebeh inštalácie sa zobrazuje v nasledujúcom okne (Obr. 1.9).

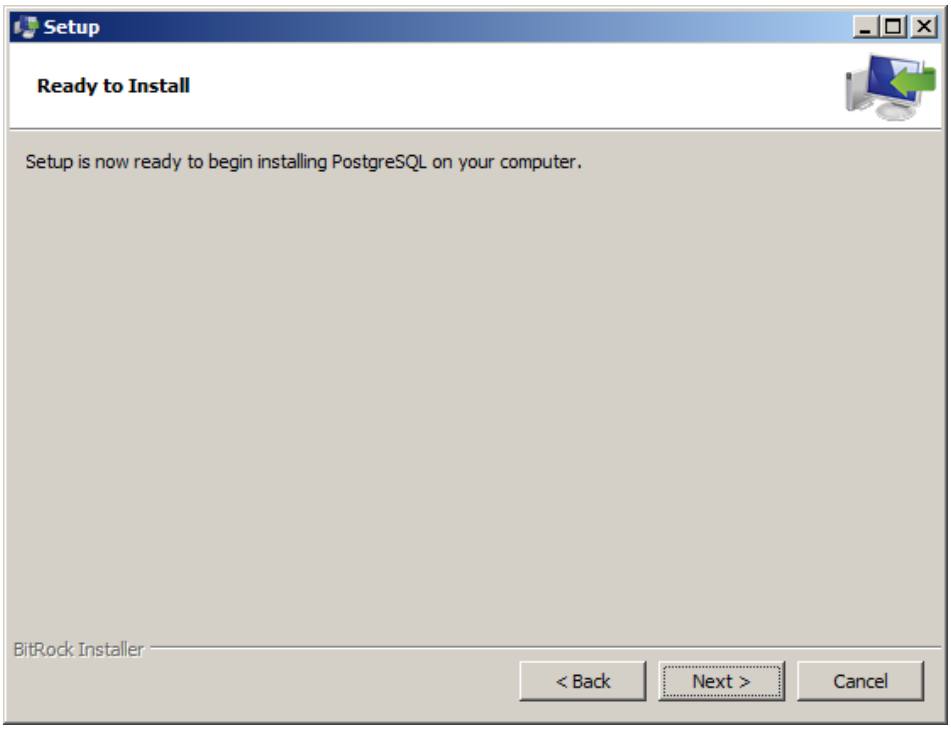
[17] Klaster tvorí súbor počítačov, ktorý môže prevádzkovať jeden virtuálny server. Počítače v klastri sú sieťovo prepojené.



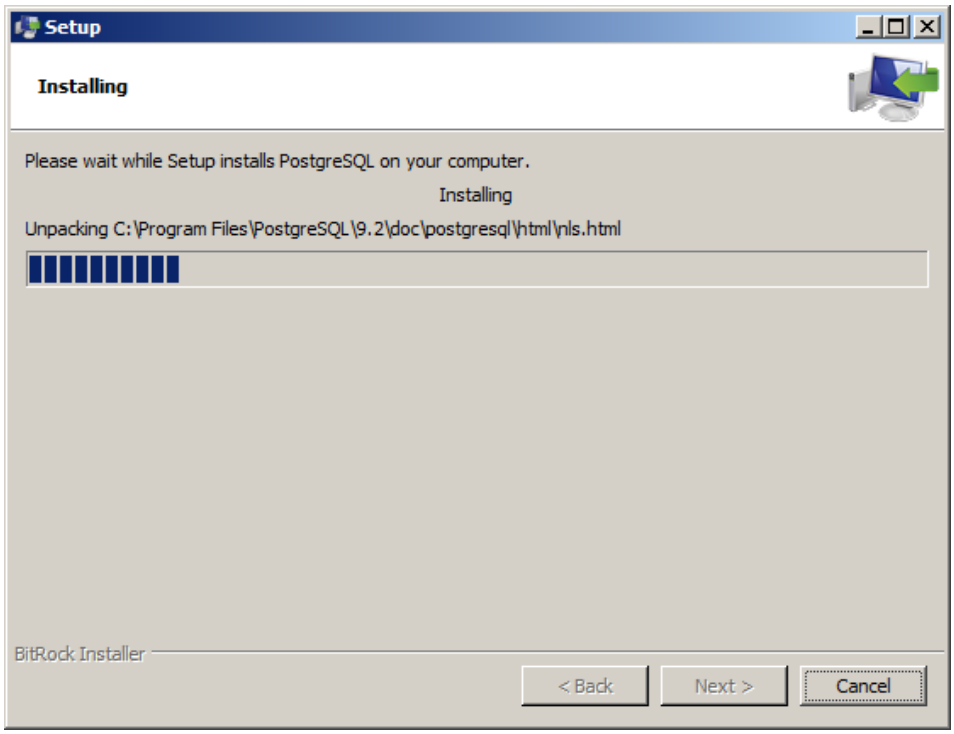
Obr. 1.6. Definovanie portu, na ktorom bude systém dostupný



Obr. 1.7. Miestne nastavenia pre databázový klaster



Obr. 1.8. Spustenie inštalácie

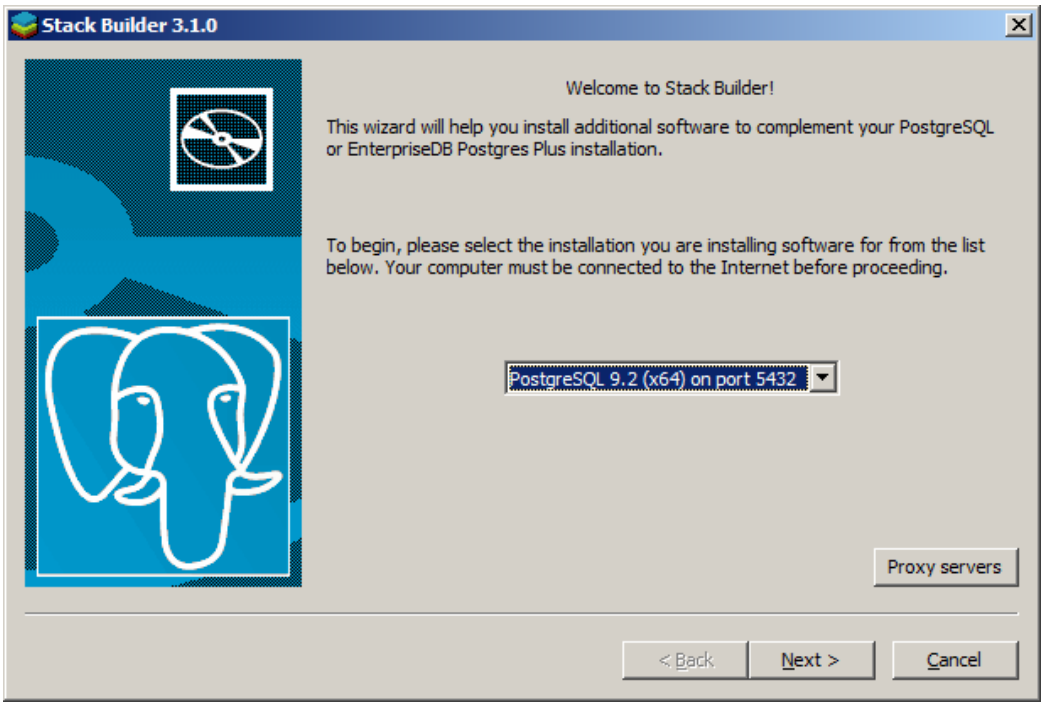


Obr. 1.9. Zobrazenie priebehu inštalácie

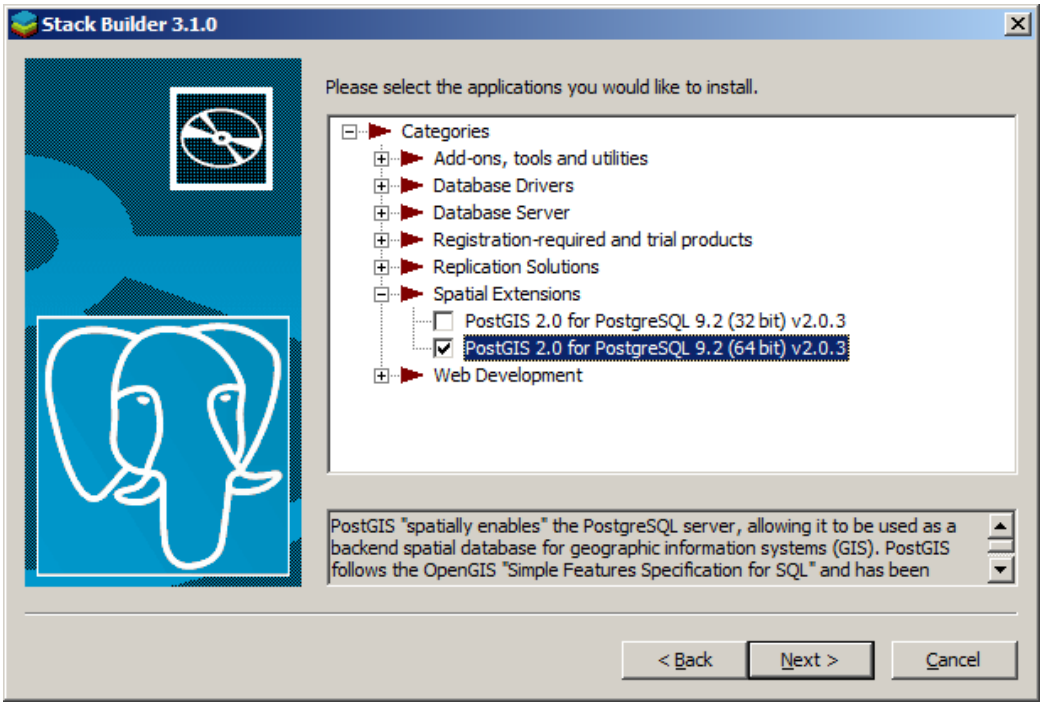
10. Po ukončení inštalácie možno spustiť nástroj „*Stack Builder*“, prostredníctvom ktorého nainštalujete rozšírenie PostGIS (Obr. 1.10).

Obr. 1.10. Spustenie nástroja „*Stack Builder*“

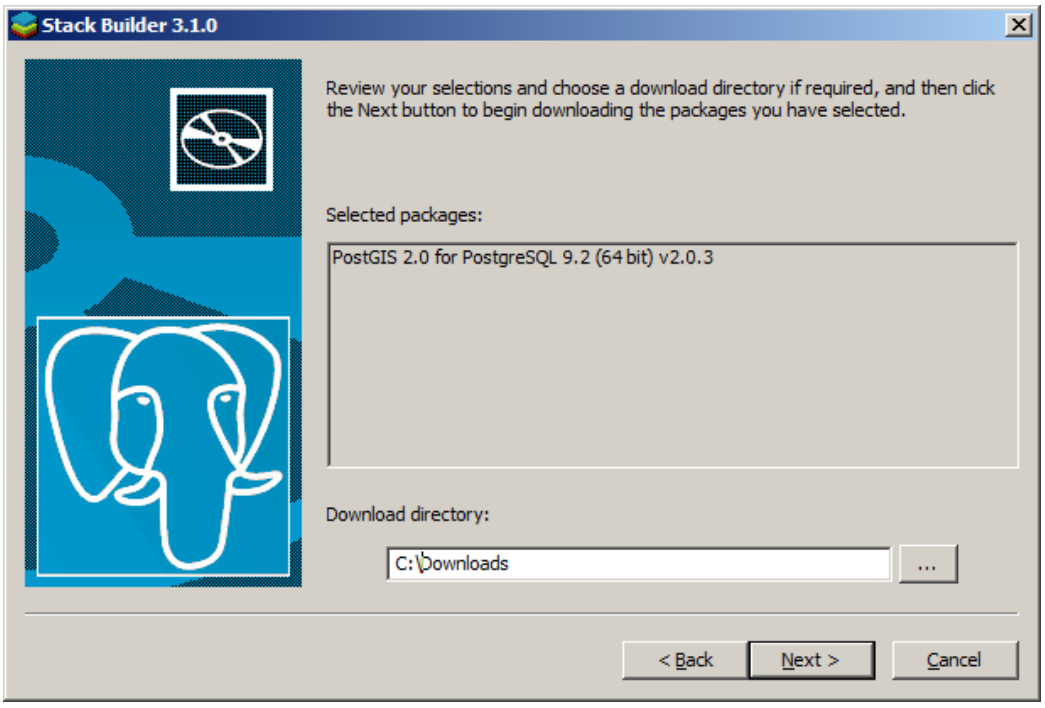
11. Vyberte inštanciu nainštalovaného databázového systému, do ktorého budete inštalovať rozšírenie PostGIS. Počas tohto kroku je potrebné pripojenie do siete internet (Obr. 1.11).
12. V časti „Spatial Extensions“ vyberte PostGIS 2.0 pre 64 bitovú verziu PostgreSQL (Obr. 1.12).
13. Vyberte adresár, do ktorého budú uložené inštalačné súbory (Obr. 1.13).
14. Počkajte, kým sa uložia inštalačné súbory (Obr. 1.14).
15. Po uložení súborov môžete spustiť inštaláciu pomocou tlačidla „Next“ (Obr. 1.15).



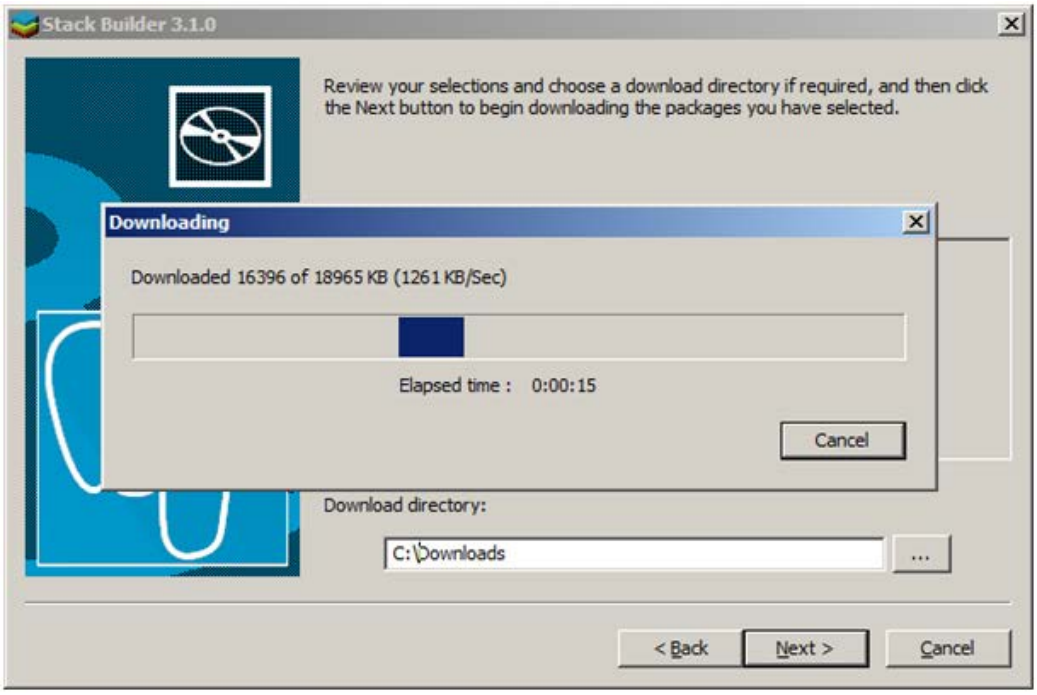
Obr. 1.11. Výber inštancie systému



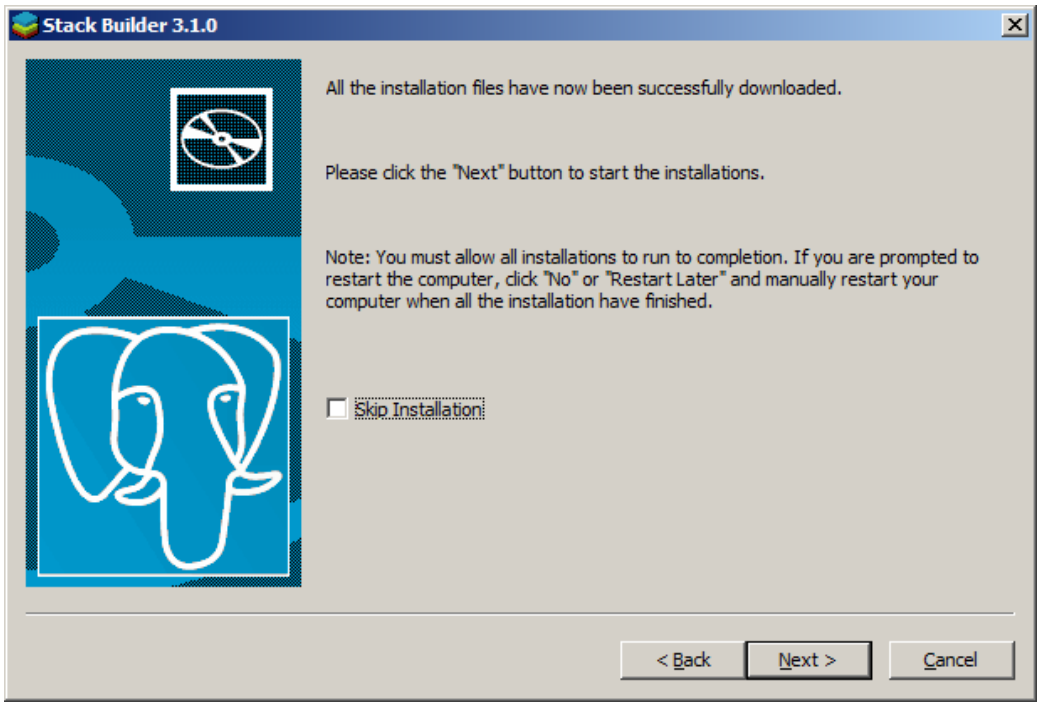
Obr. 1.12. Výber rozšírenia PostGIS



Obr. 1.13. Výber adresára pre inštalačné súbory

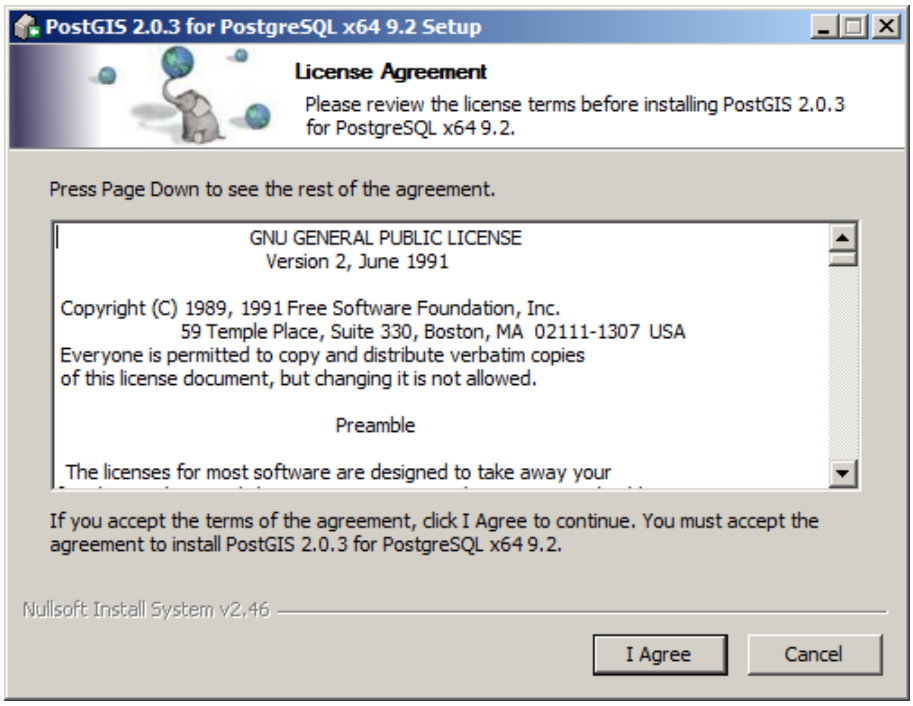


Obr. 1.14. Uloženie inštalačných súborov

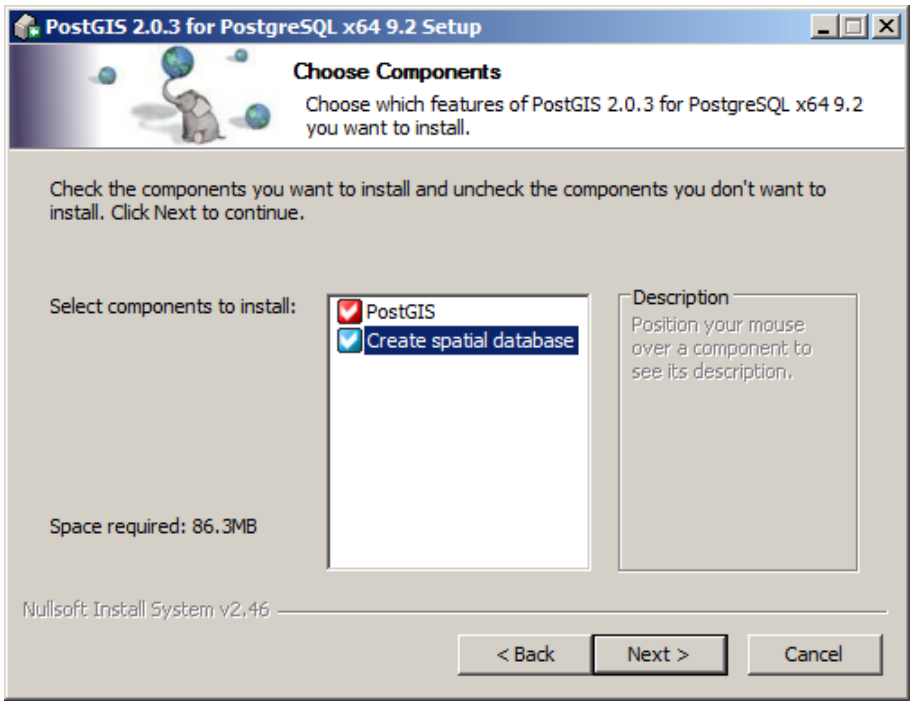


Obr. 1.15. Spustenie inštalácie

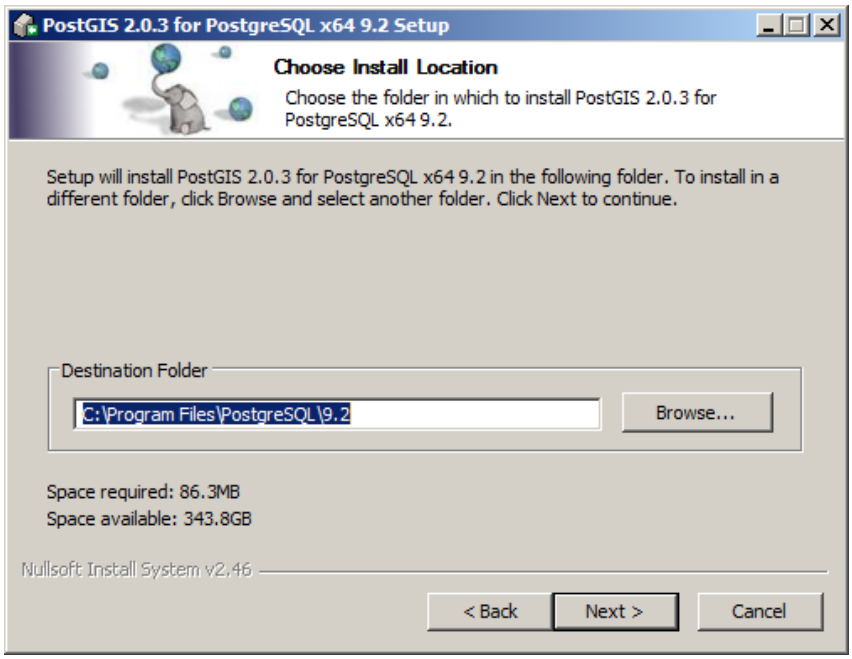
16. Licenčné podmienky potvrďte tlačidlom „I Agree“ (Obr. 1.16).
17. Označte zaškrtávacie políčko *(check box)* „Create spatial database“, čím povolíte vytvorenie vzorovej priestorovej databázy pri inštalácii (Obr. 1.17).
18. Zvoľte adresár pre inštaláciu rozšírenia PostGIS (Obr. 1.18).
19. Definujte parametre pripojenia na inštalovaný databázový systém (Obr. 1.19).
20. Zvoľte názov priestorovej databázy (napr. postgis20) a môžete spustiť inštaláciu pomocou tlačidla „Install“ (Obr. 1.20).
21. Počkajte, pokiaľ sa rozšírenie nainštaluje (Obr. 1.21).



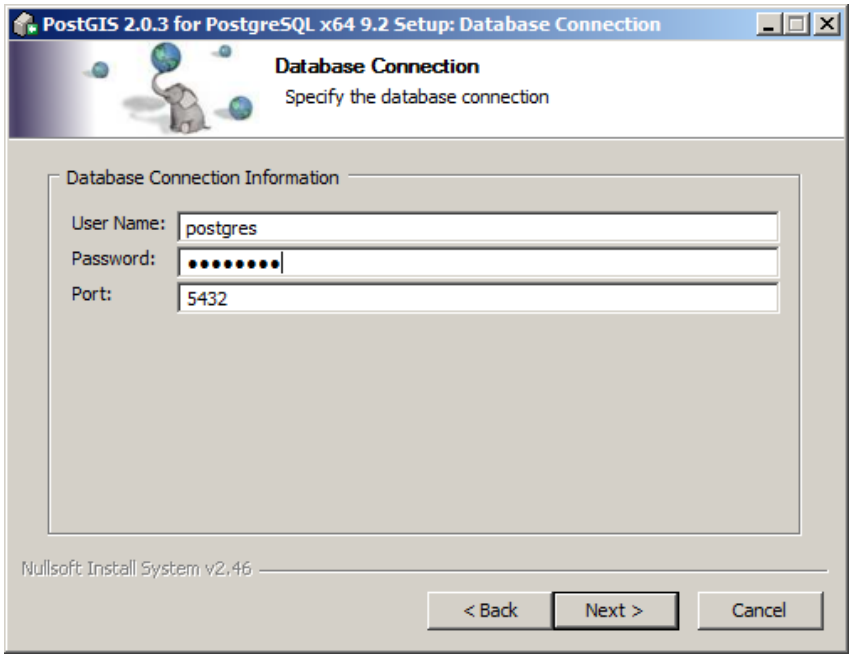
Obr. 1.16. Potvrdenie licenčných podmienok



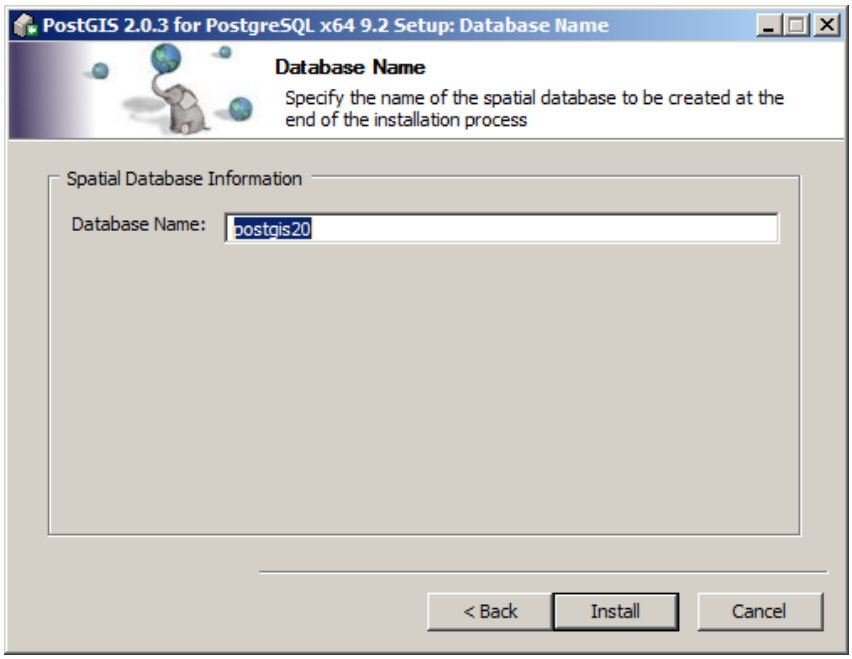
Obr. 1.17. Povolenie vytvorenia vzorovej priestorovej databázy



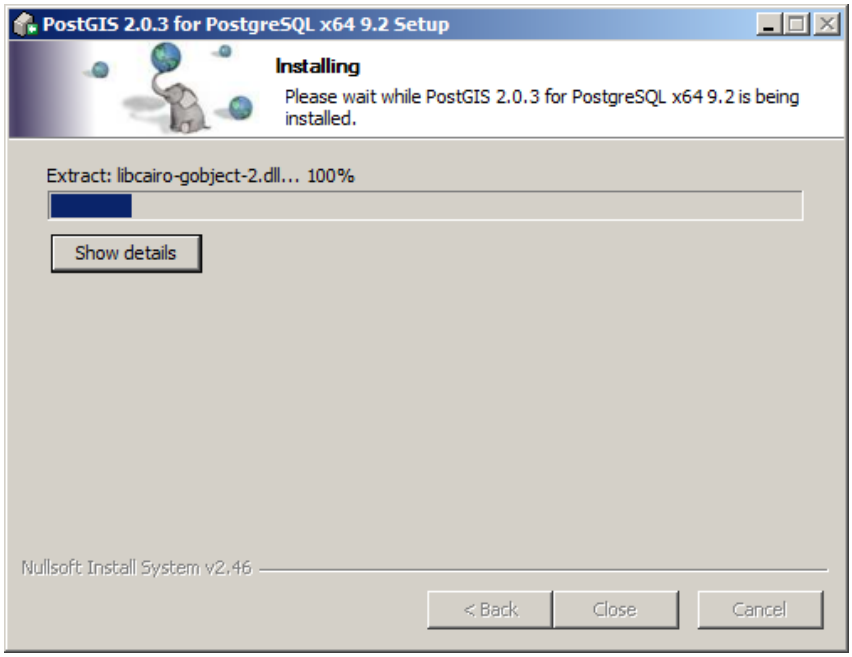
Obr. 1.18. Nastavenie adresára pre inštaláciu rozšírenia PostGIS



Obr. 1.19. Definovanie parametrov pripojenia

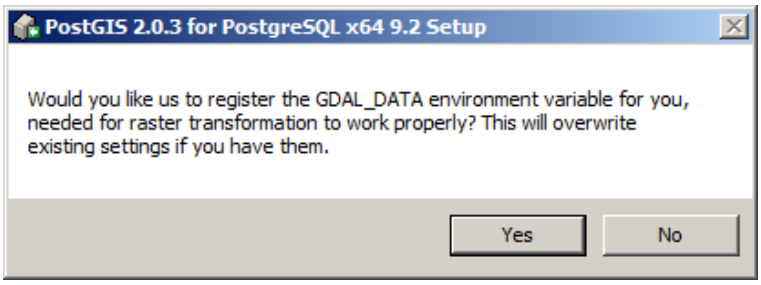


Obr. 1.20. Nastavenie názvu priestorovej databázy



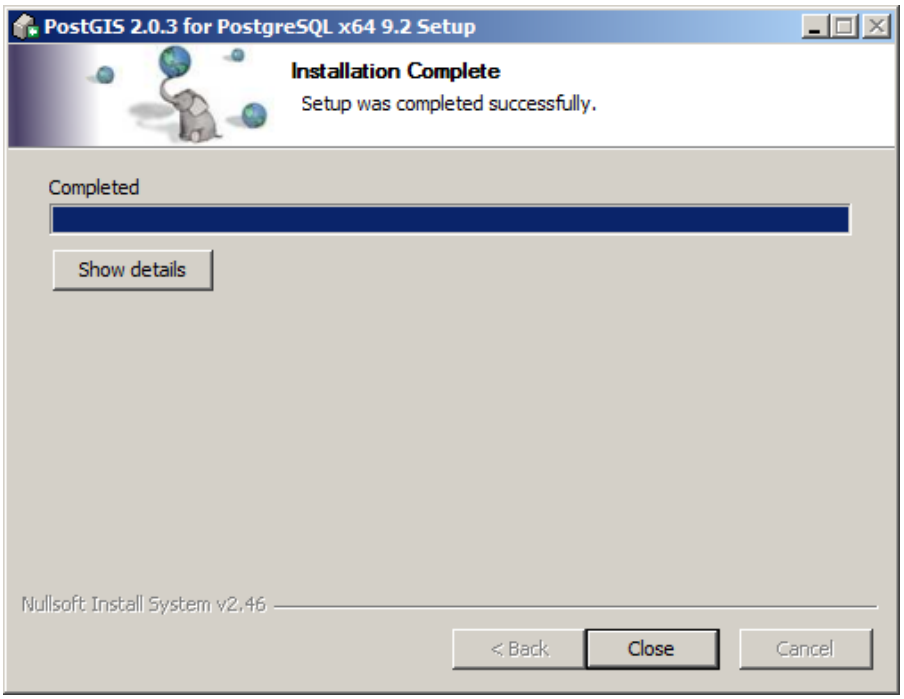
Obr. 1.21. Inštalácia rozšírenia PostGIS

22. Potvrďte prepísanie premennej prostredia „GDAL_DATA“ (v prípade, že vyžadujete aj transformácie rastrových dát), (Obr. 1.22).



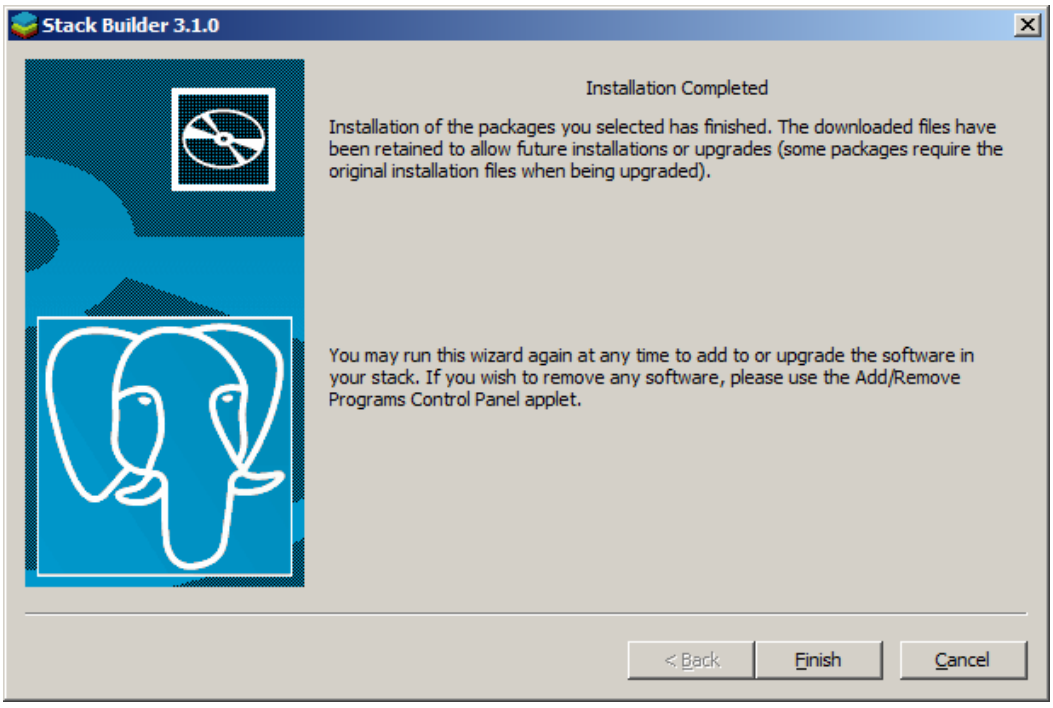
Obr. 1.22. Potvrdenie registrácie prostredia „GDAL_DATA“

23. Zavrite inštalačné okno rozšírenia PostGIS pomocou tlačidla „Close“ (Obr. 1.23).



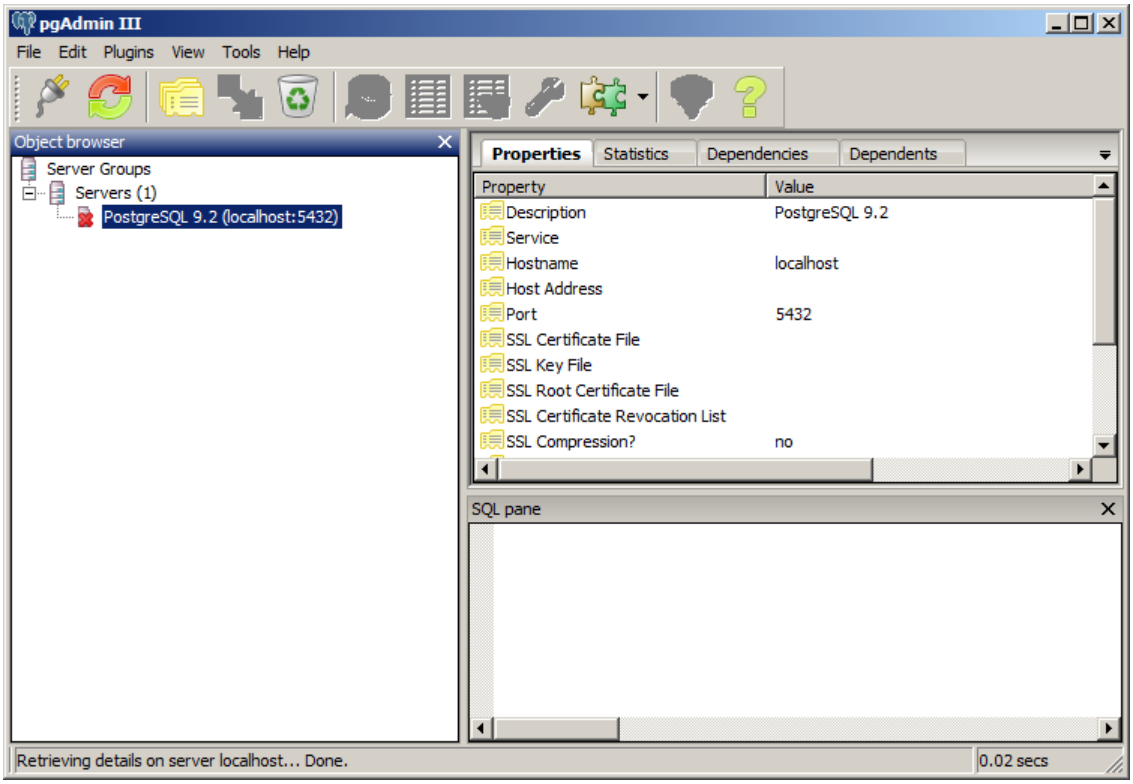
Obr. 1.23. Ukončenie inštalácie rozšírenia PostGIS

24. Zavrite okno „Stack Builder“ pomocou tlačidla „Finish“ (Obr. 1.24).



Obr. 1.24. Zatvorenie okna „*Stack Builder*“

25. Po spustení nástroja pgAdmin III sa zobrazí nasledujúce okno (Obr. 1.25).

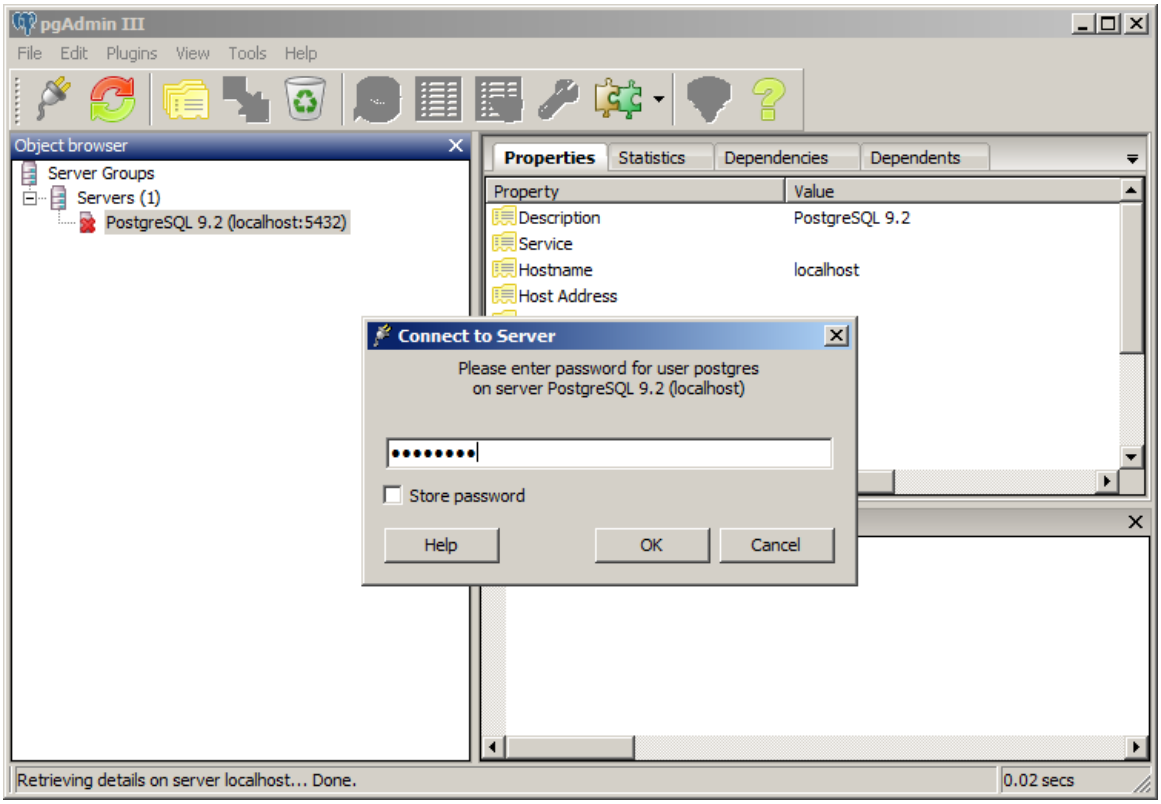


Obr. 1.25. Používateľské rozhranie pgAdmin III

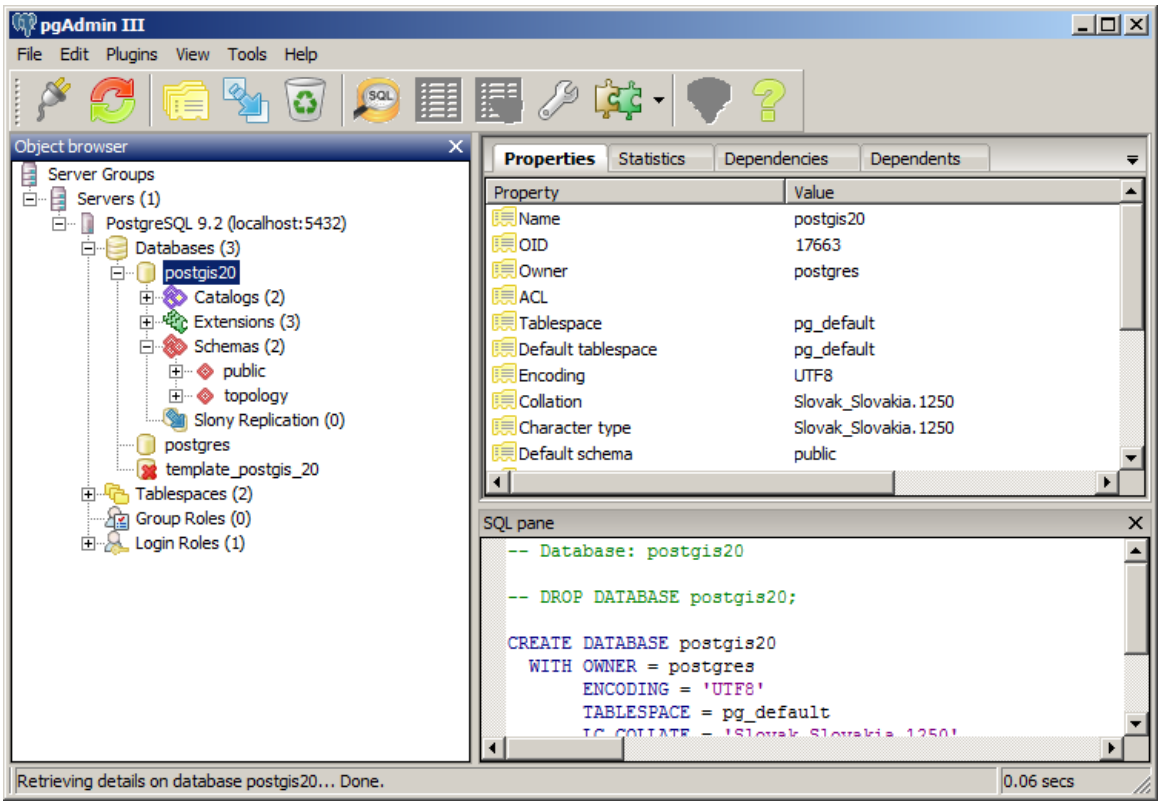
26. Na nainštalovanú inštanciu sa pripojíte pomocou používateľa „postgres“ a hesla zadaného počas inštalácie (Obr. 1.26).

27. Po prihlásení možno pracovať v databázovom systéme a vo vytvorenej databáze postgis20 (Obr. 1.27).

28. V prípade potreby ďalších nastavení môžete vykonať rozšírenú konfiguráciu systému v súboroch „C:\Program Files\PostgreSQL\9.2\data\pg_hba.conf“ a „C:\Program Files\PostgreSQL\9.2\data\postgresql.conf“.



Obr. 1.26. Pripojenie sa na nainštalovanú inštanciu



Obr. 1.27. Práca v databázovom systéme s vytvorenou databázou „postgis20“

2 Relačné databázy a základy jazyka SQL

(2. – 6. cvičenie)

Teoretické minimum:

2.1 Relačné a objektovo relačné databázy, relačná algebra

Relačné databázy majú názov odvodený od matematického pojmu relácia a ich matematický základ tvorí relačná algebra. Definícia pojmu relácia vychádza z definície karteziánskeho súčinu množín.

Karteziánsky súčin *n* množín A_1, A_2,... A_n je množina $A_1 \times A_2 \times ... \times A_n$, prvkami ktorej sú usporiadané *n*-tice $[x_1, x_2, ..., x_n]$, kde $x_i \in A_i$, $1 \leq i \leq n$. Napríklad karteziánsky súčin dvoch množín A_1, A_2 je množina $A_1 \times A_2$ obsahujúca všetky usporiadané dvojice, ktorých prvý prvok patrí do množiny A_1 a druhý prvok patrí do množiny A_2.

Podmnožina karteziánskeho súčinu $A_1 \times A_2 \times ... \times A_n$ sa nazýva **relácia**. Relácia v databázach je tá podmnožina karteziánskeho súčinu, ktorá zodpovedá vzťahom v realite (resp. obsahuje dáta, ktoré vzťahy v realite reprezentujú). Napr. ak množina

$A_1 = \{11, 22, 33\}$

je množina parciel a množina

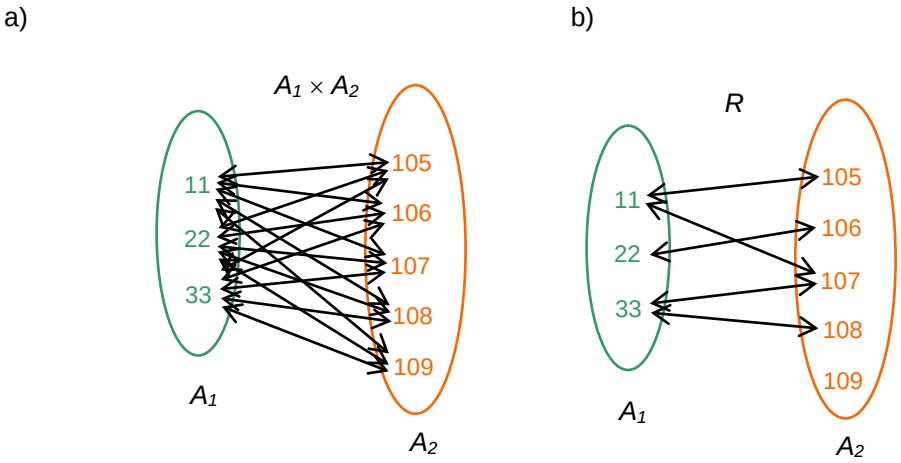
$A_2 = \{105, 106, 107, 108\}$

je množina osôb, relácia môže byť tá podmnožina karteziánskeho súčinu $A \times B$ (Obr. 2.1a):

$A_1 \times A_2 = \{[11,105], [11,106], [11,107], [11,108],$
$[22,105], [22,106], [22,107], [22,108],$
$[33,105], [33,106], [33,107], [33,108]\},$

ktorá zodpovedá skutočným vlastníckym vzťahom osôb a parciel, t. j. napr. množina R (Obr. 2.1b):

R = {[11,105], [11,107], [22,106], [22,107], [22,108], [33,107], [33,108]}.



Obr. 2.1. a) Karteziánsky súčin A1 × A2, b) relácia R

V relačnom databázovom modeli sú relácie vyjadrené dvojrozmernými tabuľkami, ktoré môžu byť vzájomne prepojené. **Tabuľka** je štruktúra, ktorá reprezentuje reláciu napr. na monitore počítača alebo v analógovej forme na papieri. Na Obr. 2.2 je uvedená tabuľka, ktorá reprezentuje reláciu *R* z Obr. 2.1b.

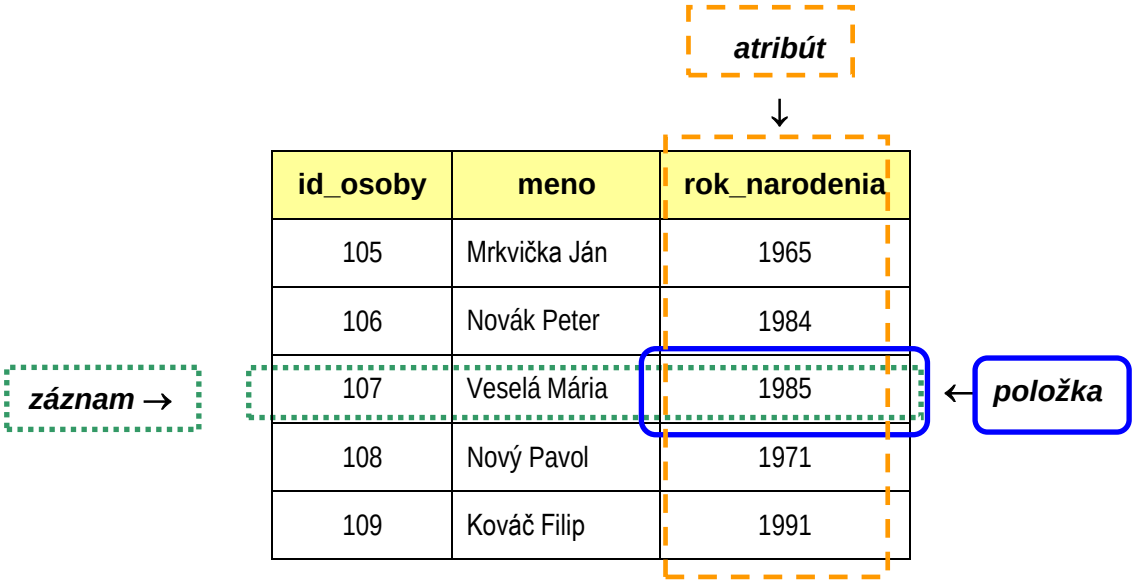
id_parcely	id_osoby
11	105
11	107
22	106
33	107
33	108

Obr. 2.2. Relácia R reprezentovaná tabuľkou

Atribút relácie je pomenovaná množina použitá v karteziánskom súčine. V tabuľke ho reprezentuje jeden stĺpec. Množina hodnôt, ktoré môže atribút nadobúdať, sa nazýva **doména**[18] **atribútu**.

[18] **Doména hodnôt** je podľa STN 73 0401-3 množina akceptovaných (prípustných) hodnôt (pre jeden alebo viac atribútov). Doména môže byť definovaná ako interval alebo ako ľubovoľná iná množina hodnôt.

Prvky relácií sa nazývajú ***n*-tice** *(tuple)*. V databáze ich reprezentuje jeden **záznam** (riadok tabuľky). Záznam (dátová *n*-tica) (Obr. 2.3) je tvorený jednotlivými **položkami** (hodnotami atribútov). Atribút, záznam a položka sú znázornené na Obr. 2.3.



Obr. 2.3. Atribút, záznam a položka

Na identifikáciu záznamov v databáze a tiež na vzájomné prepojenie tabuliek slúžia tzv. **kľúče relácií**. **Primárny kľúč** (*Primary key*) je minimálna množina atribútov, ktorá jednoznačne identifikuje prvok relácie (t. j. identifikuje záznam v tabuľke). Hodnota (alebo v prípade zloženého primárneho kľúča kombinácia hodnôt atribútov) primárneho kľúča musí byť v rámci databázy jedinečná a nesmie obsahovať prázdne hodnoty (obmedzenie NOT NULL (kapitola **2.2.2**)). **Cudzí kľúč** (*Foreign key*) slúži na vzájomné prepojenie relácií (tabuliek). Je to väčšinou primárny kľúč prepojenej relácie (tabuľky).

Relačná algebra je množina operácií na relačnom modeli. Podľa počtu relácií, ktoré vstupujú do operácie, rozlišujeme operácie **unárne** (vstupuje do nich len jedna relácia) a ***n*-árne** (vstupuje do nich *n* relácií). Unárne operácie sú **selekcia** (výber niektorých záznamov (riadkov)) *(selection)* a **projekcia** (výber niektorých atribútov (stĺpcov)) *(projection)*. Medzi n-árne (najčastejšie binárne) **operácie relačnej algebry** patria **zjednotenie** *(union)*, **prienik** *(intersection)*, **rozdiel** *(difference)*, **symetrický rozdiel** *(symmetric difference)*, **spojenie** *(join)* a **karteziánsky súčin** (spojenie systémom každý s každým) *(cross join)*.

V relačných databázach sa využívajú aj ďalšie operácie, funkcie a charakteristické vlastnosti, ktoré už síce nepatria do relačnej algebry, ale významne rozširujú možnosti výsledného jazyka (zvyšujú jeho silu) (Pokorný, 2000). Patrí medzi ne napr. povolenie prázdnych hodnôt v databáze (hodnota **NULL**[19]), **aritmetické operácie** (sčítanie, odčítanie, násobenie, delenie) a **agregačné funkcie** (**COUNT** (počet prvkov množiny (relácie)), $\mathbf{SUM}_A$ (súčet hodnôt atribútu A), $\mathbf{MIN}_A$ (minimum), $\mathbf{MAX}_A$ (maximum) a $\mathbf{AVG}_A$ (priemer), (kapitola **2.2.1**).

Relačné databázy, ktoré sú rozšírené o niektoré prvky objektovo orientovaného modelovania (objektovo orientovaných databáz), sa nazývajú **objektovo relačné databázy**. Objektovo relačné databázové systémy vznikli doplnením princípov objektovo orientovaných databáz, ktoré sú vhodnejšie na správu a ukladanie priestorových dát, do relačných databáz, ktoré sú v súčasnosti podstatne viac vyvinuté a hlavne rozšírenejšie v existujúcich informačných systémoch. V súčasnosti sú už aj niektoré objektovo rozšírené pôvodne relačné databázové systémy označované ako objektovo relačné. Zahŕňajú výhody relačných databáz, ale aj možnosť práce s objektmi. Objektový prístup umožňuje efektívnejšie pracovať napr. aj s priestorovými dátami (kapitola **3.1**), ktoré dôležité práve pre GIS.

Medzi výhody relačných a objektovo relačných databáz patrí jednoduchý matematický aparát, ale aj možnosť využitia viacerých spôsobov na prístup k požadovaným dátam prostredníctvom štandardného dopytovacieho jazyka (kapitola **2.2**).

2.2 Dopytovací jazyk SQL

Štruktúrovaný dopytovací jazyk SQL *(Structured Query Language)* je základný a zároveň štandardizovaný počítačový jazyk na manipuláciu s dátami (výber, vkladanie, úpravu, mazanie) a definíciu dát v relačných databázach. V súčasnosti je SQL jednoznačne najpoužívanejší dopytovací jazyk tohto druhu v relačných a objektovo relačných SRDB.

Príkazy jazyka SQL môžeme rozdeliť do štyroch skupín (jazykov):

[19] Hodnota **NULL** reprezentuje v relačných databázach prázdnu hodnotu, t. j. hodnotu neznámu, nezadanú alebo neplatnú pre daný záznam. Hodnoty NULL umožňujú prácu s neúplnými dátami alebo s dátami, ktoré obsahujú výnimky (v porovnaní s relačnou algebrou). Hodnota NULL neznamená to isté ako číselná hodnota nula alebo reťazec medzier, pretože nula a medzera sú hodnoty, ale NULL predstavuje práve absenciu akejkoľvek (inej) hodnoty.

- **jazyk na manipuláciu s dátami – DML** *(Data Manipulation Language)*, ktorý obsahuje príkazy na výber a aktualizáciu dát, napr. SELECT[20], INSERT, UPDATE, DELETE (kapitola **2.2.1**),
- **jazyk na definíciu dát – DDL** *(Data Definition Language)*, ktorý obsahuje príkazy na definíciu dát, napr. CREATE, ALTER, DROP (kapitola **2.2.2**),
- **jazyk na riadenie prístupu k dátam – DCL** *(Data Control Language)*, ktorý zahŕňa príkazy na riadenie prístupu k dátam – GRANT, REVOKE,
- **jazyk na riadenie transakcií – TCL** *(Transaction Control Language)*, ktorý zahŕňa príkazy na riadenie transakcií, ako napr. START TRANSACTION (alebo BEGIN TRANSACTION), COMMIT a ROLLBACK.

2.2.1 Jazyk DML – výber, vkladanie, modifikácia a vymazanie záznamov

Výber dát z databázy

Príkaz **SELECT** je samostatným príkazom jazyka DML, ktorý slúži na **výber (selekciu) dát** z databázy podľa stanovenej podmienky. Ide o silný nástroj na výber, usporiadanie, ale aj spájanie dát (SELECT JOIN). Základná syntax príkazu SELECT je:

```
SELECT [*][21] [zoznam_stĺpcov_výstupnej_zostavy]
FROM Meno_tabuľky
WHERE podmienka_výberu
GROUP BY položky
HAVING podmienka_agregácie
ORDER BY zoznam_stĺpcov [ASC] [DESC];
```

Príkaz na zobrazenie vybratej tabuľky (tabuliek) alebo jej časti sa vždy začína príkazom **SELECT**, za ktorým nasleduje príkaz:

[20] Príkazy v jazyku SQL sa podľa konvencie obvykle píšu veľkými písmenami, aj keď niektoré databázové systémy veľkosť písmen v príkazoch a názvoch databázových objektov nerozlišujú, prípadne umožňujú nastavenie *Case-sensitive* (rozlišovať veľkosť písmen) alebo *Not Case-sensitive* (nerozlišovať veľkosť písmen). Pri nastavení *Not Case-sensitive* možno písať príkazy a názvy objektov (napr. tabuliek alebo triggrov) veľkými písmenami, malými písmenami alebo aj kombinovane, avšak v hodnotách dát musí byť správna veľkosť písma vždy dodržaná.

[21] Symbol * znamená, že vo výsledku budú zahrnuté všetky stĺpce (atribúty).

- **FROM**, v rámci ktorého je uvedená špecifikácia, kde sa požadované informácie nachádzajú (tabuľka, viac tabuliek alebo pohľad),

Za ním sú väčšinou umiestnené aj ďalšie klauzuly jazyka SQL, medzi ktoré patria (v uvedenom poradí):

- **WHERE**, za ktorou sú uvedené podmienky, presne špecifikujúce podmnožinu dát, ktorú je potrebné vybrať,
- **GROUP BY**, ktorá určuje zoskupovanie dát vo výstupe podľa určených pravidiel,
- **HAVING**, ktorá stanovuje obmedzenie výberu jednotlivých skupín vytvorených pomocou klauzuly GROUP BY,
- **ORDER BY**, ktorá určuje spôsob zotriedenia záznamov podľa zvolených atribútov:
  - **ASC** – vzostupne *(ascending)* (ak nie je uvedený spôsob zoradenia, predvolené *(default)* nastavenie je ASC),
  - **DESC** – zostupne *(descending)*.

Jednoduchý príklad na využitie príkazu SELECT je napr. dopyt, ktorým možno zistiť z tabuľky s údajmi o osobách (Obr. 2.3) rok narodenia Márie Veselej:
Dopyt v bežnom jazyku:

> Z tabuľky `Osoby` vyber údaj, ktorý predstavuje rok narodenia osoby s menom Veselá Mária.

Dopyt v jazyku SQL:

```
SELECT rok_narodenia
FROM Osoby
WHERE meno='Veselá Mária';
```

V podmienkach výberu v príkaze SELECT sa na ich bližšiu špecifikáciu často používajú logické operátory, operátory porovnávania alebo špeciálne operátory jazyka SQL[22]. Patria medzi ne napr.:

logické operátory:

- **AND** (a zároveň) – konjunkcia**,**

[22] Uvedený zoznam operátorov jazyka SQL nie je úplný, obsahuje len základné operátory, ktoré sa používajú pri vypracovaní úloh zadaných v rámci skrípt.

- **OR** (alebo) – disjunkcia,
- **NOT** (nie) – negácia,

operátory porovnávania:

=, **<**, **>**, **<=**, **>=**,

<> (alebo symboly **!=**) – nerovná sa,

špeciálne operátory:

- **BETWEEN** – kontroluje, či je hodnota atribútu v stanovených hraniciach,
- **IS NULL** – kontroluje, či je hodnota atribútu prázdna,
- **LIKE** – kontroluje, či hodnota atribútu zodpovedá určitému vzoru,
- **IN** – kontroluje, či hodnota atribútu zodpovedá niektorej hodnote v zozname,
- **EXISTS** – kontroluje, či poddopyt vráti nejaké riadky alebo nie,
- **DISTINCT** – odstraňuje duplicitné hodnoty z výberu.

agregačné funkcie:

- **COUNT (*)** – počet riadkov tabuľky (prvkov relácie),
- **COUNT (atribút)** – počet riadkov s nenulovou hodnotou,
- **MIN (atribút)** – minimálna hodnota nájdená v určenom atribúte,
- **MAX (atribút)** – maximálna hodnota nájdená v určenom atribúte,
- **SUM (atribút)** – súčet všetkých hodnôt v určenom atribúte,
- **AVG (atribút)** – priemerná hodnota všetkých hodnôt určeného atribútu.

operácie relačnej algebry:

- **UNION –** zjednotenie (dvoch alebo viacerých tabuliek[23]),
- **INTERSECT** – prienik tabuliek,
- **EXCEPT** – rozdiel tabuliek.

Všetky uvedené klauzuly, operátory, funkcie a operácie sú na konkrétnych príkladoch vysvetlené napr. v (Ďuračiová, 2014) alebo v (Pokorný, 2000).

[23] Tabuľky, ktoré vstupujú do operácie UNION, musia byť **kompatibilné**, t. j. musia mať rovnaký počet stĺpcov a príslušné stĺpce musia mať rovnaký dátový typ.

Vkladanie nových záznamov do databázy

Na **vkladanie nových záznamov** do databázy (pridávanie nových riadkov do databázových tabuliek) slúži príkaz **INSERT INTO**. Poskytuje dva základné varianty:

- vkladanie dát do všetkých stĺpcov v tabuľke:

```
INSERT INTO Meno_tabulky VALUES (zoznam_hodnot);
```

- vkladanie dát iba do niektorých (vybraných) stĺpcov (napr. dvoch):

```
INSERT INTO Meno_tabulky (meno_stlpca_1, meno_stlpca_2,...)
          VALUES (hodnota1, hodnota2,...);
```

Modifikácia dát v databáze

Na **modifikáciu dát** (obnova záznamu, zmeny v databáze alebo aktualizácia) sa používajú príkazy **UPDATE** a **SET**:

```
UPDATE Meno_tabulky
SET meno_stlpca=nova_hodnota
WHERE podmienka;
```

Klauzula WHERE je v príkaze nepovinná. Ak sa vynechá, budú všetky polia s určeným názvom zmenené na hodnotu za klauzulou SET.

Vymazanie záznamov z databázy

Na **vymazanie záznamov** z databázy (odstránenie vybraných riadkov z tabuľky) slúži príkaz **DELETE FROM**:

```
DELETE FROM Meno_tabulky
WHERE podmienka;
```

2.2.2 Jazyk DDL – príkazy na definíciu dát

Jazyk SQL umožňuje definovať (príkaz **CREATE**) a rušiť (príkaz **DROP**) databázové objekty, medzi ktoré patria samotné databázy, schémy, tabuľky, domény, indexy, pohľady[24], procedúry[25], triggre[26], funkcie a pod.

Vytvorenie a zrušenie schémy

Na **vytvorenie schémy** sa používa príkaz **CREATE SCHEMA**, ktorý môže zahŕňať všetky elementy definície schémy. Schéma predstavuje určitý nadstavbový aparát nad tabuľkami, ktorý je pre väčšinu v rámci SRDB voliteľný. Syntax príkazu je:

```
CREATE SCHEMA Nazov_schemy
AUTHORIZATION Meno_pouzivatela;
```

Za **AUTHORIZATION** sa uvádza meno používateľa, ktorý schému vytvoril.

Zrušenie schémy sa vykonáva pomocou príkazu **DROP SCHEMA**, ktorý má dve voľby:

- **CASCADE** – odstránia sa všetky elementy zvolenej schémy (databázy), t. j. tabuľky, domény, konšrukty a pod.,
- **RESTRICT** – schéma je zrušená len vtedy, ak neobsahuje žiadne elementy.

  Syntax príkazu potom je:

```
DROP SCHEMA Nazov_schemy CASCADE|RESTRICT;
```

[24] **Pohľad** v SQL je virtuálna tabuľka vytvorená nad dopytom SQL. Vytvára sa na požiadanie, keď k nemu pristupuje používateľ. Pohľad je logická tabuľka, ale v skutočnosti (fyzicky) je to predpripravený dopyt s príkazom SELECT, ktorý sa spúšťa vždy pri práci s príslušným pohľadom.

[25] **Procedúra** je skupina SQL príkazov, ktoré možno spoločne spustiť a vykonať. Procedúra môže byť uložená priamo v databáze (uložená procedúra).

[26] **Trigger** (spúšťač, spúšť) je uložená (preddefinovaná) databázová procedúra, ktorá podmienene alebo nepodmienene automaticky nasleduje, nahrádza alebo predchádza databázovú operáciu. Trigger sa môže spustiť len pri vykonávaní príkazov INSERT, DELETE alebo UPDATE.

Vytvorenie, zrušenie a modifikácia tabuľky

Príkaz **CREATE TABLE** slúži na špecifikáciu (deklaráciu) novej tabuľky. Tabuľke sa priradí meno, špecifikujú sa jej atribúty a počiatočné obmedzenia *(constraints)*. Syntax príkazu je:

```
CREATE TABLE Meno_tabulky (
      meno_stlpca_1 datovy_typ [špecifikácie],
      [meno_stlpca_2 datovy_typ [špecifikácie],...]
      PRIMARY KEY (meno_stlpca_1 [meno_stlpca_2,...]),
      FOREIGN KEY (meno_stlpca_1 [meno_stlpca_2,...])
            REFERENCES      Meno_tabulky,
      CONSTRAINT obmedzenia]);
```

Ako súčasť vytvorenia tabuľky možno určiť obmedzenia špecifikácie kľúča, referenčnej integrity, domén atribútov, prázdnych hodnôt a špecifikácie pre individuálne riadky v rámci relácií. Príkazmi na stanovenie obmedzení v rámci príkazu CREATE TABLE sú:

- **PRIMARY KEY** (nazov_primarneho_kluca) – špecifikácia primárneho kľúča (kapitola **2.1**),
- **FOREIGN KEY** (nazov_cudzieho_kluca) – špecifikácia cudzieho kľúča (kapitola **2.1**),
- **REFERENCES** – špecifikácia tabuľky cudzieho kľúča,
- **NOT NULL** – špecifikácia, že stĺpec nemôže obsahovať prázdnu hodnotu,
- **UNIQUE** – špecifikácia, že všetky hodnoty v stĺpci sú jedinečné (unikátne) (tzv. sekundárny kľúč),
- **DEFAULT** – priradenie preddefinovanej hodnoty atribútu pri vytvorení riadku,
- **CHECK** – rozsah kontroly domény (kontroluje, či záznam spĺňa určenú podmienku, napr. či sa hodnota nachádza medzi povolenými hodnotami domény).

Tabuľky sa odstraňujú prostredníctvom príkazu **DROP TABLE**, opäť s dvomi možnosťami voľby:

- **CASCADE** – zrušenie bez špecifikácií,
- **RESTRICT** – zrušenie, len ak tabuľka nemá na nej závislé špecifikácie (pohľady, triggre alebo tabuľku, ktorá je od nej závislá)

Syntax príkazu potom je:

```
DROP TABLE Meno_tabulky CASCADE|RESTRICT;
```

Príkazy na manipuláciu s tabuľkou dopĺňa príkaz **ALTER TABLE**, ktorý sa používa na modifikáciu definície tabuľky. Príkaz umožňuje:

- pridanie a zrušenie stĺpcov,
- zmenu definícií stĺpcov,
- pridanie a zrušenie špecifikácií.

Syntax príkazu je:

```
ALTER TABLE Meno_tabulky
(ADD|MODIFY|DROP) meno_stlpca datovy_typ
[(ADD|MODIFY|DROP) meno_stlpca datovy_typ]);
```

Syntax príkazu na definovanie primárneho kľúča v existujúcej tabuľke je:

```
ALTER TABLE Meno_tabulky
ADD PRIMARY KEY (meno_stlpca);
```

Vytvorenie a odstránenie domény

Na vytvorenie domény hodnôt sa používa príkaz **CREATE DOMAIN**. Jeho základná syntax je:

```
CREATE DOMAIN Meno_domeny AS datovy typ;
```

Preddefinovaná hodnota domény sa nastavuje príkazom **DEFAULT**:

```
CREATE DOMAIN Meno_domeny AS datovy typ
      DEFAULT hodnota;
```

Ak je potrebné obor hodnôt v rámci dátového typu zúžiť, k základnému príkazu možno pridať ďalšie obmedzenia ako napr. **NOT NULL** alebo **CHECK**:

Na zmenu definície domény je určený príkaz **ALTER DOMAIN** a na jej zrušenie príkaz **DROP DOMAIN**.

Vytvorenie a odstránenie indexov, pohľadov a triggrov

Na vytvorenie indexu sa používa príkaz **CREATE INDEX**, na jeho zrušenie príkaz **DROP INDEX**. Na vytvorenie a zrušenie pohľadu sa používajú príkazy **CREATE VIEW** a **DROP VIEW**, na vytvorenie a zrušenie triggra príkazy **CREATE TRIGGER** a **DROP TRIGGER**.

2.3 Tvorba jednoduchých dopytov v jazyku SQL

(2. cvičenie)

Zadanie č. 2.1: V cvičnej modelovej databáze `Vlastnictvo_parciel`[27] (Obr. 2.4) vytvorte a otestujte dopyty v jazyku SQL (úlohy **A1** – **A33**). Zamerajte sa aj na výsledky dopytov, v ktorých skontrolujte ich súlad so slovným zadaním dopytu.

Osoby

id_osoby	meno	rok_narodenia
105	Mrkvička Ján	1965
106	Novák Peter	1984
107	Veselá Mária	1985
108	Nový Pavol	1971
109	Kováč Filip	1991
...		

Parcely

id_parcely	vymera	druh_pozemku
11	10422	orná pôda
22	344	zastavaná plocha
33	987	záhrada
...		

Vlastnictvo

id_parcely	id_osoby	podiel_citatel	podiel_menovatel
11	105	1	2
11	107	1	2
22	106	1	1
33	107	2	5
33	108	3	5
...			

Prenajom

id_parcely	id_osoby
11	106
33	105
...	

Obyvatelia

id_obyvateľa	meno	adresa
105	Mrkvička Ján	Zelená 4, 976 64 Beňuš
107	Veselá Mária	Ružová 5, 976 64 Beňuš
209	Rýchly Peter	Tehelná 12, 949 01 Nitra
210	Kováčová Eva	Vysoká 13, 010 01 Žilina
...		

Najomnici

id_osoby	meno	rok_narodenia
303	Adamová Zuzana	1983
304	Biely Miroslav	1988
...		

Obr. 2.4. Modelová databáza `Vlastnictvo_parciel` reprezentovaná tabuľkami

[27] Súbor s databázou `Vlastnictvo_parciel` pre softvérové prostredie MS Access sa nachádza v prílohe č. 1., pre databázový systém PostgreSQL v prílohe č. 2.

Úlohy:

Selekcia a projekcia[28]

A1: Zistite mená osôb (projekcia) narodených najneskôr v roku 1985 (selekcia).

Zjednotenie

A2: Nájdite parcely, ktoré vlastní osoba s číslom 105 alebo osoba s číslom 107.

Prienik

A3: Nájdite parcely, ktoré vlastní osoba s číslom 105 a zároveň osoba s číslom 107.

Rozdiel

A4: Nájdite parcely, ktoré vlastní osoba s číslom 107, ale nevlastní ich osoba s číslom 105.

Jednoduché spojenie tabuliek

A5: a) Zistite, kto vlastní ktorú parcelu a do výsledku vypíšte aj všetky dostupné informácie o vlastníkoch jednotlivých parciel.

Eliminácia (duplicitných) stĺpcov a zoradenie výsledku (klauzula ORDER BY)

b) Vypíšte výsledok predchádzajúceho dopytu tak, aby neobsahoval duplicitné stĺpce a ani informácie o vlastníckych podieloch. Výsledok zoraďte podľa atribútu `id_parcely` vzostupne a podľa atribútu `rok_narodenia` zostupne (od najväčšieho po najmenší).

Karteziánsky súčin

A6: Vytvorte karteziánsky súčin `Vlastnictvo` × `Parcely`

Projekcia – výpis obsahu vybraných stĺpcov (atribútov)

A7: a) Vypíšte tabuľku mien osôb a rokov ich narodenia.

[28] Názvy úloh zodpovedajú pomenovaniu dopytov v (Ďuračiová, 2014) a v (Pokorný, 2000), kde sú analogické príklady aj vysvetlené. Pri samostatnom štúdiu skrípt bez nadväznosti na (Ďuračiová, 2014) je vhodné najprv vypracovať zadanie č. 2.2, v ktorom je postup riešenia čiastočne naznačený, a až potom samostatne vypracovať zadanie č. 2.1, ktoré v tom prípade slúži na upevnenie si vedomostí z jazyka SQL. Zadania je potom vhodné vypracovať v poradí: 2.2, 2.3, 2.4, 2.1.

Projekcia s elimináciou duplicitných riadkov

b) Vypíšte tabuľku mien osôb a rokov ich narodenia s elimináciou duplicitných riadkov.

Projekcia s usporiadaním tabuľky

c) Vypíšte tabuľku mien osôb a rokov ich narodenia s usporiadaním tabuľky podľa mena zostupne.

Projekcia bez vertikálnej reštrikcie (zahrnutie všetkých stĺpcov vo výsledku)

A8: Vypíšte z tabuľky `Osoby` všetky dostupné informácie o osobách, ktoré sa narodili pred rokom 1985.

Selekcia s logickými operátormi AND a OR

A9: Vypíšte z tabuľky `Osoby` informácie o osobách, ktoré sa narodili v rokoch 1980 až 1990.

Selekcia s logickým operátorom NOT (negácia)

A10: Vypíšte z tabuľky `Osoby` tie osoby, ktoré sa nenarodili v rokoch 1980 až 1990.

Kombinácia spojenia s viacnásobnou reštrikciou

A11: Nájdite tie parcely a mená ich vlastníkov, ktoré vlastnia osoby narodené pred rokom 1970 alebo po roku 1980, ale vlastníkom ktorých nie je Peter Novák a ani osoba s číslom 109.

Vnorené dopyty (poddopyty)

A12: a) Vyberte tie parcely, ich výmery a mená ich vlastníkov, ktoré vlastnia aspoň dve osoby. Výsledok zoraďte podľa čísel parciel.

Lokálne premenovania (aliasy) tabuliek

b) Napíšte dopyt A12 (s vnoreným dopytom) s využitím lokálnych premenovaní (aliasov).

Klauzula GROUP BY

A13: Zistite pre každú osobu počet parciel, ktoré vlastní.

Klauzula HAVING

A14: Zistite pre každú osobu, ktorá vlastní aspoň dve parcely, počet parciel, ktoré vlastní.

A15: Zistite pre každú osobu, ktorá je narodená po roku 1980 a vlastní aspoň dve parcely, počet parciel, ktoré vlastní.

Klauzula ORDER BY

A16: Zoraďte osoby, ktoré sa narodili pred rokom 1990 podľa roku narodenia zostupne.

Predikát LIKE (NOT LIKE)

A17: Vypíšte mená osôb, ktoré sa začínajú písmenom N.

A18: Vypíšte mená osôb, ktoré majú krstné meno Ján.

Aritmetické operácie v príkaze SELECT

A19: Vypíšte pre všetky osoby ich vek, ktorý dosiahli v roku 2014.

Výraz CASE – podmienené hodnotové výrazy

A20: Je druh pozemku niektorej parcely záhrada?

Predikát IN

A21: a) Zistite všetky druhy pozemkov parciel, ktoré vlastní osoba s číslom 107.

A22: Zistite všetky osoby, ktoré vlastnia parcely s číslom 22 alebo 33.

Predikáty ALL, SOME a ANY

A23: Zistite čísla a mená osôb, ktoré sú mladšie (t. j. majú neskorší rok narodenia) ako aspoň jedna osoba, ktorej meno sa začína textovým reťazcom 'Nov'.

A24: Zistite čísla a mená osôb, ktoré sú mladšie ako ktorákoľvek osoba, ktorej meno sa začína textovým reťazcom 'Nov'.

A25: Ktorá z osôb, ktorej meno obsahuje textový reťazec 'ov' je najstaršia?

Operátory relačnej algebry – UNION, INTERSECT, EXCEPT

A26: Zistite čísla osôb, ktoré vlastnia alebo majú prenajatú parcelu číslo 11.

Výraz AS – aliasy stĺpcov

A27: Vytvorte (vyselektujte) tabuľku, ktorá bude obsahovať stĺpec `cp` obsahujúci čísla parciel a stĺpec `vym` s výmerami prepočítanými z m^2 na hektáre (ha).

Spojenie tabuľky samej so sebou

A28: Nájdite dvojice osôb (reprezentovaných identifikačnými číslami), ktoré majú rovnaký rok narodenia.

Spojenie tabuliek – SELECT + JOIN

A29: Doplňte adresy z tabuľky `Obyvatelia` do tabuľky `Osoby` (vytvor spojenie), ak:

a) výsledkom sú len tie záznamy, ktoré spĺňajú podmienku spojenia (uvedenú za klauzulou ON):

b) výsledkom sú všetky riadky z pravej tabuľky (`Osoby`) (aj keď neobsahujú zodpovedajúce dáta v ľavej tabuľke (`Obyvatelia`)) a k ním zodpovedajúce hodnoty z ľavej tabuľky,

c) výsledkom sú všetky riadky z ľavej tabuľky (`Obyvatelia`) a k nim zodpovedajúce hodnoty z pravej tabuľky (`Osoby`),

d) výsledkom sú spojené riadky, ale aj ostatné hodnoty z obidvoch tabuliek.

Eliminácia duplicity stĺpcov vo výsledku spojenia

e) Napíšte dopyt A29 s elimináciou duplicitných stĺpcov vo výsledku.

Rôzne možnosti formulácie dopytov

A21: b) až **h)** Vytvorte dopyt A21 (Nájdi všetky druhy pozemkov parciel, ktoré vlastní osoba číslo 107) nasledujúcimi rôznymi spôsobmi:

b) variant s operáciou jednoduchého spojenia v zložke WHERE,

c) variant s operáciou prirodzeného spojenia v zložke FROM,

d) variant s operáciou vnútorného spojenia s podmienkou spojenia v klauzule ON,

e) variant s predikátom IN a vnoreným dopytom,

f) variant s predikátom = ANY,

g) variant s predikátom EXISTS,

h) variant s predikátom IN a zložitejším vnoreným dopytom.

Prázdne hodnoty v jazyku SQL

A30: Nájdite osoby, ktoré majú v tabuľke vytvorenej plným spojením tabuliek `Osoby` a `Obyvatelia` (podmienkou spojenia je rovnaká hodnota atribútu `id_osoby`) v stĺpci `adresa` prázdnu hodnotu.

Vkladanie záznamov do tabuľky pomocou príkazu SELECT

A31: Vložte do tabuľky `Osoby` tie záznamy z tabuľky `Najomnici`, ktoré v nej ešte nie sú uložené.

Modifikácia záznamov v databáze (príkaz UPDATE – SET)

A32: V tabuľke `Parcely` zmeňte hodnotu výmery parcely č. 33 na hodnotu 989.

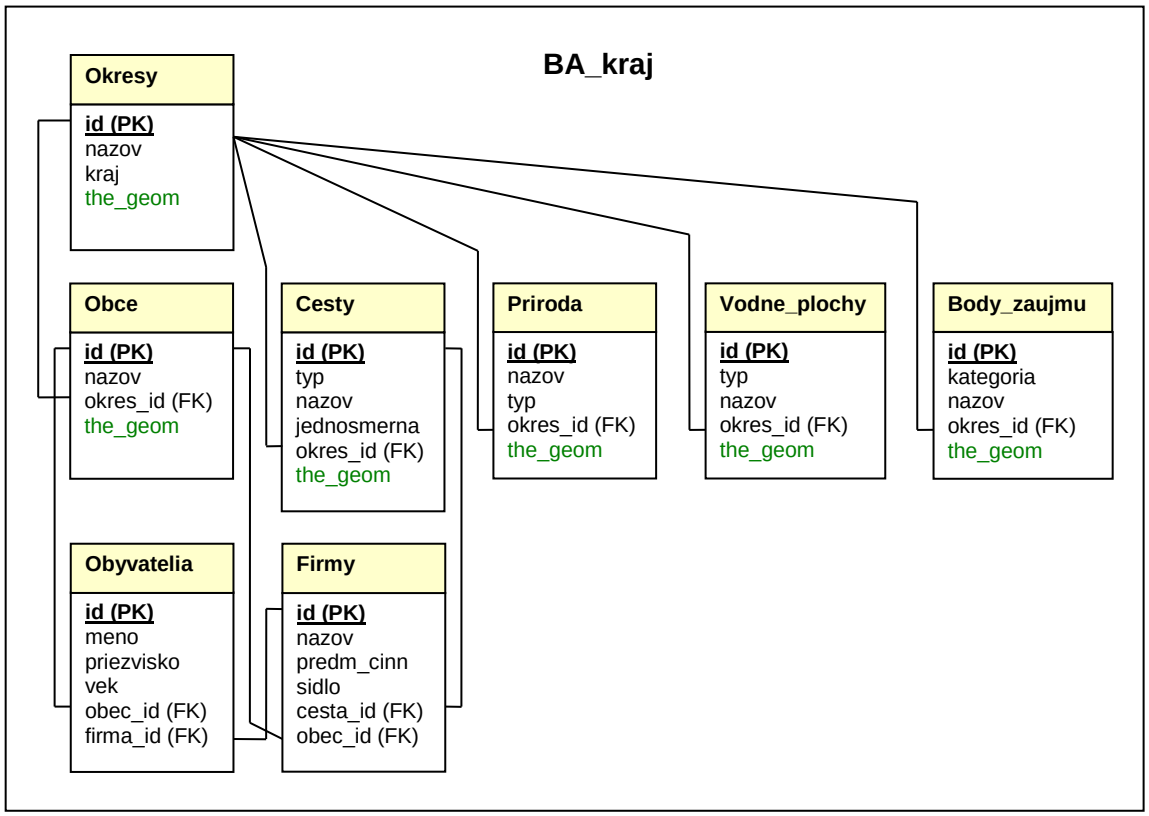
Vymazanie záznamov z databázy (príkaz DELETE FROM)

A33: Z tabuľky `Osoby` vymažte záznamy, ktoré boli do nej vložené v dopyte **A31** (`id_osoby`=303 a `id_osoby`=304).

2.4 Jazyk SQL v databázovom systéme PostgreSQL – príkazy jazyka DML

(3. a 4. cvičenie)

Zadanie č. 2.2: V softvérovom prostredí PostgreSQL vytvorte dopyty (úlohy **B1** – **B42**) v jazyku SQL s využitím príkazov jazyka DML pre modelovú databázu `BA_kraj`[29] (Obr. 2.8).



Obr. 2.5. Schéma modelovej databázy `BA_kraj`[30] v prostredí PostgreSQL

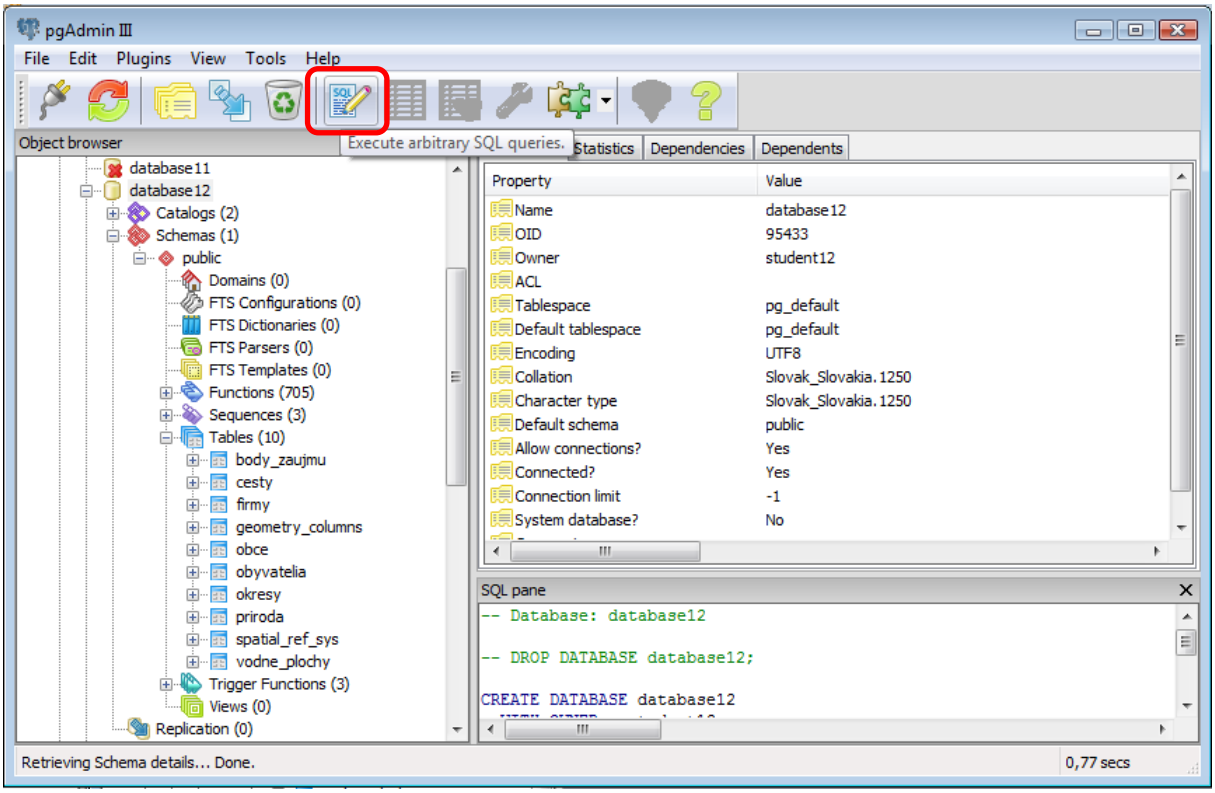
Postup riešenia:

1. V prostredí PGAdmin otvorte okno „SQL Editor“ (pomocou ikony „Execute arbitrary SQL queries“ (Obr. 2.6).

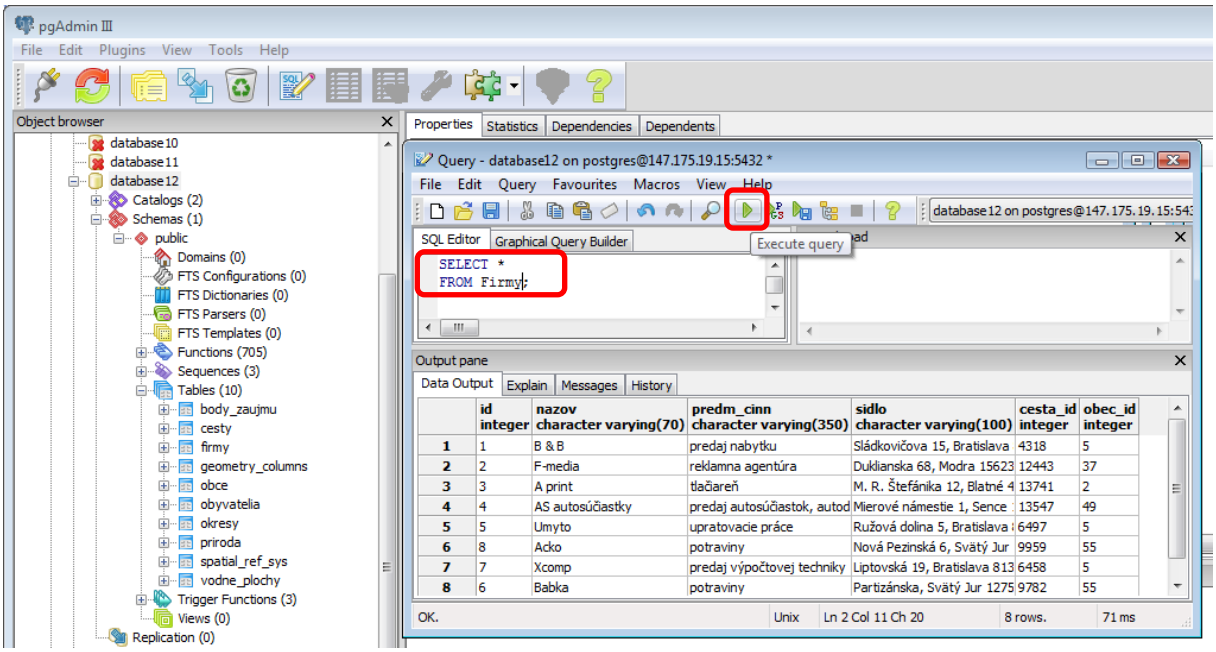
[29] Súbor s kompletnou databázou sa nachádza v prílohe č. 3. Postup tvorby záložnej kópie databázy, exportu a importu dát je uvedený v kapitole **4**.

[30] V schéme modelovej databázy je použité nasledujúce označenie: 1. názvy tabuliek sú uvádzané s veľkým začiatočným písmenom; 2. atribúty, ktoré tvoria primárny kľúč relácie sú podčiarknuté a označené skratkou PK *(Primary Key)*; 3. atribúty, ktoré tvoria cudzí kľúč, sú označené skratkou FK *(Foreign Key)*. Podrobnejšie je označeniu schémy databázy venovaná kapitola **5**.

2. Do okna „SQL editor“ napíšte dopyt v jazyku SQL (úlohy **B1** – **B42**) a vykonajte ho prostredníctvom ikony „Execute query“ (Obr. 2.7).



Obr. 2.6. Otvorenie okna na tvorbu dopytov v jazyku SQL



Obr. 2.7. Tvorba a vykonanie dopytov v jazyku SQL v prostredí PostgreSQL

Niektoré dopyty v úlohách **B1** – **B42** , najmä tie, ktoré si vyžadujú aplikáciu nového prístupu, operátora alebo funkcie, majú v rámčeku pri zadaní konkrétnej úlohy uvedenú nápovedu, ostatné je potrebné vytvoriť samostatne.

Úlohy:

B1: Vyselektujte všetky údaje z tabuľky `Firmy`.

```
SELECT čo
FROM nazov_tabulky;
```

B2: Vyselektujte všetky údaje z tabuľky `Firmy`, kde atribút `predm_cinn` je '`upratovacie práce`'.

```
SELECT čo
FROM nazov_tabulky
WHERE stlpec='hodnota';
```

B3: Vyselektujte `meno`, `priezvisko` a `vek` zo všetkých záznamov z tabuľky `Obyvatelia`, v ktorých `meno` je Peter a vek je do 35 rokov.

```
SELECT stlpec1, stlpec2, stlpec3
FROM nazov_tabulky
WHERE stlpec='hodnota' AND stlpec<'hodnota';
```

B4: Vyselektujte `meno`, `priezvisko` a `vek` všetkých záznamov z tabuľky `Obyvatelia`, kde `meno` je Pavol alebo Ivan, s tým, že vo výpise premenujete názov stĺpca `meno` na `krstne_meno`.

```
SELECT stlpec1 AS novy_nazov, stlpec2, stlpec3
FROM nazov_tabulky
WHERE stlpec3='hodnota' OR stlpec3='hodnota';
```

B5: Vyselektujte všetky záznamy z tabuľky `Obyvatelia`, kde `meno` sa začína písmenom „J”.

```
SELECT čo
FROM nazov_tabulky
WHERE stlpec LIKE 'hodnota%';
```

B6: Vyselektujte všetky záznamy z tabuľky `Obyvatelia`, kde `meno` sa začína písmenom „J” a dĺžka mena je 5 znakov.

```
SELECT čo
FROM nazov_tabulky
WHERE stlpec LIKE 'hodnota____';
```

B7: Vyselektujte `priezvisko`, `meno`, `id` a `vek` v mesiacoch z tabuľky `Obyvatelia`, a výpis zoraďte podľa stĺpca `priezvisko`.

```
SELECT stlpec1, stlpec2, stlpec3*12 AS novy_nazov
FROM nazov_tabulky
ORDER BY stlpec1;
```

B8: Vyselektujte priemerný `vek` z tabuľky `Obyvatelia`.

```
SELECT AVG(stlpec)
FROM nazov_tabulky;
```

B9: Zistite (vyselektujte) `nazov obce`, v ktorej býva Peter Oravec.

```
SELECT tab1.stlpec
FROM tab1, tab2
WHERE stlpec='Peter' AND stlpec2='Oravec'
     AND tab1.kluc=tab2.kluc;
```

B10: Vyselektujte všetkých `obyvatelov`, ktorí bývajú v `obci` Blatné.

B11: Vyselektujte záznamy s rovnakým `nazvom` z tabuliek `Priroda` a `Vodne_plochy`.

```
(SELECT čo
FROM tabulka1)
INTERSECT
(SELECT čo
FROM tabulka2);
```

B12: Vyselektujte `nazvy` objektov z tabuľky `Priroda`, ktoré sa nenachádzajú v tabuľke `Vodne_plochy`.

```
(SELECT čo
FROM tabulka1)
EXCEPT
(SELECT čo
FROM tabulka2);
```

B13: Vyselektujte všetky `obce`, ktoré sú v okrese Pezinok.

```
SELECT čo
FROM tabulka1
WHERE kluc1=(SELECT kluc2
                FROM tabulka2
                WHERE atribut='Pezinok');
```

B14: Zistite počet riadkov v tabuľke `Obyvatelia`.

```
SELECT COUNT(*)
FROM tabulka;
```

B15: Vyselektujte `mená` z tabuľky `Obyvatelia` tak, aby bolo každé meno vypísané iba raz.

```
SELECT stlpec
FROM tabulka
GROUP BY stlpec;
```

B16: Vyselektujte `mená` a počet ich výskytov z tabuľky `Obyvatelia`.

```
SELECT stlpec, COUNT(stlpec)
FROM tabulka
GROUP BY stlpec;
```

B17: Vyselektujte `meno`, `priezvisko` a `vek` troch najstarších obyvateľov.

```
SELECT čo
FROM tabulka
ORDER BY stlpec DESC;
LIMIT 3;
```

B18: Vložte do tabuľky `Obyvatelia` hodnoty `meno`: Martin, `priezvisko`: Králik, `vek`: 37, `obec_id`: 34 `firma_id`: 7.

```
INSERT INTO tabulka(stlpec1, stl2, stl3, stl4, stl5)
VALUES ('hodnota','hodn','hodn','hodn','hodn');
```

B19: Vyselektujte všetky údaje z tabuľky `Obyvatelia`, kde `id` je väčšie ako 152.

B20: Vymažte záznam z tabuľky `Obyvatelia` s menom Martin Králik.

```
DELETE FROM tabulka
WHERE stlpec1=hodnota AND stlpec2=hodnota;
```

B21: Vložte do tabuľky `Firmy` ďalší záznam.

B22: Aktualizujte záznam v tabuľke `Obyvatelia`. Obyvateľovi s `id`=119 zadajte hodnotu `firma_id`=5.

```
UPDATE tabulka
SET stlpec=hodnota
WHERE stlpec=hodnota;
```

B23: Vyselektujte záznam z tabuľky `Obyvatelia`, ktorý ste práve aktualizovali.

B24: Vymažte záznamy z tabuľky `Obyvatelia`, kde `id` je v intervale od 134 do 137.

B25: Vložte do tabuľky `Body_zaujmu` hodnoty `nazov`, `okres_id` a `the_geom` z tabuľky `Obce`.

```
INSERT INTO tabulka1 (stlp1, stlp2, stlp3)
SELECT stlp1, stlp2, stlp3
FROM tabulka2;
```

B26: Vyselektujte všetky záznamy z tabuľky `Obyvatelia`, ktoré majú rovnaké `meno` a rovnaký `vek`.

```
SELECT *
FROM tabulka a, tabulka b
WHERE a.stlpec=b.stlpec AND .......;
```

B27: Sčítajte vek všetkých obyvateľov z tabuľky obyvatelia.

B28: Vyselektujte všetkých obyvateľov pracujúcich vo firme s názvom Xcomp.

B29: Pomocou jedného príkazu zistite, v ktorom okrese býva Imrich Struhár.

B30: Vyselektujte všetky cesty, ktoré sa nachádzajú v okrese Senec a obsahujú v názve reťazec „ale“.

B31: Vyselektujte obce a okresy, ktoré majú rovnaký názov.

B32: Obyvateľovi Vojtechovi Kováčikovi zvýšte vek o 5 rokov.

B33: Do tabuliek `Firmy` a `Obyvatelia` vložte po dva nové záznamy.

B34: Vyselektujte mená a priezviská obyvateľov zoradené podľa priezviska od Z po A.

B35: Vyselektujte mená a priezviská obyvateľov zoradené podľa priezviska od A po Z.

B36: Vyselektujte priemerný vek obyvateľov starších ako 18 rokov.

B37: Vyselektujte všetky okresy okrem okresu Senec.

B38: Vyselektujte 30. až 40. záznam z tabuľky `Obyvatelia`.

B39: Vyselektujte všetky záznamy z tabuľky `Obce`, s tým že zľava pripojíte tabuľku `Okresy` na základe rovnajúcich sa kľúčov `Obce.okres_id` a `okresy.id`.

B40: Zistite v ktorých obciach bývajú obyvatelia z tabuľky `Obyvatelia`.

B41: Zistite v ktorých obciach nebývajú obyvatelia z tabuľky `Obyvatelia`.

B42: Zistite vek najmladšieho obyvateľa.

2.5 Jazyk SQL v databázovom systéme PostgreSQL – príkazy jazyka DDL

(5. cvičenie)

Zadanie č. 2.3: V softvérovom prostredí PostgreSQL vytvorte dopyty (úlohy **B43** – **B71**) v jazyku SQL s využitím príkazov jazyka DDL do modelovej databázy `BA_kraj` definovanej v predchádzajúcom zadaní ( ).

Úlohy:

B43: Vytvorte tabuľku `Zamestnanci`, ktorá bude obsahovať stĺpce `id`, `meno`, `priezvisko`, `pohlavie`, `dat_nar`, `firma_id`. Primárny kľuč bude stĺpec `id`.

```
CREATE TABLE nazov(
stlpec1 dat_typ1,
stlpec2 dat_typ2,
stlpec3 dat_typ3,
stlpec4 dat_typ4,
PRIMARY KEY (stlpec1)
);
```

Základné dátové typy v PostgreSQL sú:

- `INT` – celé číslo,
- `VARCHAR(d)` – reťazec do dĺžky `d`,
- `CHAR(d)` – reťazec dĺžky `d`,
- `FLOAT` – číslo s pohyblivou desatinnou čiarkou,
- `DECIMAL(d,n)` – reálne číslo s dĺžkou `d` a s `n` desatinnými miestami,
- `DATE` – dátum,
- `TIME` – čas,

B44: Vytvorte tabuľku `Tovar`, ktorá bude obsahovať stĺpce `nazov`, `popis`, `sklad`, `cena`. Primárny kľúč budú stĺpce `nazov` a `popis`.

```
CREATE TABLE nazov(
stlpec1 dat_typ1,
stlpec2 dat_typ2,
stlpec3 dat_typ3,
stlpec4 dat_typ4,
PRIMARY KEY (stlpec1,stlpec2)
);
```

B45: Do tabuľky `Tovar` pridajte stĺpec `pocet` s dátovým typom `INT`.

```
ALTER TABLE tabulka ADD stlpec dat_typ;
```

B46: Z tabuľky `Tovar` vymažte stĺpec `sklad`.

```
ALTER TABLE tabulka DROP stlpec;
```

B47: Vymažte tabuľku `Tovar`.

```
DROP TABLE tabulka;
```

B48: Vytvorte tabuľku `Produkty`, ktorá bude obsahovať stĺpce `id`, `nazov`, `typ`, `popis`, `cena`. Stĺpcu `typ` nastavte preddefinovanú hodnotu „základný“.

```
CREATE TABLE nazov(
stlpec1 dat_typ1,
stlpec2 dat_typ2 DEFAULT 'základný',
stlpec3 dat_typ3,
stlpec4 dat_typ4,
PRIMARY KEY (stlpec1)
);
```

B49: Do tabuľky `Produkty` vložte aspoň 3 záznamy tak, že stĺpec typ necháte prázdny.

B50: Vyselektujte všetko z tabuľky produkty.

B51: Vytvorte tabuľku `Objednavky` so stĺpcami `cislo_ob`, `datum`, `produkt_id`, `doprava`, `pocet`, `cena`. Stĺpec `doprava` môže obsahovať iba 3 hodnoty: „osobny odber“, „posta“ a „kurier“.

```
CREATE TABLE nazov (
stlpec1 dat_typ1,
stlpec2 dat_typ2,
stlpec3 dat_typ3,
PRIMARY KEY (stlpec1),
CHECK (stlpec2 IN ('hodn1','hodn2','hodn3'))
);
```

B52: Stĺpec `produkt_id` z tabuľky `Objednavky` nastavte ako cudzí kľúč pre referenčnú tabuľku `Produkty` a jej stĺpec `id`.

```
ALTER TABLE nazov
ADD CONSTRAINT nazov_kluca
FOREIGN KEY (lokal_stl) REFERENCES ref_tab (ref_stl);
```

B53: Vložením údajov do tabuľky `Objednavky` si overte, že cudzí kľúč je zadaný správne.

B54: Vytvorte doménu s názvom `Pohlavie_dom`, použiteľnú pri definovaní novej tabuľky, ktorá bude obsahovať stĺpec `pohlavie`. Takýto stĺpec v našom prípade bude môcť obsahovať tri hodnoty (`neznáme`, `muž`, `žena`) z toho predvolená hodnota bude `neznáme`.

```
CREATE DOMAIN nazov_domeny AS dat_typ
DEFAULT 'predvolena_hodn'
CHECK (VALUE IN ('hodn1','hodn2','hodn3'));
```

B55: Vytvorte tabuľku, v ktorej použijete vytvorenú doménu `Pohlavie_dom` pre stĺpec `pohlavie`.

```
CREATE TABLE nazov(
stlpec1 dat_typ1,
stlpec2 dat_typ2,
stlpec3 domena,
stlpec4 dat_typ4,
PRIMARY KEY (stlpec1)
);
```

B56: Vkladaním údajov do vytvorenej tabuľky sa presvedčte, že hodnoty pre stĺpec `pohlavie` môžu byť iba tri a prednastavená hodnota býva taká, ako sme ju definovali.

B57: Vytvorte doménu `PSC_dom`, ktorá bude môcť byť použitá pri definícii stĺpca s dátovým typom `INT` dĺžky 5.

B58: Pridajte doméne `PSC_dom` také obmedzenie, aby údaje v stĺpcoch mali dĺžku práve 5.

```
ALTER DOMAIN nazov_dom ADD CONSTRAINT
nazov_obmedz CHECK (char_length(VALUE) = 5);
```

B59: Vytvorte dve nové ľubovoľné tabuľky, ktoré budú obsahovať minimálne 5 stĺpcov. Pri ich definícii použite doménu `PSC_dom`. Tabuľky prepojte pomocou cudzieho kľúča.

B60: Do každej tabuľky vložte minimálne 3 záznamy.

B61: Vytvorte ľubovoľnú doménu.

B62: Vymažte doménu vytvorenú v rámci úlohy **B61**.

```
DROP DOMAIN nazov_domeny;
```

B63: Vytvorte pohľad z tabuľky `Produkty`, ktorý bude obsahovať stĺpce `id`, `nazov` a `cena`.

```
CREATE VIEW nazov_pohl AS
SELECT čo
FROM tabulka;
```

B64: Vytvorte pohľad, ktorého obsahom budú obyvatelia starší ako 17 rokov.

B65: Modifikujte pohľad vytvorený v rámci úlohy **B64**, tak, aby obsahoval obyvateľov starších ako 17 a mladších ako 65 rokov.

```
CREATE OR REPLACE VIEW nazov_pohl AS
SELECT čo
FROM tabulka
WHERE podmienky... ;
```

B66: Vymažte pohľad vytvorený v rámci úlohy **B63**.

```
DROP VIEW nazov_pohladu;
```

B67: Vytvorte pohľad, ktorý bude obsahovať obyvateľov z obce Most pri Bratislave.

B68: Vytvorte pohľad, ktorý bude obsahovať 2 stĺpce `Obce.nazov` a `Okresy.nazov`.

B69: Aktualizujte pohľad vytvorený v rámci úlohy **B68** tak, že v ňom zoradíte záznamy podľa názvu okresu a potom podľa názvu obce.

B70: Aktualizujte pohľad zmenený v rámci úlohy **B69** tak, že v ňom zoradíte záznamy podľa dĺžky názvu obce od najkratšieho po najdlhší. Funkcia určujúca dĺžku reťazca je `char_length()`.

B71: Vymažte tabuľku `Objednavky`.

2.6 Vytvorenie modelovej databázy v PostgreSQL

(6. cvičenie)

Zadanie č. 2.4: Vytvorte a potom postupne modifikujte (podľa príkazov v úlohách **A34** – **A42**) modelovú databázu `Vlastnictvo_parciel` zo zadania č. 2.1 v softvérovom prostredí databázového systému PostgreSQL. Schéma databázy je:

`Osoby` (`id_osoby`, `meno`, `rok_narodenia`)
{údaje o osobách vlastniacich parcely}
`Parcely` (`id_parcely`, `vymera`, `druh_pozemku`)
{údaje o parcelách}
`Vlastnictvo` (`id_parcely`, `id_osoby`, `podiel_citatel`, `podiel_menovatel`)
{údaje o vlastníctve parciel}
`Prenajom` (`id_parcely`, `id_osoby`)
{údaje o prenájme parciel}
`Obyvatelia` (`id_obyvatela`, `meno_obyvatela`, `adresa`)
{údaje o obyvateľoch}
`Najomnici` (`id_osoby`, `meno`, `rok_narodenia`)
{údaje o nájomníkoch}

Postup riešenia:

1. Vytvorte databázu (schému) `Vlastnictvo_parciel` pomocou príkazu CREATE DATABASE / CREATE SCHEMA).
2. Vytvorte všetky tabuľky s uvedenými atribútmi (s vhodnými dátovými typmi) a definujte v nich primárne kľúče. Výsledná databáza vytvorená v prostredí PostgreSQL je zobrazená na Obr. 2.8.

Úlohy:

Vytvorenie tabuľky (príkaz CREATE TABLE)

A34: Vytvorte tabuľku `Osoby` s atribútmi `id_osoby`, `meno` a `rok_narodenia`, ktorej primárny kľúč bude atribút `id_osoby`.

Podobným spôsobom je potrebné vytvoriť všetky tabuľky modelovej databázy.

Modifikácia tabuľky (príkaz ALTER TABLE)

A35: Pridajte do tabuľky `Parcely` stĺpec `bonita`, ktorého hodnoty budú celočíselné (dátový typ INT).

A36: Zrušte v tabuľke `Parcely` stĺpec `bonita`.

A37: Vytvorte tabuľku `Budovy` s atribútmi `id_budovy` a `vymera` a potom definujte jej primárny kľúč `id_budovy`.

Domény hodnôt (príkaz CREATE DOMAIN)

A38: Vytvorte doménu `Druh_pozemku`, ktorá bude obsahovať len hodnoty kódov druhov pozemkov: 2, 3, 4, 5, 6, 7, 10, 11, 13, 14.

A39: Vytvorte doménu `Vymera`, ktorá bude obsahovať len kladné čísla.

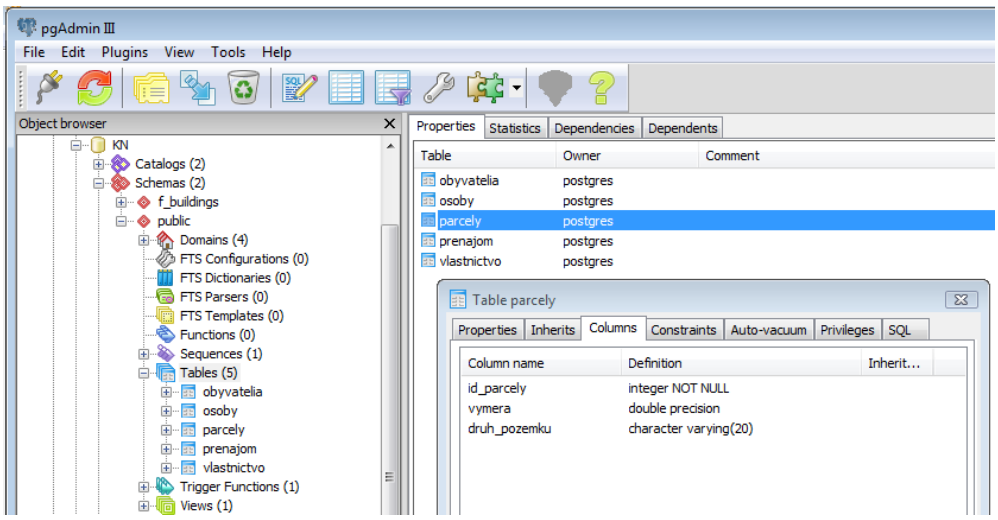
A40: Vytvorte doménu `Druh_stavby`, ktorá bude obsahovať len celé čísla z intervalu ⟨1, 23⟩.

Indexy (príkaz CREATE INDEX)

A41: Vytvorte zložený index obsahujúci meno a rok narodenia v tabuľke `Osoby`.

Pohľady (príkaz CREATE VIEW)

A42: Vytvorte pohľad `Obyvatelia_BA` z tabuľky `Obyvatelia`, ktorý bude obsahovať len dáta o obyvateľoch Bratislavy.



Obr. 2.8. Modelová databáza `Vlastnictvo_parciel` v prostredí PostgreSQL

3 Priestorové databázy

(7. a 8. cvičenie)

Teoretické minimum:

3.1 Priestorové a geografické dáta

Priestorové dáta *(spatial data)* sú dáta, ktoré poskytujú informácie a poznatky vzťahujúce sa k určitým miestam v priestore (miestam ich vzniku alebo miestam ich použitia) a pre ktoré sú zároveň na potrebnej úrovni rozlíšenia známe lokalizácie týchto miest. Obsahujú formálnu priestorovú referenciu a spravidla bývajú vyjadrené pomocou geometrickej a topologickej informácie. **Priestorové databázy** sú databázy, v ktorých sú uložené aj priestorové dáta (resp. informácie o objektoch reálneho sveta reprezentované priestorovými dátami). Ak sú priestorové dáta vztiahnuté k Zemi, nazývajú sa **geografické dáta** alebo **geodáta** a priestorové databázy sa potom označujú ako **geografické databázy**[31] alebo **geodatabázy**.

V súčasnosti sa vyvíjajú a postupne implementujú do objektovo relačných databázových systémov (kapitola **2.1**) aj štandardy definujúce priestorové a geografické dáta. OGC definuje priestorové objekty v štandarde SFA *(Simple Feature Access)* (OGC 06-103r4, 2011), (OGC06-104r4, 2010), ktorý sa stal základom noriem STN EN ISO 19125-1[32] a STN EN ISO 19125-2[33]. Základné priestorové dátové typy spolu s ich stručným opisom sú uvedené v Tabuľka 3.1.

[31] Niekedy sa výraz **geografické databázy** používa ako synonymum pojmu priestorové databázy, ale z vyššie uvedeného vyplýva, že pojem priestorové databázy má všeobecnejší význam.

[32]STN EN ISO 19125-1 Geografická informácia. Prístup k jednoduchým objektom. Časť 1: Všeobecná architektúra (ISO 19125-1:2004 Geographic information — Simple feature access — Part 1: Common architecture)

[33]STN EN ISO 19125-2 Geografická informácia. Prístup k jednoduchým objektom. Časť 2: SQL alternatíva (ISO 19125-2:2004 Geographic information — Simple feature access — Part 2: SQL option)

Tabuľka 3.1. Základné priestorové dátové typy

Dátový typ	Charakteristika
Point	bod (definovaný prostredníctvom súradníc)
Line	líniový reťazec tvorený dvoma bodmi
LineString	líniový reťazec tvorený aspoň dvoma bodmi
LinearRing	uzavretá línia (podmienkou je totožnosť začiatočného a koncového bodu)
Polygon	polygón tvorený líniami a bodmi (je uzavretý)
MultiPoint	objekt zložený z viacerých bodov
MultiLineString	zložený líniový reťazec
MultiPolygon	zložený polygón
Surface	plocha (plošný objekt)
PolyhedralSurface	mnohosten (podtriedou je TIN model)

Niektoré objektovo relačné databázové systémy (napr. PostgreSQL s nadstavbou PostGIS (kapitola **1.2.1**), SQL Server (od verzie SQL Server 2008) (kapitola **1.2.4**), Oracle (Oracle Spatial and Graph) (kapitola **1.2.3**) a IBM DB2 (DB2 Spatial Extender)) podporujú priestorové dátové typy a poskytujú voliteľné rozšírenia a nadstavby určené na spracovanie priestorových dát. Priestorové dátové typy sa môžu v jednotlivých databázových systémoch značne odlišovať[34], avšak dátové typy ako napr. *Point*, *Line* a *Polygon* obsahujú všetky databázové systémy (alebo ich rozšírenia) podporujúce priestorové dáta.

V súčasných databázových systémoch sa priestorové dátové typy rozdeľujú do dvoch základných kategórií, ktorými sú skupiny dátových typov *Geometry* a *Geography*. Geometrické dátové typy *(Geometry)* umožňujú pracovať s priestorovými objektmi len v rámci pravouhlých súradníc v rovine. Novšie (neskôr implementované) dátové typy kategórie *Geography* už podporujú aj ukladanie a spracovávanie priestorových dát v zemepisných súradniciach.

[34] Keďže vývoj prostriedkov na správu a spracovanie priestorových dát v databázach sa veľmi rýchlym tempom vyvíja, podporu dátových typov a ich podrobnú charakteristiku je vhodné si vždy overiť v dokumentácii aktuálnej verzie databázového systému.

Z voľne dostupných open source databázových systémov poskytuje v súčasnosti najširšiu podporu priestorových dátových typov práve databázový systém PostgreSQL. Na prácu s priestorovými a geografickými dátami je určený voľne dostupný modul PostGIS (kapitola **1.2.1**), ktorý podporuje vektorové dátové typy definované podľa špecifikácie OGC a tiež normy ISO 19125-1 (napr. bod, línia, polygón alebo súbor geometrických elementov). Vybrané priestorové dátové typy podporované v PostgreSQL spolu s ich opisom sú uvedené v Tabuľka 3.2. Podrobnejšie sú priestorové dáta a priestorové databázy opísané v (Koreň, 2009) alebo v (Ďuračiová, 2014).

Tabuľka 3.2. Geometrické dátové typy v PostgreSQL (podľa: http://www.postgresql.org)

Meno	Veľkosť	Charakteristika	Reprezentácia
Point	16 bajtov	bod v rovine	(x,y)
Line	32 bajtov	línia	((x1,y1),(x2,y2))
Lseg	32 bajtov	líniový segment	((x1,y1),(x2,y2))
Box	32 bajtov	obdĺžnik (ktorého osi sú rovnobežné s osami súradnicového systému	((x1,y1),(x2,y2))
Path	16+16n bajtov	uzavretá „cesta“	((x1,y1),...)
Path	16+16n bajtov	otvorená „cesta“	[(x1,y1),...]
Polygon	40+16n bajtov	polygón	((x1,y1),...)
Circle	24 bajtov	kružnica (stred a polomer)	⟨(x,y),r⟩

3.2 Priestorové dopyty a priestorové funkcie v databázových systémoch

Pre všetky triedy priestorových dát sú okrem dátových typov špecifikované v štandarde SFA aj metódy (funkcie), ktoré možno s nimi vykonávať. Patria medzi ne napr. základné funkcie na manipuláciu s priestorovými dátami (kapitola **3.2.1**), funkcie na realizáciu priestorových analýz (kapitola **3.2.2**), funkcie na zistenie topologických vzťahov priestorových objektov (kapitola **3.2.3**) a funkcie na konverziu a vytváranie priestorových dát (kapitola **3.2.4**).

3.2.1 Základné funkcie na manipuláciu s priestorovými dátami

Základné funkcie slúžia na zistenie vlastností objektov (napr. či je objekt uzavretý) alebo ich číselných charakteristík (napr. počet bodov v objekte). Patria medzi ne funkcie uvedené v Tabuľka 3.3.

Tabuľka 3.3. Základné funkcie na manipuláciu s priestorovými dátami

Funkcia	Charakteristika
Centroid(geometry):Point[35]	vráti ťažisko *(centroid)* geometrického objektu
Area(geometry):Double	vráti výmeru geometrického objektu
Length(geometry):Double	vráti dĺžku geometrického objektu
Dimension(geometry):Integer	vráti dimenziu geometrického objektu, ktorá musí byť väčšia ako dimenzia referenčného súradnicového systému
GeometryType(geometry):String	vráti názov geometrického typu objektu
SRID(geometry):Integer	vráti SRID referenčného súradnicového systému geometrického objektu
Envelope(geometry):Geometry	vráti súradnice X, Y minimálneho opísaného obdĺžnika *(bounding box)* v tvare [(MINX, MINY), (MAXX, MINY), (MAXX, MAXY), (MINX, MAXY), (MINX, MINY)]
IsEmpty(geometry):Integer	vráti hodnotu 1 (TRUE), ak je geometrický objekt prázdny
IsClosed(geometry):Integer	vráti hodnotu 1 (TRUE) ak je línia uzavretá (t. j. začiatočný a koncový bod línie sú totožné; StartPoint ( ) = EndPoint ( ))
NumGeometries(geometry):Integer	vráti počet geometrických objektov v skupine (kolekcii)
GeometryN(geometry, i):Geometry	vráti i-ty objekt zo skupiny (kolekcie) objektov
NumPoints(geometry):Integer	vráti počet bodov v geometrickom objekte
PointN(geometry, i):Point	vráti i-ty bod geometrického objektu
ExteriorRing(geometry):LineString	vráti polygón definujúci obvod zloženého polygónu

[35] V rozšírení PostGIS databázového systému PostgreSQL je od verzie 1.2.2 preferované označenie priestorových funkcií s prefixom ST_. Pôvodné funkcie (bez prefixu) sú aj naďalej platné a s novými funkciami sú si rovnocenné (t. j. preferovaná je funkcia *ST_Centroid*, ale možno použiť aj funkciu *Centroid*).

Tabuľka 3.3. Základné funkcie na manipuláciu s priestorovými dátami (pokračovanie)

Funkcia	Charakteristika
AsText(geometry):String	vráti textovú reprezentáciu geometrického objektu vo formáte OGC WKT *(Well-Known Text)*
AsBinary(geometry):BLOB	vráti binárnu reprezentáciu geometrického objektu vo formáte OGC WKB *(Well-Known Binary)*

3.2.2 Funkcie na realizáciu priestorových analýz

Výsledkom funkcií na realizáciu priestorových analýz sú väčšinou nové priestorové objekty, ktoré vznikli na základe pôvodných objektov. Patria medzi ne funkcie uvedené v Tabuľka 3.4.

Tabuľka 3.4. Funkcie na realizáciu priestorových analýz

Funkcia	Charakteristika
Distance(geometry1, geometry2):Double	vráti najmenšiu vzdialenosť dvoch objektov
Buffer(geometry, distance):Geometry	vytvorí vzdialenostnú zónu do vzdialenosti *distance* od geometrického objektu
Intersection(geometry1, geometry2):Geometry	vytvorí prienik dvoch geometrických objektov
Union(geometry1, geometry2):Geometry	vytvorí zjednotenie dvoch geometrických objektov
Difference(geometry1, geometry2):Geometry	vytvorí priestorový rozdiel dvoch geometrických objektov
SymDifference(geometry1, geometry2):Geometry	vytvorí symetrický priestorový rozdiel dvoch geometrických objektov

3.2.3 Funkcie na zistenie topologických vzťahov priestorových objektov

Funkcie na zistenie topologických vzťahov priestorových objektov (funkcie, ktoré testujú vzájomnú polohu geometrických objektov) podporované v rozšírení jazyka SQL sú uvedené v Tabuľka 3.5.

Výsledkom napr. funkcie *Intersects* na rozdiel od funkcie *Intersection* nie je vytvorenie nového priestorového objektu (prieniku existujúcich objektov), ale zistenie, či prienik daných objektov existuje (TRUE) alebo je ním prázdna množina (FALSE).

Tabuľka 3.5. Funkcie na zistenie topologických vzťahov priestorových objektov

Funkcia	Charakteristika
Equals(geometry1, geometry2):Integer	vráti hodnotu 1 (TRUE), ak sú geometrické objekty priestorovo zhodné
Disjoint(geometry1, geometry2):Integer	vráti hodnotu 1 (TRUE), ak sa priestorové objekty nepretínajú (nemajú ani jeden spoločný bod)
Intersects(geometry1, geometry2):Integer	vráti hodnotu 1 (TRUE), ak sa geometrické objekty pretínajú (majú spoločný priestorový prienik (aspoň jeden bod))
Touches(geometry 1, geometry2):Integer	vráti hodnotu 1 (TRUE), ak sa geometrické objekty priestorovo dotýkajú
Crosses(geometry 1, geometry2):Integer	vráti hodnotu 1 (TRUE), ak sa geometrické objekty priestorovo križujú (majú spoločný priestorový prienik, ale jeden objekt nezahŕňa celý druhý objekt)
Within(geometry1, geometry2):Integer	vráti hodnotu1 (TRUE), ak sa jeden geometrický objekt nachádza celý vo vnútri druhého objektu
Contains(geometry1, geometry2):Integer	vráti hodnotu1 (TRUE), ak jeden geometrický objekt *(geometry1)* obsahuje celý druhý objekt *(geometry2)*
Overlaps(geometry1, geometry2):Integer	vráti hodnotu 1 (TRUE), ak jeden geometrický objekt *(geometry1)* prekrýva druhý objekt *(geometry2)* a ich prienik má rovnakú dimenziu, ako pôvodné geometrické objekty

Topologické vzťahy priestorových objektov sú definované a podrobne opísané napr. v (Shekhar a Xiong, 2008).

3.2.4 Funkcie na konverziu a vytváranie priestorových dát

Funkcie na konverziu a vytváranie priestorových dát v jazyku SQL sa využívajú napr. pri konverzii priestorových objektov z textovej formy do binárnej a naopak. Sú definované aj ako samostatné funkcie pre bodové, líniové a polygónové objekty. Patria medzi ne funkcie uvedené v Tabuľka 3.6.

Tabuľka 3.6. Funkcie na konverziu a vytváranie priestorových dát

Funkcia	Charakteristika
GeomFromText(text):Geometry	vytvorenie geometrického objektu z textu vo formáte OGC WKT
PointFromText(text):Point, (**LineFromText**(text):Line, **LineStringFromText**(text):LineString, **PolyFromText**(text):Polygon),	vytvorenie bodu (línie, líniového reťazca, polygónu) z textu vo formáte OGC WKT
GeomFromWKB(text):Geometry	vytvorenie geometrického objektu z textu vo formáte OGC WKB
PointFromWKB(text):Point, (**LineFromWKB**(text):Line, **LineStringFromWKB**(text):LineString, **PolyFromWKB**(text):Polygon)	vytvorenie bodu (línie, líniového reťazca, polygónu) z geometrického objektu vo formáte OGC WKB

3.3 Priestorové dopyty v jazyku SQL v databázovom systéme PostgreSQL s rozšírením PostGIS

(7. a 8. cvičenie)

Zadanie č. 3: V softvérovom prostredí PostgreSQL s rozšírením PostGIS vytvorte nasledujúce dopyty (úlohy **B72** – **B111**) v jazyku SQL s využitím implementovaných funkcií pre priestorové dáta do modelovej databázy `BA_kraj` ( ).

Úlohy

B72: Vyselektujte `id`, `nazov` a stĺpec `the_geom` s geometriou z tabuľky `Body_zaujmu` a prezrite si výsledok stĺpca `the_geom`.

B73: Vyselektujte `id`, `nazov` a stĺpec `the_geom` ako text z tabuľky `Body_zaujmu` a prezrite si výsledok stĺpca `the_geom`. Klauzula SELECT bude mať nasledovný tvar:

```
SELECT id, nazov, AsText(the_geom);
```

B74: Dopyt B73. opakujte pre tabuľky `Vodne_plochy` a `Cesty`.

B75: Vytvorte tabuľku `Body`, ktorá bude obsahovať stĺpce `id` (INT), `oznacenie` (VARCHAR (5)) a `typ` (VARCHAR (30)).

B76: Do tabuľky `Body` pridajte stĺpec `the_geom`, ktorý bude obsahovať geometriu typu POINT. Hodnoty v úvodzovkách v zátvorke sú postupne: schéma, názov tabuľky, názov stĺpca s geometriou, číslo súradnicového systému, typ geometrie, počet dimenzií.

```
SELECT AddGeometryColumn('','Body','the_geom','-1','POINT',2);
```

B77: Do tabuľky `Body` vložte nasledovné údaje: `id` = 1, `oznacenie` = MOPI, `typ` = perm. stanica, `the_geom` = 17.30655 48.334538.

```
INSERT INTO body
VALUES ('', ... ,GeomFromText('POINT(17.30655 48.334538)'));
```

B78: Spôsobom uvedeným v úlohe **B77**. vložte do tabuľky `Body` ďalšie 2 záznamy, a prezrite ich dvoma spôsobmi. Vyselektovaním geometrie ako textu a v programe QGIS.

B79: Vytvorte tabuľku `Linie`, ktorej pridáte stĺpec s geometriou typu LINESTRING.

```
SELECT AddGeometryColumn('','Linie','the_geom','-1','LINESTRING',2);
```

B80: Do tabuľky `Linie` vložte záznam ktorý bude v stĺpci `the_geom` obsahovať líniu 17.27 48.33, 17.28 48.34

```
INSERT INTO linie
VALUES ('',...,GeomFromText('LINESTRING(17.27 48.33, 17.28 48.34)'));
```

B81: V programe QGIS si prezrite tabuľku `Linie` a pomocou tohto programu do nej vložte ďalšie 3 záznamy (ľubovoľné línie).

B82: Vyselektujte všetky údaje z tabuľky `Linie` a presvedčte sa či sa v nej nachádzajú údaje vložené pomocou programu QGIS.

B83: Vytvorte tabuľku `Budovy` a pridajte jej stĺpec s geometriou typu POLYGON, ktorý bude mať definovaný súradnicový systém WGS 84 (s kódom EPSG[36]: 4326).

```
SELECT AddGeometryColumn('','Budovy','the_geom','4326','POLYGON',2);
```

B84: Do tabuľky `Budovy` vložte záznam, ktorý bude obsahovať nasledovnú geometriu: 17.25 48.31, 17.24 48.32, 17.25 48.33, 17.26 48.32, 17.25 48.31

```
INSERT INTO Budovy
VALUES ('', ... ,GeomFromText('POLYGON((17.25 48.31, 17.24 48.32,
17.25 48.33, 17.26 48.32, 17.25 48.31))',4326));
```

B85: V programe QGIS si prezrite tabuľku `Budovy` a pomocou tohto programu do nej vložte ďalšie 3 záznamy.

B86: Vyselektujte všetky údaje z tabuľky `Budovy` a presvedčte sa, či sa v nej nachádzajú údaje vložené pomocou programu QGIS.

[36] Databáza EPSG kódov obsahuje identifikátory referenčných súradnicových systémov a transformácií medzi nimi. Číselné kódy alebo názvy príslušných súradnicových systémov možno vyhľadať na stránke *www.epsg-registry.org*.

B87: V tabuľke `Body` v zázname s `id` = 1 aktualizujte geometriu na hodnotu 17.3021 48.3349.

```
UPDATE nazov_tabulky
SET the_geom=GeomFromText('POINT(17.3021 48.3349)')
WHERE .....;
```

B88: Vyselektujte všetky `obce`, ktoré svojou geometriou ležia v ploche `okresu` Senec.

```
SELECT čo
FROM z akých tabuliek
WHERE intersects(Obce.the_geom,Okresy.the_geom) AND
Okresy.nazov='...';
```

B89: Zistite vzdialenosť medzi obcami Malacky a Pernek.

```
SELECT ST_Distance(a.the_geom,b.the_geom)
FROM Obce a, Obce b
WHERE a.nazov='Malacky' AND b.nazov='Pernek';
```

B90: Vyselektujte obalové zóny so vzdialenosťou[37] 0.005 okolo geometrie tabuľky `Vodne_plochy`.

```
SELECT id, nazov, ST_Buffer(the_geom,vzdialenost)
FROM nazov_tabulky;
```

B91: Vytvorte pohľad z vyselektovaných hodnôt z úlohy **B90**.

B92: Vytvorený pohľad a tabuľku `Vodne_plochy` si prezrite v programe QGIS a skontrolujte či je obalová zóna vytvorená správne.

B93: Vyselektujte názvy 10 obcí, ktoré majú najkratšiu vzdialenosť k niektorej (ľubovoľnej) vodnej ploche.

```
...
...
ORDER BY ST_Distance( ... )
LIMIT 10;
```

[37] Jednotky, v ktorých sú uvádzané vzdialenosti, závisia od zvoleného súradnicového systému. Pre súradnicový systém WGS 84 sú vzdialenosti uvádzané v stupňoch.

B94: Vyselektujte `id`, `nazov` a dĺžku ciest z tabuľky `Cesty` kde `id` je menšie ako 15.

```
SELECT id, nazov, length(the_geom)
FROM ...
WHERE ...;
```

B95: Vyselektujte 3 najdlhšie `Cesty`, ktoré majú zadaný `nazov`.

```
SELECT id, nazov
FROM ...
WHERE nazov IS NOT NULL
ORDER BY ...
LIMIT 3;
```

B96: Vyselektujte prienik geometrií dvoch tabuliek. Celej tabuľky `Cesty` a tabuľky `Priroda` kde `nazov` objektu je „Bažantnica".

```
SELECT Cesty.id, ST_Intersection(cesty.the_geom,priroda.the_geom)
FROM Cesty,Priroda
WHERE Priroda.nazov='Bažantnica';
```

B97: Vytvorte tabuľku zo selekcie v úlohe **B96**.

```
CREATE TABLE Bazantica AS(
SELECT ...);
```

B98: Tabuľku vytvorenú v úlohe **B97** si prezrite v programe QGIS a porovnajte jej obsah s obsahom tabuliek `Cesty` a `Priroda`.

B99: Zistite výmeru vodnej plochy s názvom „Slnečné jazerá".

```
SELECT ST_Area(the_geom)
...
...;
```

B100: Vyselektujte `id` a `the_geom` záznamov z tabuľky `Cesty`, ktoré sa priestorovo dotýkajú záznamov v tabuľke `Priroda`.

```
SELECT ...
FROM ...
WHERE ST_Touches(Priroda.the_geom,Cesty.the_geom);
```

B101: Zo selekcie z úlohy **B100** vytvorte tabuľku.

```
CREATE TABLE nazov AS (
SELECT ...
FROM ...
WHERE ST_Touches(Priroda.the_geom,Cesty.the_geom));
```

B102: Tabuľku vytvorenú v úlohe **B103** si prezrite v programe QGIS a presvedčte sa, že obsahuje cesty, ktoré sa priestorovo dotýkajú záznamov z tabuľky `Priroda`.

B103: Vyselektujte `id`, `nazov` a `the_geom` záznamov z tabuľky `Cesty`, ktoré sa priestorovo nachádzajú v geometrii záznamov z tabuľky `Priroda`.

```
SELECT ...
FROM ...
WHERE ST_Within(Cesty.the_geom,Priroda.the_geom);
```

B104: Zo selekcie z úlohy **B103** vytvorte tabuľku.

```
CREATE TABLE nazov AS (
SELECT ...
FROM ...
WHERE ST_Within(Cesty.the_geom,Priroda.the_geom));
```

B105: Tabuľku vytvorenú v úlohe **B104** si prezrite v programe QGIS a presvedčte sa, že obsahuje cesty, ktoré priestorovo ležia v záznamoch tabuľky `Priroda`.

B106: Vyselektujte zjednotenú geometriu objektov s `id` 166 a 169 z tabuľky `Vodne_plochy`.

```
SELECT ST_Union(the_geom) AS the_geom
FROM ...
WHERE ... ;
```

B107: Vytvorte tabuľku zo selekcie v úlohe **B106**.

```
CREATE TABLE nazov_tab AS (
SELECT ST_Union(the_geom) AS the_geom
FROM Vodne_plochy
WHERE id = 166 OR id= 169);
```

B108: Do tabuľky vytvorenej v rámci úlohy **B107** pridajte stĺpec `id` s dátovým typom `INT`.

```
ALTER TABLE nazov_tab ADD id INT;
```

B109: Do stĺpca `id` vložte údaj 1.

```
UPDATE nazov_tab SET id=1;
```

B110: Stĺpec `id` v tabuľke zadefinujte ako primárny kľúč.

```
ALTER TABLE nazov_tab ADD PRIMARY KEY (id);
```

B111: Tabuľku si prezrite v programe QGIS a porovnajte jej obsah s obsahom tabuľky `Vodne_plochy`. Vzniknutý objekt by mal obsahovať multipolygón dvoch jazier.

4 Export, import, zálohovanie a vizualizácia priestorových dát

(9. cvičenie)

Teoretické minimum:

4.1 Export/Import dát z a do databázy

Databázové systémy (konkrétne SRDB) väčšinou ponúkajú rovnako ako iné softvéry niekoľko spôsobov importu a exportu dát (Zadanie č. 4) z a do rôznych súborov, prípadne aj rôznych formátov.

Export dát (resp. výstup dát) z databázy sa využíva najmä pri zálohovaní databázy (alebo jej časti), pri prenose databázy od jedného poskytovateľa dát k inému, pri prenose databázy do iného softvérového prostredia alebo z jedného média na iné.

Import dát (resp. vstup dát) do databázy sa využíva pri obnove existujúcej databázy z jej zálohy alebo pri načítavaní databázy z iného pôvodného zdroja (tzv. databázy tretej strany).

Pri spracovaní a analýze priestorových dát je často potrebné exportovať priestorové dáta z databázového systému v požadovanom formáte príslušného GIS, v ktorom budú dáta analyzované, alebo naopak, priestorové dáta z GIS importovať do databázového systému, kde budú potom spoločne spravované s dátami z iných zdrojov. V súčasnosti sa v GIS na správu a ukladanie priestorových dát využíva napríklad formát *Shapefile*[38], preto je užitočné poznať rôzne spôsoby výmeny dát medzi softvérovými prostrediami databázových systémov a GIS.

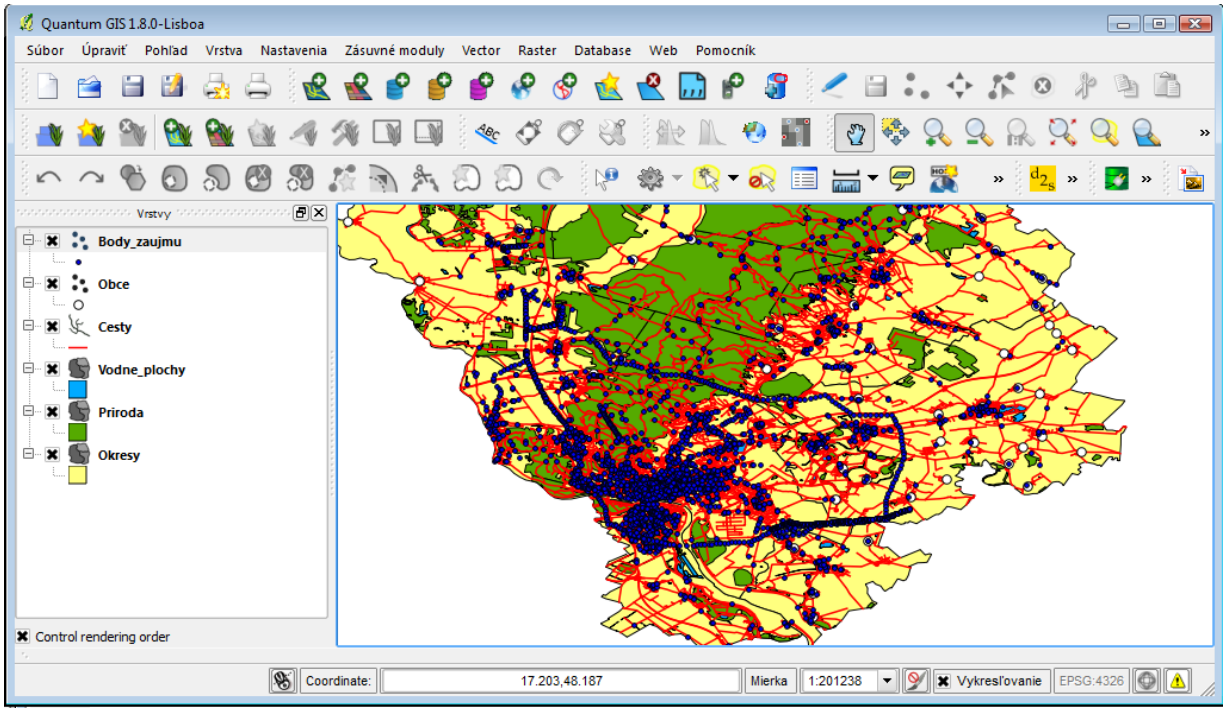
4.2 Vizualizácia dát

Vizualizácia dát umožňuje prezentovať alebo aj analyzovať objekty, javy a vzájomné vzťahy prostredníctvom ich grafického zobrazenia. V prípade priestorových dát v databázach má vizualizácia špeciálne postavenie, pretože priestorové dáta poskytované v textovom alebo

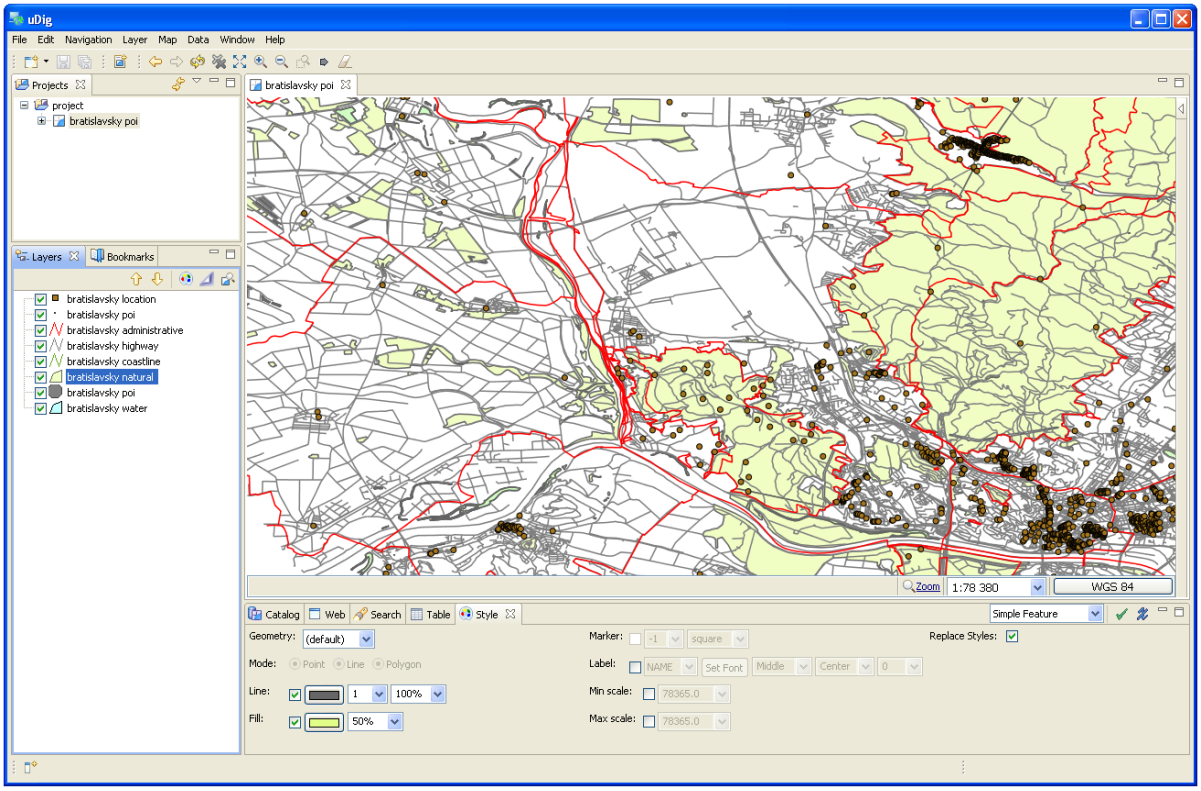
[38] **Shapefile** je pôvodne proprietárny formát spoločnosti ESRI (*ESRI Shapefile*), ktorý sa v súčasnosti využíva aj v iných softvérových prostrediach GIS na správu a ukladanie vektorových priestorových dát.

binárnom tvare nepredstavujú pre používateľa takú zrozumiteľnú formu reprezentácie, ako informácie zobrazené napr. na mape alebo v digitálnej forme na monitore počítača. Niektoré informácie o priestorových objektoch, akými sú napr. tvar, veľkosť alebo topologické vzťahy objektov, sú síce v databáze implicitne zahrnuté, ale pomocou grafickej vizualizácie ich možno jednoduchou formou prezentovať používateľovi. Topologické vzťahy, v rámci ktorých je napr. budova umiestnená vo vnútri parcely alebo líniový objekt reprezentujúci rieku nepretína objekt budovy, vyplývajú potom pre používateľa priamo z grafického zobrazenia predmetnej situácie.

Výhodou vizualizácie priestorových dát je aj možnosť voľby rôznych mapových znakov (ich vlastností) a mapových kompozícií na zvýraznenie alebo odlíšenie významných objektov, javov a ich vlastností. Na vizualizáciu priestorových dát z open source databázového systému PostgreSQL s rozšírením PostGIS je vhodné využiť open source softvéry, napr. QGIS (http://www.qgis.org/en/site/) (Obr. 4.1) alebo uDig (http://udig.refractions.net/) (Obr. 4.2). Podobným spôsobom možno ale využiť aj niektorý z proprietárnych softvérov GIS (napr. softvérové prostredia ArcGIS alebo GeoMedia).



Obr. 4.1. Vizualizácia priestorových dát v prostredí QGIS



Obr. 4.2. Vizualizácia priestorových dát v prostredí uDig

QGIS (QuantumGIS) je používateľsky prívetivý open source GIS softvér licencovaný pod licenciou GNU GPL ((http://www.gnu.org/copyleft/gpl.html). Podporuje množstvo vektorových, rastrových a databázových formátov a tiež rôznych funkcií na spracovanie, analýzu a prezentáciu priestorových dát. QGIS možno využívať v operačných systémoch Linux, Unix, Mac OS X, Windows a Android.

Softvér uDig je tiež open source desktopová GIS aplikácia, ktorá je zameraná na zobrazovanie a editovanie priestorových dát z databáz alebo z webových zdrojov. Softvér uDig je poskytovaný pod licenciami EPL *(Eclipse Public Licence)* (https://www.eclipse.org/legal/epl-v10.html) a BSD *(Berkeley Software Distribution)* (http://opensource.org/licenses/BSD-3-Clause). uDig je spustiteľný ako hrubý klient *(thick klient)*[39] v operačných systémoch Linux, Mac OS X a Windows.

[39] Hrubý klient je v niektorých zdrojoch označovaný aj ako tučný klient alebo silný klient. Predstavuje riešenie, v rámci ktorého je na strane klienta celá prezentačná a aplikačná vrstva, ale aj časť dátovej vrstvy v zmysle označenia architektúry klient/server. V zmysle GIS to znamená, že celá funkcionalita GIS softvéru je umiestnená na klientskom počítači a môže byť na ňom umiestnená aj časť dát, s ktorým softvérová aplikácia pracuje.

4.3 Export/import a zálohovanie databázy v PostgreSQL a vizualizácia priestorových dát v softvérovom prostredí QGIS

(9. cvičenie)

Zadanie č. 4:

a) Importujte do databázy v PostgreSQL pripravené súbory priestorových dát[40] vo formáte *Shapefile*. Výsledok importu overte prostredníctvom vizualizácie priestorových dát v softvérovom prostredí QGIS.

b) Exportujte vybrané súbory z PostgreSQL vo formáte *Shapefile*.

c) Z pripravených priestorových dát vytvorte v softvérovom prostredí QGIS a uDig aspoň dva rôzne mapové výstupy.

d) Vytvorte zálohu celej databázy v PostgreSQL.

Postup riešenia:

1. Do pracovného adresára si pripravte všetky súbory vo formáte *Shapefile*, ktoré budete importovať do databázy.
2. Zo súborov vo formáte *Shapefile* vytvorte SQL skripty pomocou nasledujúceho príkazu, ktorý zadáte do príkazového riadku (otvorenie príkazového riadku: Start → Run → „cmd“ → Enter). (Podčiarknuté údaje je potrebné zmeniť podľa konkrétnej situácie. V príkaze sa postupne nachádzajú: názov programu, ktorý preloženie vykoná, cesta k prekladanému súboru vo formáte *Shapefile*, schéma a názov tabuľky v databáze a cesta k vytvorenému SQL skriptu.)

```
"C:\Program Files\pgAdmin III\1.10\shp2pgsql" D:\USER\...\shapefile
public.nazov_tabulky > D:\USER\...\nazov.sql
```

Pre jeden súbor vytvorte skript, v ktorom bude geometria uložená vo forme textu:

[40] Súbory sú v prílohe č. 4.

```
"C:\Program Files\pgAdmin III\1.10\shp2pgsql" -w
D:\USER\...\shapefile public.nazov_tabulky > D:\USER\...\nazov.sql
```

Skripty vytvorte postupne pre všetky súbory vo formáte *Shapefile*.

3. Vytvorené SQL skripty si prezrite v textovom editore.
4. Vytvorené skripty aplikujte do databázy s vašimi parametrami pripojenia. Štruktúra príkazu bude nasledujúca:

```
"C:\Program Files\pgAdmin III\1.10\psql" -f D:\USER\.. ..\nazov.sql
-U user -d database -h host -p port
```

V uvedenom príkaze po názve programu nasleduje umiestnenie skriptu, používateľ, databáza, hostiteľ (URL *(Uniform Resource Locator)* alebo IP *(Internet Protocol)* adresa) a port. Postupne aplikujte všetky skripty.

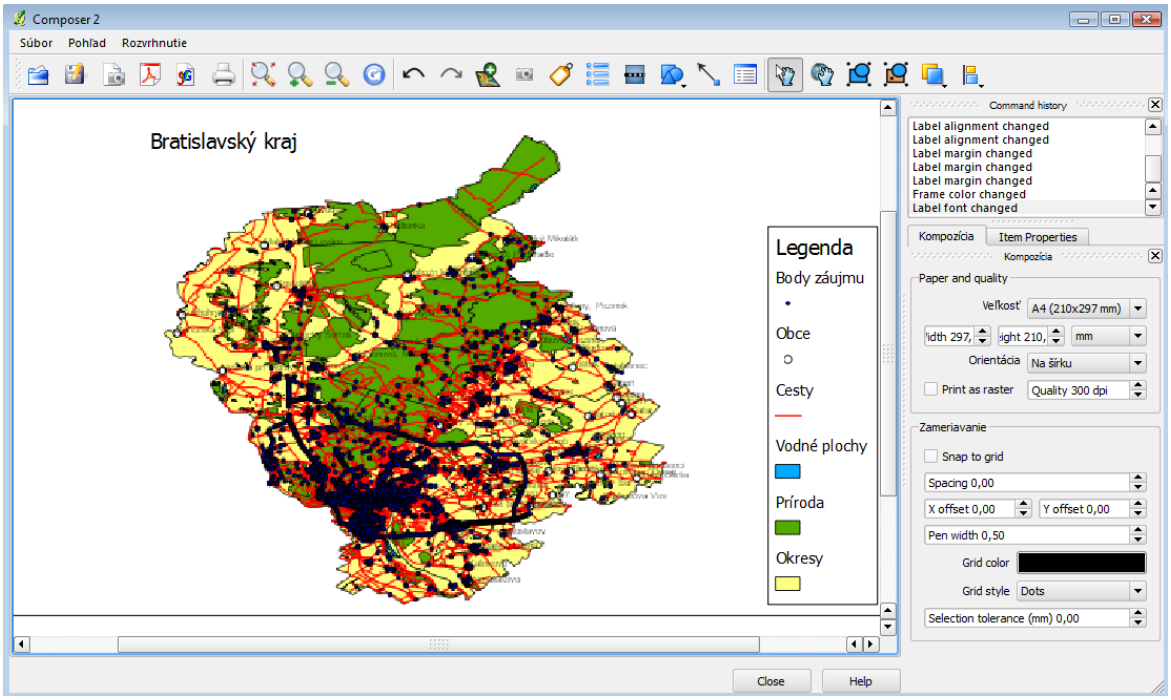
5. V programe QGIS sa presvedčte, že súbory vo formáte *Shapefile* boli importované do databázy.
6. Ako druhú možnosť importovania súborov do databázy si vyskúšajte nástroj SPIT *(Shapefile to PostGIS Import Tool)* programu QGIS, ktorý nájdete v položke „Zásuvné moduly“. Pri importovaní zmeňte „Názov relácie v databáze“ tak, aby sa začínal textovým reťazcom „spit_“.
7. Exportujte tabuľky (tie, ktorých názov sa nezačína textovým reťazcom „spit_“) z používanej databázy do formátu *Shapefile*. Za parametrom –f sa nachádza cesta k exportovanému súboru.

```
"C:\Program Files\pgAdmin III\1.10\pgsql2shp" -f D:\USER\.. ..\nazov
-h host -p port -u user -P password database public.nazov_tabulky
```

8. Ako jednoduchšiu variantu exportovania si vyskúšajte možnosť načítania vrstvy do programu QGIS, potom kliknite na vrstvu pravým tlačidlom myši a zvoľte „Uložiť ako *Shapefile*...“ (alebo označte časť vrstvy a potom zvoľte možnosť „Uložiť výber ako *Shapefile* ...“).

9. Načítajte vrstvy z používanej databázy do programu QGIS. Definujte súradnicový systém s kódom (EPSG:102067) a povoľte priamy prevod medzi súradnicovými systémami (Nastavenia – Vlastnosti projektu). Zoraďte vrstvy (vertikálne) do vhodného poradia, objektom vo vrstvách definujte mapové znaky a ich vlastnosti (farba, hrúbka, typ čiary, transparentnosť a pod.). Zvoleným vrstvám definujte popis s jeho parametrami.

10. Z takto a podobne nastavených vrstiev vytvorte aspoň dve rozdielne mapy vo formátoch PNG, JPEG, TIFF alebo PDF. Použite na to nástroj „Skladateľ tlačových výstupov“ *(Composer Manager)* programu QGIS. Pre mapy zostavte ich kompozície (nadpis, legendu, mierku, prípadne doplňte aj iné dôležité informácie) (Obr. 4.3).



Obr. 4.3. Tvorba mapového výstupu v prostredí QGIS

11. Do programu uDig si načítajte tabuľky z pracovnej databázy (Layer – Add – PostGIS). Vhodne zvoľte štýlovanie a kompozíciu vrstiev, pridajte legendu (Layer – Legend), mierku (Layer – Scalebar) a exportujte mapu ako rastrový súbor (obrázok) (File – Export – Map to Image).

12. Zálohujte si pracovnú databázu.

```
"C:\Program Files\pgAdmin III\1.10\pg_dump.exe" --host host --port
port --username user --format custom --blobs --verbose --file
"C:\USERS\.. ..\nieco.backup" database
```

13. To isté vykonajte prostredníctvom programu pgAdmin III, kde po kliknutí pravým tlačidlom myši na názov databázy, ktorú chcete zálohovať, zvoľte „Backup...“ Zadajte názov súboru, jeho umiestnenie a potvrďte pomocou voľby „OK“.
14. Obdobnými spôsobmi možno databázu obnoviť. Slúži na to nasledujúci príkaz. (Na obnovu databázy zo zálohy je potrebné mať používateľské konto s dostatočnými právami zápisu.)

```
"C:\Program Files\pgAdmin III\1.10\pg_restore.exe" --host
147.175.19.15 --port 5432 --username user --dbname database --list
"C:\USER\.. ..\nieco.backup"
```

5 Návrh a tvorba databázy

(10. – 12. cvičenie)

Teoretické minimum:

5.1 Entitno-relačné diagramy – modelovanie entít a relácií

Základ konceptuálneho modelovania relačných databáz je vyjadrenie modelov databáz pomocou grafických symbolov. Entitno-relačné diagramy (E-R diagramy) sa využívajú na modelovanie entít *(**E**ntities)* a vzťahov *(**R**elationships)* pomocou príslušných grafických symbolov. K uvedeným dvom základným konštrukciám sa často pridáva tretia – atribút *(**A**ttribute)*. V tom prípade sa používa aj pomenovanie **ERA diagram**.

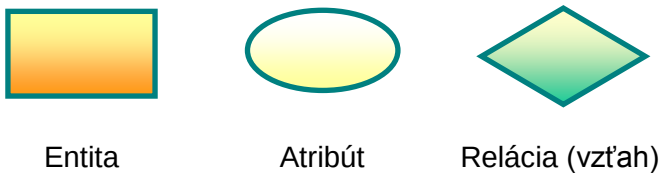
V E-R diagramoch entity vyjadrujú množiny entít (ich názvy sa uvádzajú v pluráli), atribúty vyjadrujú vlastnosti entít a relácie ich vzájomné vzťahy (spojenia). E-R diagramy opisujú entity, ktoré v modeli vystupujú a tiež ako spolu súvisia, avšak neobsahujú žiadne operácie, t. j. neopisujú, ako sa veci menia (napr. ako vznikajú a zanikajú).

Nevýhoda E-R diagramov je, že na ich vytváranie neexistuje jednotná (štandardná) notácia. Jedným z najznámejších a v súčasnosti ešte stále používaným formátom E-R diagramov je Chenov formát, ktorý zaviedol Peter Chen v roku 1976 so snahou o zjednotenie dovtedy používaných formátov (kapitola **5.1.1**). Jeho nevýhody eliminuje tzv. relačný formát diagramov, avšak ani jeho notácia nie je jednotná (kapitola **5.1.2**).

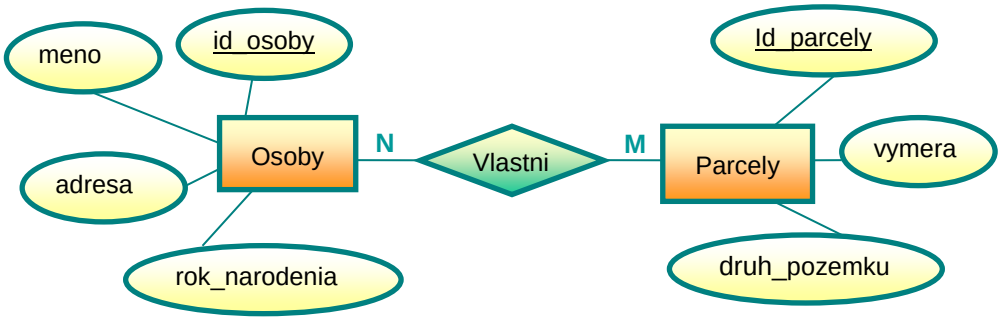
5.1.1 Chenov formát E-R diagramov

V Chenovom formáte (Chenovej notácii) E-R diagramov sa označujú entity (alebo entitné množiny) symbolom obdĺžnik, na označenie ich atribútov sa používa symbol elipsa a na znázornenie vzájomných vzťahov symbol kosoštvorec (Obr. 5.1 a Obr. 5.2). Podčiarknuté názvy atribútov v E-R diagrame zodpovedajú primárnym kľúčom.

Pri znázorňovaní relácií medzi entitami možno zobraziť aj počet výskytov jednej entity voči druhej v rámci ich vzájomného vzťahu, t. j. násobnosť (kardinalitu relácie *(multiplicity)*). Napríklad na diagrame na Obr. 5.2 je znázornený vzťah, v rámci ktorého osoba môže vlastniť M parciel, ale aj parcela môže mať N vlastníkov (vzťah typu M:N).



Obr. 5.1. E-R diagramy – grafické symboly (Chenov formát)



Obr. 5.2. E-R diagram

Násobnosť vzťahov najčastejšie vyjadruje jednu z nasledujúcich situácií (Obr. 5.3):

- MANY - MANY (M - M); napr. osoba môže vlastniť viac parciel, ale aj parcela môže mať viac vlastníkov, vzťah M:N,
- MANY - ONE (M - O); napr. katastrálne územie obsahuje viac parciel, ale parcela patrí len do jedného katastrálneho územia, vzťah 1:N,
- ONE - ONE (O - O); napr. fyzická osoba má práve jedno rodné číslo a zároveň rodné číslo patrí práve jednej fyzickej osobe, vzťah 1:1.

Obr. 5.3. Násobnosť relácií

Ďalšie možnosti znázornenia násobnosti relácií znázornené na Obr. 5.3 odpovedajú situáciám, v ktorých relácia môže, ale nemusí mať zastúpenie vo vzťahu (napr. osoba vlastní nula alebo viac budov). Ide o tzv. povinné alebo nepovinné členstvo vo vzťahu.

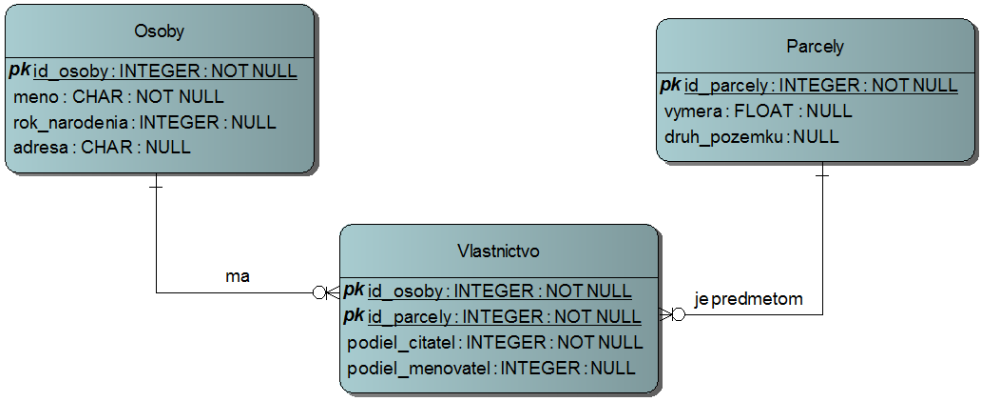
5.1.2 Relačný formát diagramov

Chenov formát diagramov je pri zložitých modeloch nevyhovujúci, pretože jeho grafické symboly sú priestorovo veľmi rozsiahle. Množstvo atribútov potom robí model nečitateľným a z pohľadu priestoru neefektívnym. Z uvedeného dôvodu bol neskôr vyvinutý tzv. **relačný formát diagramov**. Upravené E-R diagramy (relačný formát) využívajú zápis atribútov priamo do symbolov entít a tým zároveň umožňujú presnejšie špecifikovať ich atribúty napr. označením primárnych a cudzích kľúčov a definovaním ich dátového typu alebo domény. Nevýhodou relačného formátu znázornenia je, že syntax symbolov sa v jednotlivých spôsoboch vyjadrenia diagramov líši, čo platí aj o softvéroch na ich vytváranie. Príklad relačného formátu diagramu vytvoreného v rôznych softvérových prostrediach je uvedený na Obr. 5.4.

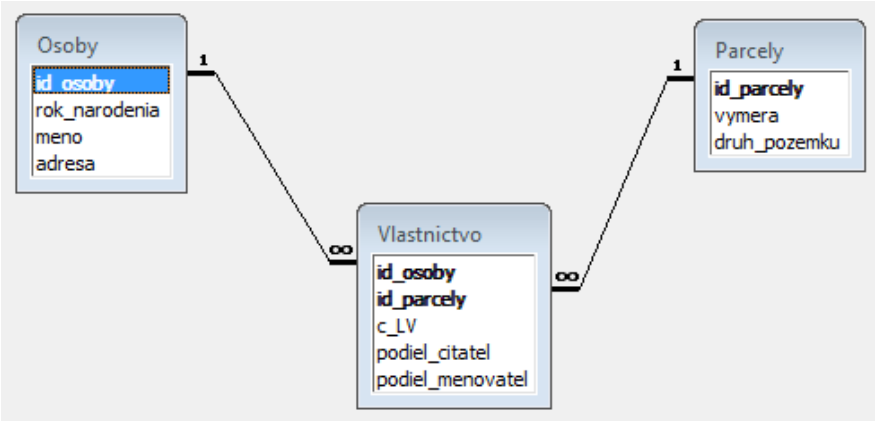
Všeobecné vlastnosti E-R diagramov v relačnom formáte možno charakterizovať nasledovne:

- **entity** (entitné množiny) reprezentujú obdĺžniky a ich atribúty sú textovo opísané vo vnútri týchto obdĺžnikov,
- **násobnosť (kardinalita) vzťahov** môže byť vyjadrená symbolmi prevzatými z E-R diagramov (Obr. 5.4a), ale napr. aj symbolmi 1 a ∞ (Obr. 5.4b) alebo šípkami, ktoré smerujú od entitnej množiny s hodnotou MANY *(viac)* k entitnej množine s hodnotou ONE *(jeden)* (Obr. 5.4c).
- **primárny kľúč** je označený skratkou „PK“ *(Primary Key)* pri názve atribútu, prípadne tučným alebo podčiarknutým písmom,
- **cudzí kľúč** je označený skratkou „FK“ *(Foreign Key)* pri názve atribútu.

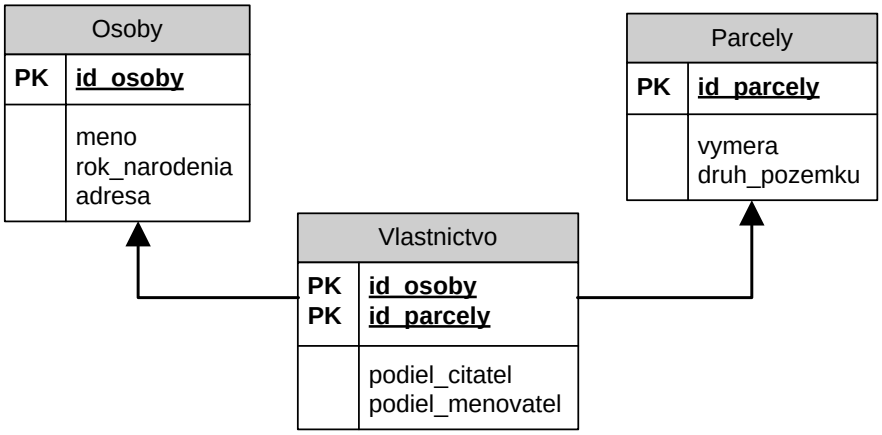
a)



b)



c)



Obr. 5.4. Relačný formát diagramov v softvérovom prostredí: a) *Select Architect (Entity Relationships Diagram)*, b) *Microsoft Access*, c) *Microsoft Visio*

5.1.3 Lineárny textový zápis

Skrátenú formu zápisu konceptuálneho modelu databázy predstavuje tzv. **lineárny textový zápis**, v ktorom sa zapíše len názov entity a v okrúhlych zátvorkách jej atribúty. Primárny kľúč sa vyznačuje tučným alebo podčiarknutým písmom. Príkladom lineárneho textového zápisu konceptuálneho modelu databázy obsahujúcej dve entitné množiny (`Osoby` a `Parcely`) a vzťah medzi nimi je:

1. entity:

   ```
   E: Osoby (id_osoby, meno, rok narodenia)
   E: Parcely (id_parcely, vymera, druh_pozemku)
   ```

2. vzťah:

   ```
   R: vlastni (Osoby, Parcely)
   ```

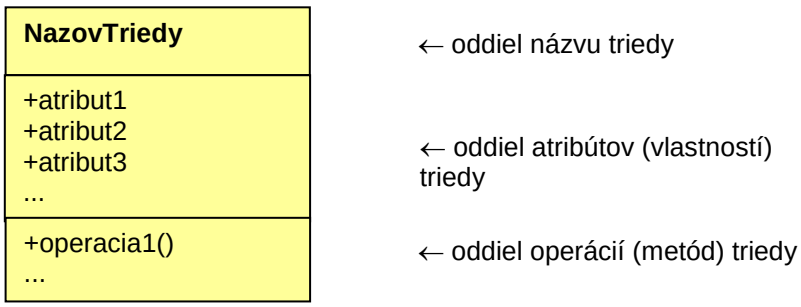
5.1.4 Diagram tried v jazyku UML

UML *(Unified Modeling Language)* je modelovací jazyk, ktorý sa stal štandardom na vizuálne modelovanie systémov. Pri návrhu databáz sa využíva z jazyka UML predovšetkým **diagram tried** *(Class diagram)*, ktorý opisuje štruktúru systému (alebo aj databázy) zobrazením tried a vzťahov medzi nimi.

Trieda *(class)* je deskriptorom objektov, t. j. združuje objekty s rovnakým typom vlastností a funkcií a zároveň tvorí šablónu na ich vytváranie. Vyjadrenie, že objekt je inštanciou triedy, znamená, že bol vytvorený na základe opisu (atribútov a metód), ktorý trieda obsahuje. Názvy tried sa v jazyku UML píšu s veľkým začiatočným písmenom, názvy objektov s malým začiatočným písmenom. Napr. inštanciou triedy Parcela môže byť konkrétny objekt „parcela číslo 123“ s konkrétnymi hodnotami atribútov.

V jazyku UML je trieda zobrazená pomocou obdĺžnikového symbolu, zloženého z troch častí (oddiel názvu triedy, oddiel atribútov a oddiel operácií) (Obr. 5.5). Povinnou zložkou symbolu triedy je len oddiel jej názvu. Okrem názvu môže obsahovať svoje vlastnosti (atribúty) v oddieli atribútov. Každý atribút je označený svojim menom, pri ktorom môžu byť uvedené aj jeho parametre (napr. dátový typ alebo preddefinovaná hodnota) a typ návratovej hodnoty. V tretej časti symbolu triedy môžu byť zobrazené metódy, v jazyku UML nazývané

v štádiu modelovania aj operácie. Triedou možno v modelovaní databáz nahradiť entitnú množinu. Keďže jazyk UML je určený na objektovo orientované modelovanie celých systémov (na rozdiel od E-R diagramov, ktoré sú určené konkrétne na modelovanie relačných databáz), rozdiel medzi triedou a entitnou množinou je napr. práve možnosť definovania metód pre triedy.



Obr. 5.5. Notácia jazyka UML – trieda v diagrame tried

Vzájomné väzby (vzťahy) medzi triedami sa v jazyku UML nazývajú **relácie**[41]. Medzi základné relácie využívané na modelovanie v diagrame tried patria vzťahy asociácia, závislosť, dedičnosť, agregácia a kompozícia. Grafické symboly na ich vyjadrenie v jazyku UML sú uvedené na Obr. 5.6a.

Asociácia vyjadruje všeobecný vzťah medzi triedami.

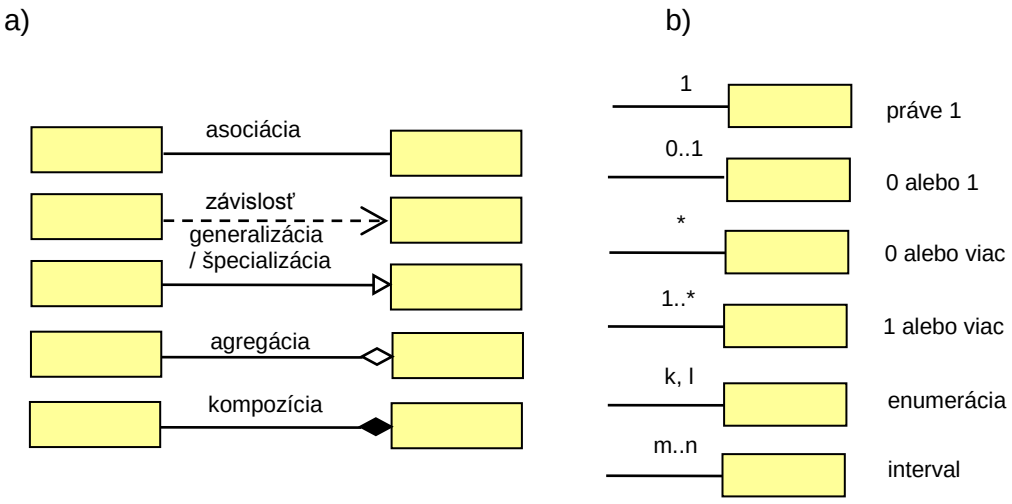
Závislosť je typ relácie, v ktorej sa zmena jedného elementu automaticky premietne do druhého elementu, ktorý je na ňom závislý.

Generalizácia (resp. špecializácia) je vzťah typu nadtrieda a podtrieda, ktorý úzko súvisí s vytváraním hierarchie tried. Hierarchiou sa nepriamo vytvorí dedičná väzba (**dedičnosť**) medzi nadradenými a podradenými triedami (nadtriedami a podtriedami). Vo vzťahu dedičnosti potomkovia (podtriedy) dedia všetky charakteristické vlastnosti svojho predka (nadtriedy), t. j. všetky atribúty má zároveň aj každá jej podtrieda, pričom podtriedy môžu obsahovať aj niektoré ďalšie atribúty, ktorými sa navzájom odlišujú. Predmetom dedenia sú všetky atribúty, operácie, relácie a obmedzenia nadtriedy.

[41] Pojmy *relation* (relácia) a *relationship* (vzťah) sa často prekladajú do slovenského jazyka rovnako – ako *relácia*. Preto sa aj v jazyku UML používa na označenie vzťahov pojem *relácia*. V kontexte problematiky relačných databáz je ale vhodné pojmy relácia (*relation*) a vzťah (*relationship*) rozlišovať a pojem *relácia* chápať aj vo všeobecnejšom význame (kapitola **2.1**).

Agregácia a **kompozícia** sú vzťahy typu celok a jeho súčasti. Agregácia vyjadruje voľnú väzbu (t. j. celok a jeho oddeliteľné súčasti), kompozícia je jej špeciálnym prípadom a predstavuje veľmi pevnú väzbu (celok a jeho neoddeliteľné súčasti).

Relácie, podobne ako v entitno-relačných diagramoch, môžu obsahovať aj vyjadrenie násobnosti príslušného vzťahu. Notácia násobnosti vzťahov v diagrame tried v jazyku UML je znázornená na Obr. 5.6b).



Obr. 5.6. a) Vzťahy (relácie) medzi triedami v diagrame tried, b) násobnosť vzťahov v diagrame tried v jazyku UML

Podrobne sa modelovaniu v jazyku UML a charakteristike jednotlivých diagramov venujú publikácie (Page-Jones, 2001), (Kanisová a Műller, 2006) a (Arlow a Neustadt, 2003).

5.2 Pravidlá a konvencie návrhu relačných databáz

Základné a najdôležitejšie pravidlo pri návrhu databázy je snaha o elimináciu redundancie dát. Nežiaduca redundancia dát znamená plytvanie pamäte, nadmerný obsah prázdnych hodnôt (hodnôt NULL), ale hlavne spôsobuje nekonzistenciu databázy, čo je v databázových systémoch zásadný problém. Medzi ďalšie dôležité pravidlá návrhu databáz patria nasledujúce (Plachetka, 2012):

- nepoužívať entitné množiny tam, kde sa dajú nahradiť atribútom,
- nesnažiť sa zviazať každú entitu s každou inou, väzba má zmysel len vtedy, keď celý primárny kľúč prvej entity tvorí cudzí kľúč v druhej,

- názvy primárnych a cudzích kľúčov sa odporúča voliť tak, aby cudzí kľúč mal rovnaký názov ako príslušný primárny kľúč,
- názvy relácií majú byť výstižné, nie všeobecné, ako napr.: „je“, „má“, „patrí do“, „je spojený s“ a pod.,
- vytvoriť si vhodnú konvenciu pomenovania entít a atribútov a dodržiavať ju počas celého procesu návrhu databázy (pomôže to udržať konzistentnosť modelu).

Pravidlá na pomenovanie tabuliek a atribútov sú rôzne a líšia sa v závislosti od použitého databázového systému. Zavedenie a dodržiavanie jednotného systému označovania a pomenovania je dôležité najmä pri rozsiahlych databázach s väčším množstvom tabuliek. Pomenovanie by malo byť vecné, jednoduché a prehľadné, aby z názvu bolo možné čiastočne odvodiť obsah tabuľky a jej účel. Názvy tabuliek a atribútov majú byť (Dobešová, 2004):

- výstižné,
- čo najkratšie (najlepšie do 25 znakov),
- bez medzier (namiesto medzier sa používajú podčiarkovníky),
- bez diakritiky (aj keď to systém umožňuje) (napr. vhodné názvy pre dátum narodenia sú: datum_narodenia, DatNar, Dat_nar a pod.),
- bez zakázaných znakov (!,., [,]) a nesmú sa začínať medzerou alebo znakmi % a ?,
- názvy objektov (tabuliek) musia byť v rámci databázy jedinečné a zároveň nesmú byť kľúčovými slovami jazyka SQL alebo inými rezervovanými slovami používaného databázového prostredia.

Postup tvorby konceptuálneho modelu relačnej databázy prostredníctvom E-R diagramov môže byť potom napr. nasledujúci:

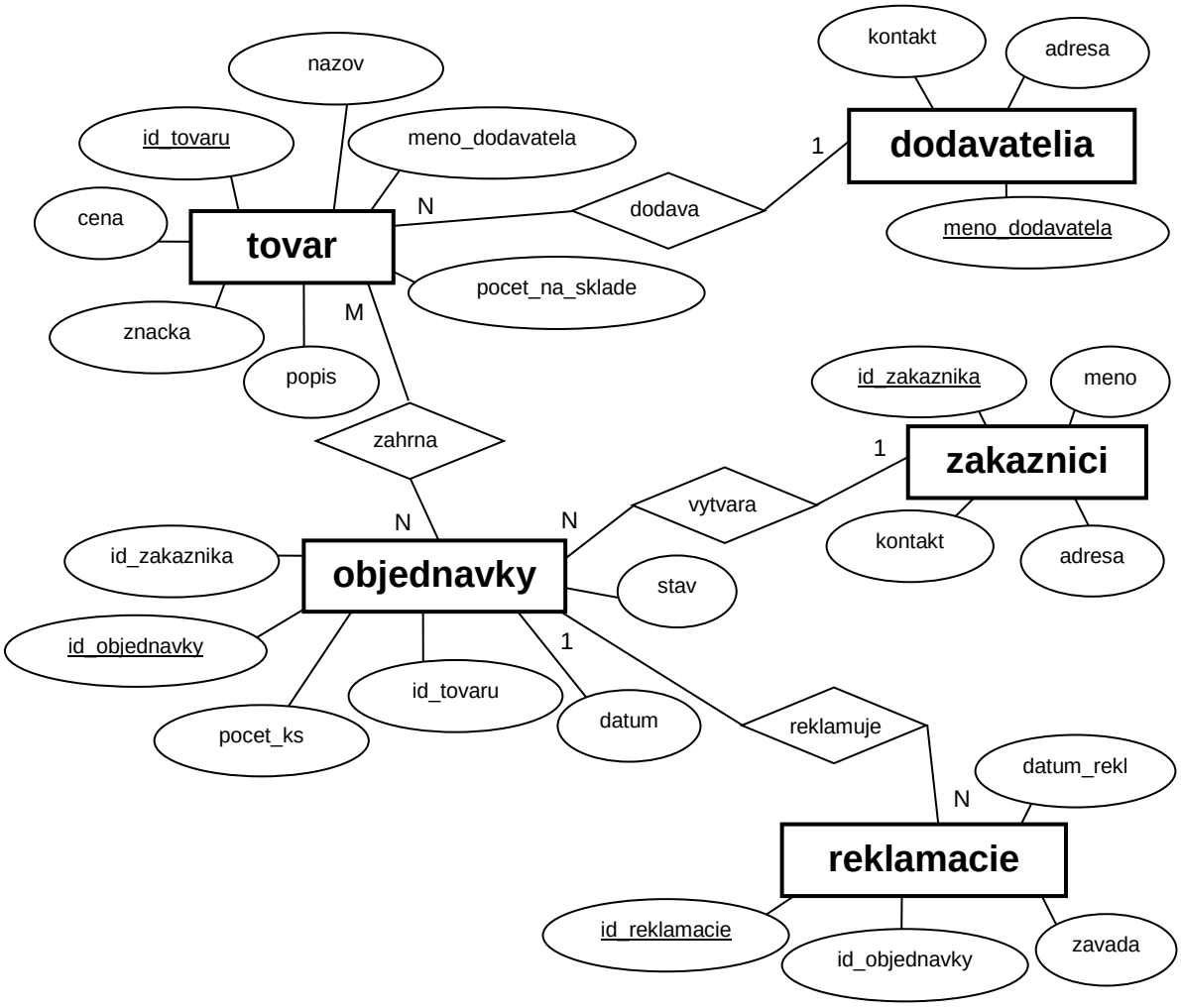
1. identifikácia entitných množín,
2. definovanie kľúčov,
3. identifikácia vzťahov, do ktorých vstupujú príslušné entitné množiny,
4. stanovenie atribútov,
5. formulácia integritných obmedzení[42].

[42] **Integritné obmedzenia** sú pravidlá, ktoré zavádzajú do databázy logické obmedzenia typov a hodnôt atribútov, entít a vzťahov tak, aby databáza čo najpresnejšie zodpovedala zobrazovanej realite. K schéme databázy sa napr. pridávajú pravidlá, ktoré obmedzujú široké možnosti vstupu záznamov do relácií (napr. že hodnoty atribútu `rok` môžu byť len z intervalu ⟨2000, 2014⟩, kód druhu pozemku môže byť len z príslušnej domény alebo výmera parcely musí byť kladné číslo).

5.3 Tvorba štruktúry tabuliek databázy v PostgreSQL

(10. cvičenie)

Zadanie č. 5.1: Vytvorte štruktúru tabuliek v prostredí PostgreSQL podľa E-R diagramu databázy „Predajňa výpočtovej techniky“ (**`Predajna_VT`**) (Obr. 5.7).



Obr. 5.7. E-R diagram databázy `Predajna_VT`

5.4 Návrh projektu v databázovom systéme PostgreSQL

(11. a 12. cvičenie)

Zadanie č. 5.2:

a) Navrhnite a potom v prostredí PostgreSQL vytvorte vlastný projekt databázy. Projekt bude obsahovať:

- E-R diagram (alebo diagram tried v jazyku UML) pre navrhnutú databázovú štruktúru,
- technickú správa s opisom projektu, použitých entít, atribútov a ich dátových typov,
- súbor so zálohou celej databázy (súbor *.backup), najmä v tom prípade, ak projekt nebude realizovaný na určenom serveri.

Databáza bude obsahovať:

- minimálne päť tabuliek, ktoré budú navzájom previazané kľúčmi,
- minimálne dve tabuľky, ktoré obsahujú aj priestorové dáta (t. j. také, ktoré zahŕňajú aspoň geometriu objektov reprezentovanú priestorovými dátovými typmi ako napr. bod, línia alebo polygón),
- minimálne jednu doménu hodnôt,
- minimálne päť záznamov v každej tabuľke, pričom v týchto vzorových záznamoch budú vyplnené všetky hodnoty atribútov,
- minimálne dva pohľady, ktoré vzniknú kombináciou vytvorených tabuliek.

b) Vytvorte minimálne päť dopytov do databázy, prostredníctvom ktorých budú realizované výbery vždy aspoň z dvoch tabuliek.

6 Výsledky

Dopyty zo zadania č. 2.1

A1.

```
SELECT meno
FROM Osoby
WHERE rok_narodenia <= 1985;
```

	meno character(30)
1	Novák Peter
2	Veselá Mária
3	Nový Pavol
4	Mrkvička Ján

A2.

```
(SELECT id_parcely
FROM Vlastnictvo
WHERE id_osoby=105)
UNION
(SELECT id_parcely
FROM Vlastnictvo
WHERE id_osoby=107);
```

	id_parcely integer
1	11
2	33

A3.

```
(SELECT id_parcely
FROM Vlastnictvo
WHERE id_osoby=105)
INTERSECT
(SELECT id_parcely
FROM Vlastnictvo
WHERE id_osoby=107);
```

	id_parcely integer
1	11

A4.

```
(SELECT id_parcely
FROM Vlastnictvo
WHERE id_osoby=105)
EXCEPT
(SELECT id_parcely
FROM Vlastnictvo
WHERE id_osoby=107);
```

	id_parcely integer
1	33

A5.

a)

```
SELECT Vlastnictvo.*, Osoby.*
FROM Vlastnictvo, Osoby
WHERE Vlastnictvo.id_osoby=Osoby.id_osoby;
```

	id_osoby integer	id_parcely integer	podiel_c integer	podiel_m integer	id_osoby integer	meno character(30)	rok_narodenia integer
1	106	22	1	1	106	Novák Peter	1984
2	107	33	2	5	107	Veselá Mária	1985
3	108	33	3	5	108	Nový Pavol	1971
4	105	11	1	4	105	Mrkvička Ján	1965
5	107	11	3	4	107	Veselá Mária	1985

alebo:

```
SELECT V.*, O.*
FROM Vlastnictvo V, Osoby O
WHERE V.id_osoby=O.id_osoby;
```

b)

```
SELECT Vlastnictvo.id_parcely, Osoby.*
FROM Vlastnictvo, Osoby
WHERE Vlastnictvo.id_osoby=Osoby.id_osoby
ORDER BY Vlastnictvo.id_parcely ASC, Osoby.rok_narodenia DESC;
```

	id_parcely integer	id_osoby integer	meno character(30)	rok_narodenia integer
1	11	107	Veselá Mária	1985
2	11	105	Mrkvička Ján	1965
3	22	106	Novák Peter	1984
4	33	107	Veselá Mária	1985
5	33	108	Nový Pavol	1971

A6.

```
SELECT *
FROM Vlastnictvo, Parcely;
```

	id_osoby integer	id_parcely integer	podiel_c integer	podiel_m integer	id_parcely integer	vymera double precision	druh_pozemku character varying(20)
1	106	22	1	1	11	10422	orná pôda
2	106	22	1	1	22	344	zastavaná plocha
3	106	22	1	1	33	987	záhrada
4	107	33	2	5	11	10422	orná pôda
5	107	33	2	5	22	344	zastavaná plocha
6	107	33	2	5	33	987	záhrada
7	108	33	3	5	11	10422	orná pôda
8	108	33	3	5	22	344	zastavaná plocha
9	108	33	3	5	33	987	záhrada
10	105	11	1	4	11	10422	orná pôda
11	105	11	1	4	22	344	zastavaná plocha
12	105	11	1	4	33	987	záhrada
13	107	11	3	4	11	10422	orná pôda
14	107	11	3	4	22	344	zastavaná plocha
15	107	11	3	4	33	987	záhrada

A7.

a)

```
SELECT meno, rok_narodenia
FROM Osoby;
```

	meno character(30)	rok_narodenia integer
1	Kováč Filip	1991
2	Novák Peter	1984
3	Veselá Mária	1985
4	Nový Pavol	1971
5	Mrkvička Ján	1965

b)

```
SELECT DISTINCT meno, rok_narodenia
FROM Osoby;
```

c)

```
SELECT DISTINCT meno, rok_narodenia
FROM Osoby
ORDER BY meno DESC;
```

	meno character(30)	rok_narodenia integer
1	Veselá Mária	1985
2	Nový Pavol	1971
3	Novák Peter	1984
4	Mrkvička Ján	1965
5	Kováč Filip	1991

A8.

```
SELECT *
FROM Osoby
WHERE rok_narodenia<1985;
```

	id_osoby integer	meno character(30)	rok_narodenia integer
1	106	Novák Peter	1984
2	108	Nový Pavol	1971
3	105	Mrkvička Ján	1965

A9.

a)

```
SELECT *
FROM Osoby
WHERE rok_narodenia>=1980 AND rok_narodenia<=1990;
```

b)

```
SELECT *
FROM Osoby
WHERE rok_narodenia BETWEEN 1980 AND 1990;
```

	id_osoby integer	meno character(30)	rok_narodenia integer
1	106	Novák Peter	1984
2	107	Veselá Mária	1985

A10.

```
SELECT *
FROM Osoby
WHERE rok_narodenia NOT BETWEEN 1980 AND 1990;
```

	id_osoby integer	meno character(30)	rok_narodenia integer
1	109	Kováč Filip	1991
2	108	Nový Pavol	1971
3	105	Mrkvička Ján	1965

A11.

```
SELECT Vlastnictvo.id_parcely, Osoby.meno
FROM Vlastnictvo, Osoby
WHERE Vlastnictvo.id_osoby=Osoby.id_osoby
AND (Osoby.rok_narodenia<1970 OR Osoby.rok_narodenia>1980)
AND NOT (Osoby.meno='Novák Peter' OR Osoby.id_osoby=109)
ORDER BY Vlastnictvo.id_parcely;
```

	id_parcely integer	meno character(30)
1	11	Mrkvička Ján
2	11	Veselá Mária
3	33	Veselá Mária

A12.

a)

```
SELECT DISTINCT Parcely.id_parcely, Parcely.vymera, Osoby.meno
FROM Parcely,Vlastnictvo,Osoby
WHERE Parcely.id_parcely=Vlastnictvo.id_parcely
AND Osoby.id_osoby=Vlastnictvo.id_osoby
AND 1<(SELECT COUNT(id_osoby)
      FROM Vlastnictvo
      WHERE Parcely.id_parcely=Vlastnictvo.id_parcely)
ORDER BY Parcely.id_parcely;
```

b)

```
SELECT DISTINCT P.id_parcely, P.vymera, O.meno
FROM Parcely P,Vlastnictvo V,Osoby O
WHERE P.id_parcely=V.id_parcely
AND O.id_osoby=V.id_osoby
AND 1<(SELECT COUNT(id_osoby)
      FROM Vlastnictvo V
      WHERE P.id_parcely=V.id_parcely)
ORDER BY P.id_parcely;
```

	id_parcely integer	vymera double precision	meno character(30)
1	11	10422	Mrkvička Ján
2	11	10422	Veselá Mária
3	33	987	Nový Pavol
4	33	987	Veselá Mária

A13.

```
SELECT id_osoby, COUNT(id_parcely) AS pocet_parciel
FROM Vlastnictvo
GROUP BY id_osoby
ORDER  BY id_osoby;
```

	id_osoby integer	pocet_parciel bigint
1	105	1
2	106	1
3	107	2
4	108	1

A14.

```
SELECT id_osoby, COUNT(id_parcely)AS pocet_parciel
FROM Vlastnictvo
GROUP BY id_osoby
HAVING 1<COUNT(id_parcely)
ORDER  BY id_osoby;
```

	id_osoby integer	pocet_parciel bigint
1	107	2

A15.

```
SELECT Osoby.id_osoby, COUNT(id_parcely)AS pocet_parciel
FROM Vlastnictvo, Osoby
WHERE Vlastnictvo.id_osoby=Osoby.id_osoby
AND Osoby.rok_narodenia>1980
GROUP BY Osoby.id_osoby
HAVING 1<COUNT(id_parcely)
ORDER  BY id_osoby;
```

	id_osoby integer	pocet_parciel bigint
1	107	2

A16.

```
SELECT *
FROM Osoby
WHERE rok_narodenia<1990 ORDER BY rok_narodenia DESC;
```

	id_osoby integer	meno character(30)	rok_narodenia integer
1	107	Veselá Mária	1985
2	106	Novák Peter	1984
3	108	Nový Pavol	1971
4	105	Mrkvička Ján	1965

A17.

```
SELECT meno
FROM Osoby
WHERE meno LIKE 'N%'
```

	meno character(30)
1	Novák Peter
2	Nový Pavol

A18.

```
SELECT meno
FROM Osoby
WHERE meno LIKE '%Ján%';
```

	meno character(30)
1	Mrkvička Ján

A19.

```
SELECT id_osoby, 2014-rok_narodenia AS vek_2014
FROM Osoby;
```

	id_osoby integer	vek_2014 integer
1	109	23
2	106	30
3	107	29
4	108	43
5	105	49

A20.

```
SELECT CASE COUNT(*) WHEN 0 THEN 'NIE' ELSE 'ÁNO' END
FROM Parcely
WHERE druh_pozemku='záhrada';
```

	case text
1	ÁNO

A21.

```
SELECT druh_pozemku
FROM Parcely
WHERE id_parcely IN (SELECT id_parcely
                     FROM Vlastnictvo
                     WHERE id_osoby=107);
```

	druh_pozemku character varying(20)
1	záhrada
2	orná pôda

A22.

```
SELECT DISTINCT id_osoby
FROM Vlastnictvo
WHERE id_parcely IN
('22','33');
```

	id_osoby integer
1	108
2	106
3	107

A23.

```
SELECT id_osoby, meno
FROM Osoby
WHERE rok_narodenia > ANY (SELECT rok_narodenia
                           FROM Osoby
                           WHERE meno LIKE 'Nov%');
```

	id_osoby integer	meno character(30)
1	109	Kováč Filip
2	106	Novák Peter
3	107	Veselá Mária

A24.

```
SELECT id_osoby, meno
FROM Osoby
WHERE rok_narodenia > ALL (SELECT rok_narodenia
                           FROM Osoby
                           WHERE meno LIKE 'Nov%');
```

	id_osoby integer	meno character(30)
1	109	Kováč Filip
2	107	Veselá Mária

A25.

```
SELECT id_osoby, meno
FROM Osoby
WHERE meno LIKE '%ov%' AND rok_narodenia <= ALL (SELECT rok_narodenia
FROM Osoby WHERE meno LIKE '%ov%');
```

	id_osoby integer	meno character(30)
1	108	Nový Pavol

A26.

```
SELECT id_osoby
FROM Vlastnictvo
WHERE id_parcely = 11
UNION
SELECT id_osoby
FROM Prenajom
WHERE id_parcely = 11;
```

	id_osoby integer
1	106
2	107
3	105

A27.

```
SELECT id_parcely AS cp, vymera/10000 AS vym
FROM Parcely;
```

	cp integer	vym double precision
1	11	1.0422
2	22	0.0344
3	33	0.0987

A28.

```
SELECT X.id_osoby AS prva, Y.id_osoby AS druha
FROM Osoby X, Osoby Y
WHERE X.rok_narodenia=Y.rok_narodenia AND
X.id_osoby<Y.id_osoby;
```

	prva integer	druha integer

A29.

a)

```
SELECT id_osoby, meno, rok_narodenia, adresa
FROM Osoby INNER JOIN Obyvatelia
ON Osoby.id_osoby=Obyvatelia.id_obyvatela;
```

	id_osoby integer	meno character(30)	rok_narodenia integer	adresa text
1	105	Mrkvička Ján	1965	Zelená 4, 976 64 Beňuš
2	107	Veselá Mária	1985	Ružová 5, 976 64 Beňuš

b)

```
SELECT *
FROM Osoby RIGHT OUTER JOIN Obyvatelia
ON Osoby.id_osoby=Obyvatelia.id_obyvatela;
```

	id_osoby integer	meno character(30)	rok_narodenia integer	id_obyvatela integer	meno_obyvatela character(30)	adresa text
1	105	Mrkvička Ján	1965	105	Mrkvička Ján	Zelená 4, 976 64 Beňuš
2	107	Veselá Mária	1985	107	Veselá Mária	Ružová 5, 976 64 Beňuš
3				209	Rýchly Peter	Tehelná 12, 949 01 Nitra
4				210	Kováčová Eva	Vysoká 13, 010 01 Žilina

c)

```
SELECT *
FROM Osoby LEFT OUTER JOIN Obyvatelia
ON Osoby.id_osoby=Obyvatelia.id_obyvatela;
```

	id_osoby integer	meno character(30)	rok_narodenia integer	id_obyvatela integer	meno_obyvatela character(30)	adresa text
1	109	Kováč Filip	1991			
2	106	Novák Peter	1984			
3	107	Veselá Mária	1985	107	Veselá Mária	Ružová 5, 976 64 Beňuš
4	108	Nový Pavol	1971			
5	105	Mrkvička Ján	1965	105	Mrkvička Ján	Zelená 4, 976 64 Beňuš

d)

```
SELECT *
FROM Osoby FULL OUTER JOIN Obyvatelia
ON Osoby.id_osoby=Obyvatelia.id_obyvatela;
```

	id_osoby integer	meno character(30)	rok_narodenia integer	id_obyvatela integer	meno_obyvatela character(30)	adresa text
1	105	Mrkvička Ján	1965	105	Mrkvička Ján	Zelená 4, 976 64 Beňuš
2	106	Novák Peter	1984			
3	107	Veselá Mária	1985	107	Veselá Mária	Ružová 5, 976 64 Beňuš
4	108	Nový Pavol	1971			
5	109	Kováč Filip	1991			
6				209	Rýchly Peter	Tehelná 12, 949 01 Nitra
7				210	Kováčová Eva	Vysoká 13, 010 01 Žilina

e)

```
SELECT id_osoby, meno, rok_narodenia, adresa
FROM Osoby LEFT OUTER JOIN Obyvatelia
ON Osoby.id_osoby=Obyvatelia.id_obyvatela;
```

	id_osoby integer	meno character(30)	rok_narodenia integer	adresa text
1	109	Kováč Filip	1991	
2	106	Novák Peter	1984	
3	107	Veselá Mária	1985	Ružová 5, 976 64 Beňuš
4	108	Nový Pavol	1971	
5	105	Mrkvička Ján	1965	Zelená 4, 976 64 Beňuš

A21.

b)

```
SELECT P.druh_pozemku
FROM Parcely P, Vlastnictvo V
WHERE P.id_parcely=V.id_parcely
AND V.id_osoby=107;
```

	druh_pozemku character varying(20)
1	záhrada
2	orná pôda

c)

```
SELECT P.druh_pozemku
FROM Parcely P NATURAL JOIN Vlastnictvo V
WHERE V.id_osoby=107;
```

d)

```
SELECT P.druh_pozemku
FROM Parcely P INNER JOIN Vlastnictvo V ON P.id_parcely=V.id_parcely
WHERE V.id_osoby=107;
```

e)

```
SELECT druh_pozemku
FROM Parcely
WHERE id_parcely IN (SELECT id_parcely
                     FROM Vlastnictvo
                     WHERE  id_osoby=107);
```

f)

```
SELECT druh_pozemku
FROM Parcely
WHERE id_parcely = ANY (SELECT id_parcely
                     FROM Vlastnictvo
                     WHERE id_osoby=107);
```

g)

```
SELECT druh_pozemku
FROM Parcely P
WHERE EXISTS (SELECT id_parcely
              FROM Vlastnictvo V
              WHERE P.id_parcely=V.id_parcely AND V.id_osoby=107);
```

h)

```
SELECT druh_pozemku
FROM Parcely P
WHERE 107 IN (SELECT id_osoby
              FROM Vlastnictvo V
              WHERE V.id_parcely=P.id_parcely);
```

A30.

```
SELECT Osoby.id_osoby, Osoby.meno
FROM Obyvatelia FULL JOIN Osoby ON Obyvatelia.id_osoby=Osoby.id_osoby
WHERE adresa IS NULL;
```

	id_osoby integer	meno character(30)
1	106	Novák Peter
2	108	Nový Pavol
3	109	Kováč Filip

A31.

```
INSERT INTO Osoby (id_osoby, meno, rok_narodenia)
SELECT id_osoby, meno, rok_narodenia
FROM Najomnici
WHERE meno NOT IN (SELECT meno
                   FROM Osoby);
```

A32.

```
UPDATE Parcely
SET vymera=989
WHERE id_parcely=33;
```

A33.

```
DELETE FROM Osoby
WHERE id_osoby IN (303, 304);
```

Dopyty zo zadania č. 2.2

B1.

```
SELECT *
FROM Firmy;
```

B2.

```
SELECT *
FROM Firmy
WHERE predm_cinn='upratovacie práce';
```

B3.

```
SELECT meno, priezvisko, vek
FROM Obyvatelia
WHERE meno='Peter' AND vek < 35;
```

B4.

```
SELECT meno AS krstne_meno, priezvisko, vek
FROM Obyvatelia
WHERE meno='Peter' OR meno = 'Ivan';
```

B5.

```
SELECT *
FROM Obyvatelia
WHERE meno LIKE 'J%';
```

B6.

```
SELECT *
FROM Obyvatelia
WHERE meno LIKE 'J____';
```

B7.

```
SELECT priezvisko, meno, id, vek*12 as vek_v_mesiacoch
FROM Obyvatelia
ORDER BY priezvisko;
```

B8.

```
SELECT avg(vek)
FROM Obyvatelia;
```

B9.

```
SELECT obce.nazov
FROM Obyvatelia, Obce
WHERE meno='Peter' AND priezvisko='Oravec' AND Obce.id=Obyvatelia.obec_id;
```

B10.

```
SELECT *
FROM Obyvatelia, Obce
WHERE nazov='Blatné' AND Obce.id=Obyvatelia.obec_id;
```

B11.

```
(SELECT nazov
FROM Priroda)
INTERSECT
(SELECT nazov
FROM Vodne_plochy);
```

B12.

```
(SELECT nazov
FROM Priroda)
EXCEPT
(SELECT nazov
FROM Vodne_plochy);
```

B13.

```
SELECT *
FROM Obce
WHERE okres_id=(SELECT id
            FROM Okresy
            WHERE nazov='Pezinok');
```

B14.

```
SELECT count(*)
FROM Obyvatelia;
```

B15.

```
SELECT meno
FROM Obyvatelia
GROUP BY meno;
```

B16.

```
SELECT meno, count(meno)
FROM Obyvatelia
GROUP BY meno;
```

B17.

```
SELECT meno, priezvisko, vek
FROM Obyvatelia
ORDER BY vek DESC
LIMIT 3;
```

B18.

```
INSERT INTO Obyvatelia(meno, priezvisko, vek, obec_id, firma_id)
VALUES ('Martin','Králik',37,34,7);
```

B19.

```
SELECT *
FROM Obyvatelia
WHERE id > 152;
```

B20.

```
DELETE
FROM Obyvatelia
WHERE meno='Martin' AND priezvisko='Králik';
```

B21.

```
INSERT INTO Firmy(nazov, predm_cinn, sidlo, cesta_id, obec_id)
VALUES ('Solid','pravnicka kancelaria','Sládkovičova 15, Bratislava
81364',4318,5);
```

B22.

```
UPDATE Obyvatelia
SET firma_id=5
WHERE id=119;
```

B23.

```
SELECT *
FROM Obyvatelia
WHERE id=119;
```

B24.

```
DELETE
FROM Obyvatelia
WHERE id > 134 AND id < 137;
```

B25.

```
INSERT INTO Body_zaujmu (nazov, okres_id, the_geom)
SELECT nazov, okres_id, the_geom
FROM Obce;
```

B26.

```
SELECT *
FROM Obyvatelia a, Obyvatelia b
WHERE a.meno = b.meno AND a.vek = b.vek AND a.id > b.id;
```

B27.

```
SELECT SUM(vek)
FROM Obyvatelia;
```

B28.

```
SELECT *
FROM Obyvatelia
WHERE firma_id=(SELECT id
            FROM Firmy
            WHERE nazov='Xcomp');
```

B29.

```
SELECT nazov
FROM Okresy
WHERE id=(SELECT okres_id
      FROM Obce
      WHERE id=(SELECT obec_id
            FROM Obyvatelia
            WHERE meno='Imrich' AND priezvisko='Struhár'));
```

B30.

```
SELECT *
FROM Cesty
WHERE nazov LIKE '%ale%' AND okres_id=(SELECT id
                              FROM Okresy
                              WHERE nazov='Senec');
```

B31.

```
SELECT *
FROM Okresy, Obce
WHERE Okresy.nazov = Obce.nazov;
```

B32.

```
UPDATE Obyvatelia SET vek=vek+5
WHERE meno='Vojtech' AND priezvisko='Kováčik';
```

B33.

```
INSERT INTO Firmy(nazov, predm_cinn, sidlo, cesta_id, obec_id)
VALUES ('Fix','cestovna kancelaria','Sládkovičova 15, Bratislava
81364',4318,5);
INSERT INTO Firmy(nazov, predm_cinn, sidlo, cesta_id, obec_id)
VALUES ('Job','pracvona agentura','Sládkovičova 15, Bratislava
81364',4318,5);
INSERT INTO Obyvatelia(meno, priezvisko, vek, obec_id, firma_id)
VALUES ('Peter','Malý',25,14,7);
INSERT INTO Obyvatelia(meno, priezvisko, vek, obec_id, firma_id)
VALUES ('František','Lacko',45,35,7);
```

B34.

```
SELECT meno, priezvisko
FROM Obyvatelia
ORDER BY priezvisko DESC;
```

B35.

```
SELECT meno, priezvisko
FROM Obyvatelia
ORDER BY priezvisko ASC;
```

B36.

```
SELECT avg(vek)
FROM Obyvatelia
WHERE vek > 18;
```

B37.

```
SELECT *
FROM Okresy
WHERE nazov <> 'Senec';
```

B38.

```
SELECT *
FROM Obyvatelia
OFFSET 30
LIMIT 10;
```

B39.

```
SELECT *
FROM Obce LEFT JOIN Okresy ON Obce.okres_id=Okresy.id;
```

B40.

```
SELECT nazov
FROM Obce
WHERE id IN (SELECT obec_id
             FROM obyvatelia);
```

B41.

```
SELECT nazov
FROM Obce
WHERE id NOT IN (SELECT obec_id
               FROM Obyvatelia);
```

B42.

```
SELECT vek
FROM Obyvatelia
ORDER BY vek ASC
LIMIT 1;
```

Dopyty zo zadania č. 2.3

B43.

```
CREATE TABLE Zamestnanci(
id INT,
meno VARCHAR(50),
priezvisko VARCHAR(50),
pohlavie CHAR(1),
dat_nar DATE,
firma_id INT,
PRIMARY KEY (id)
);
```

B44.

```
CREATE TABLE Tovar(
nazov VARCHAR(30),
popis VARCHAR(200),
sklad VARCHAR(30),
cena DECIMAL(7,2),
PRIMARY KEY (nazov,popis)
);
```

B45.

```
ALTER TABLE Tovar ADD pocet INT;
```

B46.

```
ALTER TABLE Tovar DROP sklad;
```

B47.

```
DROP TABLE Tovar;
```

B48.

```
CREATE TABLE Produkty(
id INT,
nazov VARCHAR(20),
typ VARCHAR(30) DEFAULT 'základný',
popis VARCHAR (200),
cena DECIMAL(7,2),
PRIMARY KEY (id)
);
```

B49.

```
INSERT INTO Produkty(id, nazov, popis, cena)
VALUES (1, 'tanier - biely', 'priemer 15 cm', 2.50);
INSERT INTO Produkty(id, nazov, popis, cena)
VALUES (2, 'váza', 'výška 17 cm, polomer 5cm', 3.50);
INSERT INTO Produkty(id, nazov, popis, cena)
VALUES (3, 'hrnček', 'objem 0,2 l', 2.50);
```

B50.

```
SELECT *
FROM Produkty;
```

B51.

```
CREATE TABLE Objednavky (
cislo_ob INT,
prijata DATE,
produkt_id INT,
doprava VARCHAR(15),
pocet VARCHAR(15),
cena DECIMAL(7,2),
PRIMARY KEY (cislo_ob),
CHECK (doprava IN ('osobný odber','pošta','kuriér'))
);
```

B52.

```
ALTER TABLE Objednavky
ADD CONSTRAINT obj_fk
FOREIGN KEY (produkt_id) REFERENCES Produkty (id);
```

B53.

```
INSERT INTO Objednavky(cislo_ob, prijata, produkt_id, doprava, pocet, cena)
VALUES (1, '2012-5-16', 2, 'pošta', 6, 15.00);

INSERT INTO Objednavky(cislo_ob, prijata, produkt_id, doprava, pocet, cena)
VALUES (2, '2012-5-16', 23, 'pošta', 6, 15.00);
```

B54.

```
CREATE DOMAIN pohlavie_dom AS VARCHAR(7)
DEFAULT 'neznáme'
CHECK (VALUE IN ('neznáme','muž','žena'));
```

B55.

```
CREATE TABLE Studenti(
id INT,
meno VARCHAR(100),
pohlavie pohlavie_dom,
skupina VARCHAR(50),
PRIMARY KEY (id)
);
```

B56.

```
INSERT INTO Studenti VALUES (1, 'Peter Mak', 'muž', '2-3');

INSERT INTO Studenti VALUES (2, 'Jana Malá', 'žena', '3-1');

INSERT INTO Studenti VALUES (3, 'Dušan Benko', '1', '3-2');

INSERT INTO Studenti(id, meno) VALUES (3, 'Dušan Benko');
```

B57.

```
CREATE DOMAIN psc_dom AS INT(5);
```

B58.

```
ALTER DOMAIN psc_dom ADD CONSTRAINT obmedz_dlzky CHECK (char_length(VALUE)
= 5);
```

B59.

```
CREATE TABLE Adresy(
id INT,
supis_cislo VARCHAR(20),
orient_cislo VARCHAR(20),
ulica_id INT,
psc psc_dom,
PRIMARY KEY (id)
);

CREATE TABLE Ulice(
id INT,
nazov VARCHAR(100),
obec_id int,
psc psc_dom,
lokalita VARCHAR(500),
PRIMARY KEY (id)
);

ALTER TABLE Adresy
  ADD CONSTRAINT adresy_ulica_id_fkey FOREIGN KEY (ulica_id)
      REFERENCES Ulice (id);
```

B60.

```
INSERT INTO Ulice VALUES (1, 'Prvá', 1, 54689, 'Pod lesom');
INSERT INTO Ulice VALUES (2, 'Druhá', 1, 54689, 'Pod lesom');
INSERT INTO Ulice VALUES (3, 'Tretia', 1, 54689, 'Pod lesom');

INSERT INTO Adresy VALUES (1, 155, 1, 1, 54689);
INSERT INTO Adresy VALUES (2, 156, 3, 1, 54689);
INSERT INTO Adresy VALUES (3, 157, 7, 1, 54689);
```

B61.

```
CREATE DOMAIN test AS CHAR(15);
```

B62.

```
DROP DOMAIN test;
```

B63.

```
CREATE VIEW w_produkty AS
SELECT id, nazov, cena
FROM Produkty;
```

B64.

```
CREATE VIEW Obyv_18 AS
SELECT *
FROM Obyvatelia
WHERE vek >= 18;
```

B65.

```
CREATE OR REPLACE VIEW Obyv_18 AS
SELECT *
FROM Obyvatelia
WHERE vek >= 18 AND vek < 65;
```

B66.

```
DROP VIEW Obyv_18;
```

B67.

```
CREATE VIEW Most_obyv AS
SELECT *
FROM Obyvatelia
WHERE obec_id=(SELECT id
            FROM Obce
            WHERE nazov='Most pri Bratislave');
```

B68.

```
CREATE VIEW Obce_okresy AS
SELECT Obce.nazov AS obce_nazov, Okresy.nazov AS okresy_nazov
FROM Obce LEFT JOIN Okresy ON Obce.okres_id=Okresy.id;
```

B69.

```
CREATE OR REPLACE VIEW Obce_okresy AS
SELECT Obce.nazov AS obce_nazov, Okresy.nazov AS okresy_nazov
FROM Obce LEFT JOIN Okresy ON Obce.okres_id=Okresy.id
ORDER BY Okresy.nazov, Obce.nazov;
```

B70.

```
CREATE OR REPLACE VIEW Obce_okresy AS
SELECT Obce.nazov AS obce_nazov, Okresy.nazov AS okresy_nazov
FROM Obce LEFT JOIN Okresy ON Obce.okres_id=Okresy.id
ORDER BY char_length(obce.nazov);
```

B71.

```
DROP TABLE Objednavky;
```

Dopyty zo zadania č. 2.4

A34.

```
CREATE TABLE Osoby (
id_osoby INT,
meno CHAR (30),
rok_narodenia DATE,
PRIMARY KEY (id_osoby)
);
```

A35.

```
ALTER TABLE Parcely
ADD bonita INT;
```

A36.

```
ALTER TABLE Parcely
DROP bonita;
```

A37.

```
CREATE TABLE Budovy (
id_budovy INT,
vymera FLOAT
);

ALTER TABLE Budovy
ADD PRIMARY KEY (id_budovy);
```

A38.

```
CREATE DOMAIN Druh_pozemku AS SMALLINT
CHECK (VALUE IN (2,3,4,5,6,7,10,11,13,14));
```

A39.

```
CREATE DOMAIN Vymera AS REAL
CHECK (VALUE >=0);
```

A40.

```
CREATE DOMAIN Druh_stavby AS SMALLINT
CHECK (VALUE >=1 AND VALUE <=23);
```

A41.

```
CREATE INDEX Meno_rok_narodenia ON Osoby (meno, rok_narodenia);
```

A42.

```
CREATE VIEW Obyvatelia_BA AS
SELECT id_osoby, meno, adresa
FROM Osoby
WHERE adresa LIKE '%Bratislava%';
```

Dopyty zo zadania č. 3

B72.

```
SELECT id, nazov, the_geom
FROM Body_zaujmu;
```

B73.

```
SELECT id, nazov, AsText(the_geom)
FROM Body_zaujmu;
```

B74.

```
SELECT id, nazov, AsText(the_geom)
FROM Vodne_plochy;

SELECT id, nazov, AsText(the_geom)
FROM Cesty;
```

B75.

```
CREATE TABLE Body (
id INT PRIMARY KEY,
oznacenie VARCHAR(10),
typ VARCHAR(10));
```

B76.

```
SELECT AddGeometryColumn('','Body','the_geom','-1','POINT',2);
```

B77.

```
INSERT INTO Body
VALUES (1,'MOPI','p. s.',GeomFromText('POINT(17.30655 48.334538)'));
```

B78.

```
INSERT INTO Body
VALUES (2,'bod2','p. s.',GeomFromText('POINT(17.40655 48.434538)'));
INSERT INTO Body
VALUES (3,'bod3','p. s.',GeomFromText('POINT(17.20655 48.434538)'));
```

B79.

```
CREATE TABLE Linie (
id INT PRIMARY KEY,
oznacenie VARCHAR(10),
typ VARCHAR(10));

SELECT AddGeometryColumn('','Linie','the_geom','-1','LINESTRING',2);
```

B80.

```
INSERT INTO Linie
VALUES (1,'prva','A',GeomFromText('LINESTRING(17.25 48.31, 17.26 48.32)'));
```

B82.

```
SELECT *
FROM Linie;
```

B83.

```
CREATE TABLE budovy (
id INT PRIMARY KEY,
oznacenie VARCHAR(10),
typ VARCHAR(10));

SELECT AddGeometryColumn('','Budovy','the_geom','4326','POLYGON',2);
```

B84.

```
INSERT INTO Budovy
VALUES (1,'prva','A',GeomFromText('POLYGON((17.25 48.31, 17.24 48.32, 17.25
48.33, 17.26 48.32, 17.25 48.31))',4326));
```

B86.

```
SELECT *
FROM Budovy;
```

B87.

```
UPDATE Body
SET the_geom=GeomFromText('POINT(17.3021 48.3349)')
WHERE id=1;
```

B88.

```
SELECT Obce.nazov
FROM Obce, Okresy
WHERE ST_Intersects(Obce.the_geom,Okresy.the_geom)
AND Okresy.nazov='Senec';
```

B89.

```
SELECT ST_Distance(a.the_geom,b.the_geom)
FROM Obce a, Obce b
WHERE a.nazov='Malacky' AND b.nazov='Pernek';
```

B90.

```
SELECT id, nazov, ST_Buffer(the_geom,0.005)
FROM Vodne_plochy;
```

B91.

```
CREATE OR REPLACE VIEW Vodn_buff AS(
SELECT id, nazov, ST_Buffer(the_geom,0.005)
FROM Vodne_plochy);
```

B93.

```
SELECT Obce.nazov, ST_Distance(Obce.the_geom,Vodne_plochy.the_geom)
FROM Obce, Vodne_plochy
ORDER BY Distance(Obce.the_geom,Vodne_plochy.the_geom)
LIMIT 10;
```

B94.

```
SELECT id, nazov, ST_Length(the_geom)
FROM Cesty
WHERE id < 15;
```

B95.

```
SELECT id, nazov, ST_Length(the_geom)
FROM Cesty
WHERE nazov IS NOT NULL
ORDER BY ST_Length(the_geom) DESC
LIMIT 3;
```

B96.

```
SELECT Cesty.id, ST_Intersection(Cesty.the_geom,Priroda.the_geom)
FROM Cesty, Priroda
WHERE Priroda.nazov='Bažantnica';
```

B97.

```
CREATE TABLE Bazantica AS(
SELECT Cesty.id, ST_Intersection(cesty.the_geom,priroda.the_geom)
FROM Cesty, Priroda
WHERE Priroda.nazov='Bažantnica');
```

B99.

```
SELECT ST_Area(the_geom)
FROM Vodne_plochy
WHERE nazov='Slnečné jazerá';
```

B100.

```
SELECT Cesty.id, Cesty.the_geom
FROM Priroda, Cesty
WHERE ST_Touches(Priroda.the_geom,Cesty.the_geom);
```

B101.

```
CREATE TABLE Dotyky AS (
SELECT Cesty.id, Cesty.the_geom
FROM Priroda, Cesty
WHERE ST_Touches(Priroda.the_geom,Cesty.the_geom));
```

B103.

```
SELECT Cesty.id, Cesty.nazov, Cesty.the_geom
FROM Priroda, Cesty
WHERE ST_Within(Cesty.the_geom,Priroda.the_geom);
```

B104.

```
CREATE TABLE Cesty_v_lesoch AS (
SELECT Cesty.id, Cesty.nazov, Cesty.the_geom
FROM Priroda, Cesty
WHERE ST_Within(Cesty.the_geom,Priroda.the_geom));
```

B106.

```
SELECT ST_Union(the_geom) AS the_geom
FROM Vodne_plochy
WHERE id = 166 OR id= 169;
```

B107.

```
CREATE TABLE Nazov_tab AS (
SELECT ST_Union(the_geom) AS the_geom
FROM Vodne_plochy
WHERE id = 166 OR id= 169);
```

Zoznam obrázkov

Zoznam tabuliek

Zoznam príloh

Príloha č. 1: Databáza `Vlastnictvo_parciel` v softvérovom prostredí MS Access.

Príloha č. 2: Databáza `Vlastnictvo_parciel` v databázovom systéme PostgreSQL.

Príloha č. 3: Databáza `BA_kraj`.

Príloha č. 4: Súbory priestorových dát vo formáte *Shapefile*.

Použitá literatúra

ARLOW, J. - NEUSTADT, I.: UML a unifikovaný proces vývoje aplikací: průvodce analýzou a návrhem objektově orientovaného softwaru. 1. vyd. Brno: Computer Press, 2003. 568 s. ISBN 80-7226-947-X

CONNOLLY, T. - BEGG, C. - HOLOWCZAK, R.: Mistrovství – Databáze. Profesionální průvodce tvorbou efektivních databází. Brno: Computer Press, 2009. 584 s. ISBN 978-80-251-2328-7

DELIKÁT, T.: Základy databázových systémov. Bratislava: DELINT, 2006. 210 s. ISBN 80-969484-4-X

DOBEŠOVÁ, Z.: Databázové systémy v GIS. Olomouc: Vydavatelství UP, 2004. 76 s. ISBN 80-244-0891-0

ĎURAČIOVÁ, R.: Databázové systémy v GIS. Bratislava: Slovenská technická univerzita v Bratislave v Nakladateľstve STU, 2014. 178 s. ISBN: 978-80-227-4292-4

KANISOVÁ, H. - MÜLLER, M.: UML srozumitelně. 2. aktualiz. vyd. Brno: Computer Press, 2006. 176 s. ISBN 80-251-1083-4

KOREŇ, M.: Databázové systémy. Zvolen: Technická univerzita vo Zvolene Lesnícka fakulta, 2009. 90 s. ISBN 978-80-228-2084-4

LACKO, Ľ.: 1001 tipů a triků pro SQL. Brno: Commputer Press, 2011. 416 s. ISBN 978-80-251-3010-0

PAGE-JONES, M.: Základy objektově orientovaného návrhu v UML. Praha: Grada, 2001. 368 s. ISBN 80-247-0210-X

PLACHETKA, T.: Úvod do databázových systémov 2011/2012. [online]. [cit. 2012-07-26]. Dostupné z URL: <http://www.dcs.fmph.uniba.sk/~plachetk/TEACHING/DB2011/index.html>.

POKORNÝ, J.: SQL ve třech lekcích. *GeoInfo*, 7, 2000, č. 2-4, príloha Škola, ISSN 1212-4311

SHEAKER, S. - XIONG, H.: Encyklopedia of GIS. New York: Springer-Verlag, 2008. 1370 s. ISBN 978-0-378-30858-6

STN 73 0401-3 Terminológia v geodézii a kartografii. Časť 3: Terminológia kartografie a geografických informačných systémov. Bratislava: SÚTN. 2009. 92 s.

Internetové zdroje

Eclipse Public Licence. [online]. [cit. 2014-08-19] Dostupné z URL: <https://www.eclipse.org/legal/epl-v10.html>.

EPSG. [online]. [cit. 2014-11-20]. Dostupné z URL: <http://www.epsg.org>.

GeoServer. [online]. [cit. 2014-09-24]. Dostupné z URL: <http://geoserver.org>.

GNU General Public. License. [online]. [cit. 2014-08-19] Dostupné z URL: <http://www.gnu.org/copyleft/gpl.html>.

MapServer. [online]. [cit. 2014-09-24]. Dostupné z URL: <http://mapserver.org>.

Microsoft. [online]. [cit. 2013-10-08]. Dostupné z URL: <http://www.microsoft.com>.

Microsoft Developer Network. [online]. [cit. 2013-08-15]. Dostupné z URL: <http://msdn.microsoft.com>.

MySQL [online]. [cit. 2013-10-02]. Dostupné z URL: <http://www.mysql.com>.

OGC 06-103r4: OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture. 2011 [online]. [cit. 2014-08-19]. Dostupné z URL: <http://www.opengeospatial.org/standards/sfa>.

OGC 06-104r4: OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 2: SQL option. 2010 [online]. [cit. 2014-08-19]. Dostupné z URL: <http://www.opengeospatial.org/standards/sfs>.

Open Geospatial Consortium. [online]. [cit. 2014-01-16] Dostupné z URL: <http://www.opengeospatial.org>.

Oracle. [online]. [cit. 2013-10-03]. Dostupné z URL: < http://www.oracle.com>.

PostgreSQL. [online]. [cit. 2013-08-11] Dostupné z URL: <http://www.postgresql.org>.

QGIS. [online]. [cit. 2014-08-18] Dostupné z URL: <http://www.qgis.org/en/site/>.

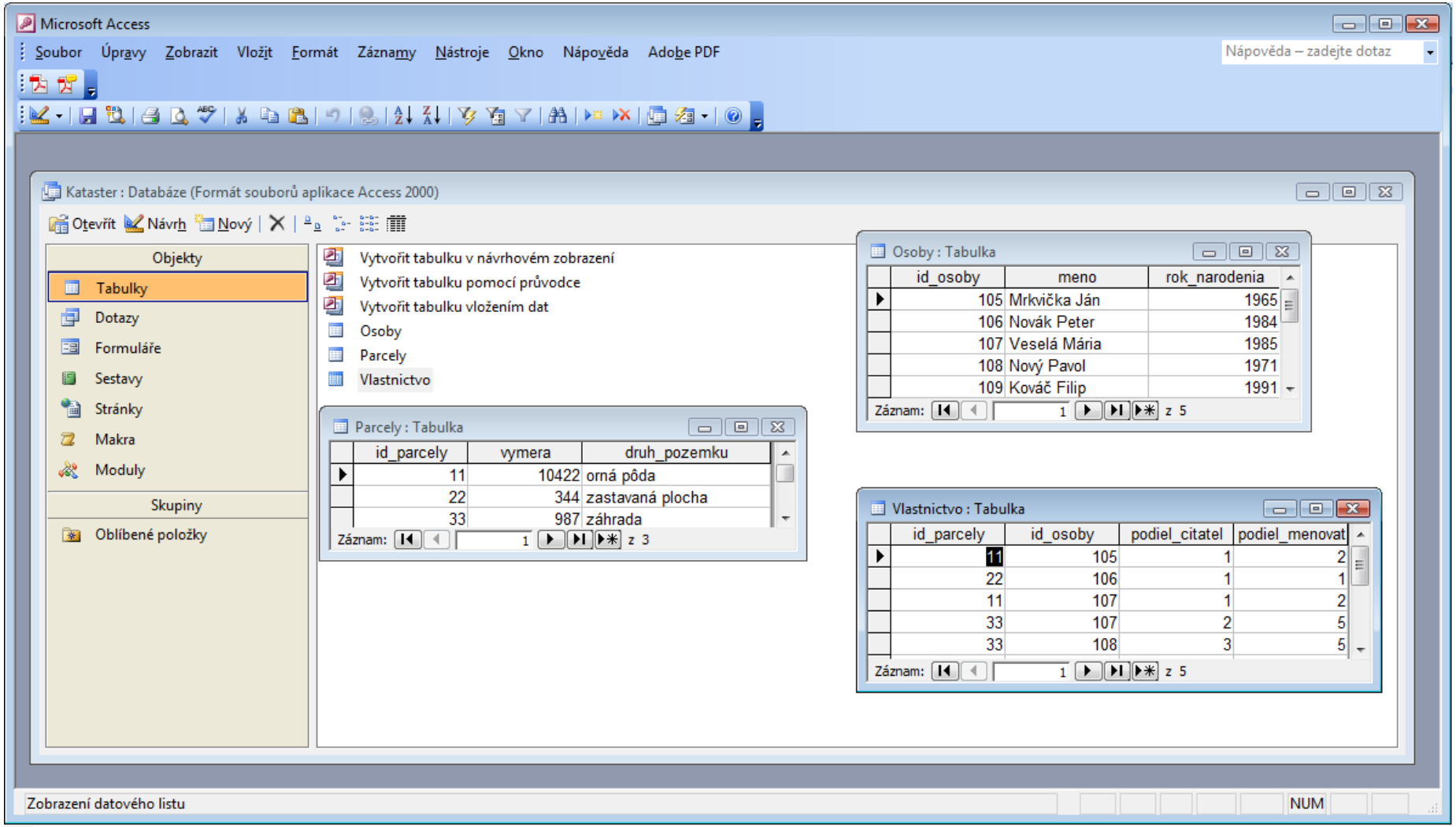
StarUML, The Open Source UML/MDA Platform. [online]. [cit. 2013-08-11] Dostupné z URL: <http://staruml.sourceforge.net>.

Prílohy

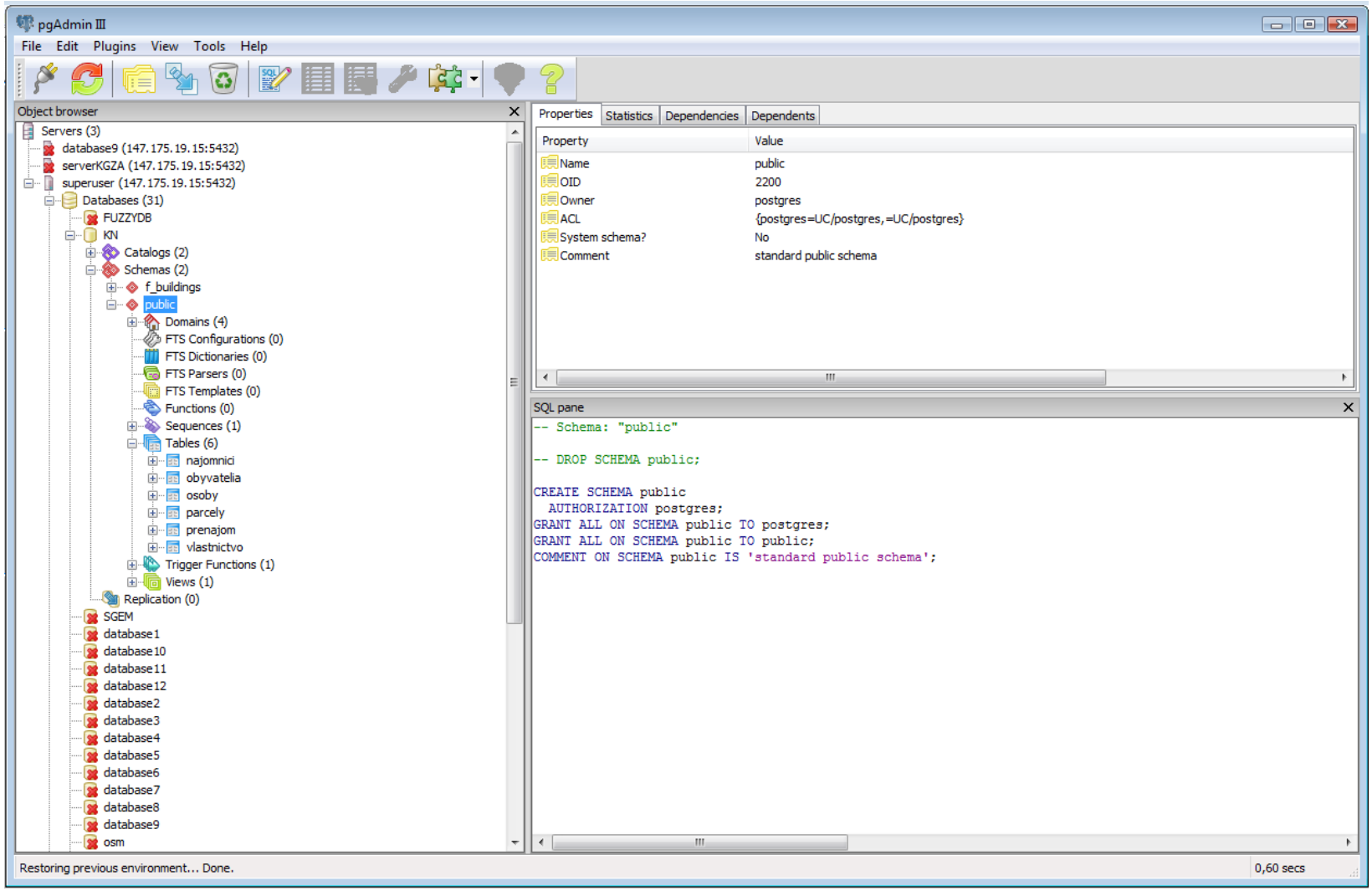
Všetky súbory sú dostupné na:

ftp://147.175.19.15/dsgis/

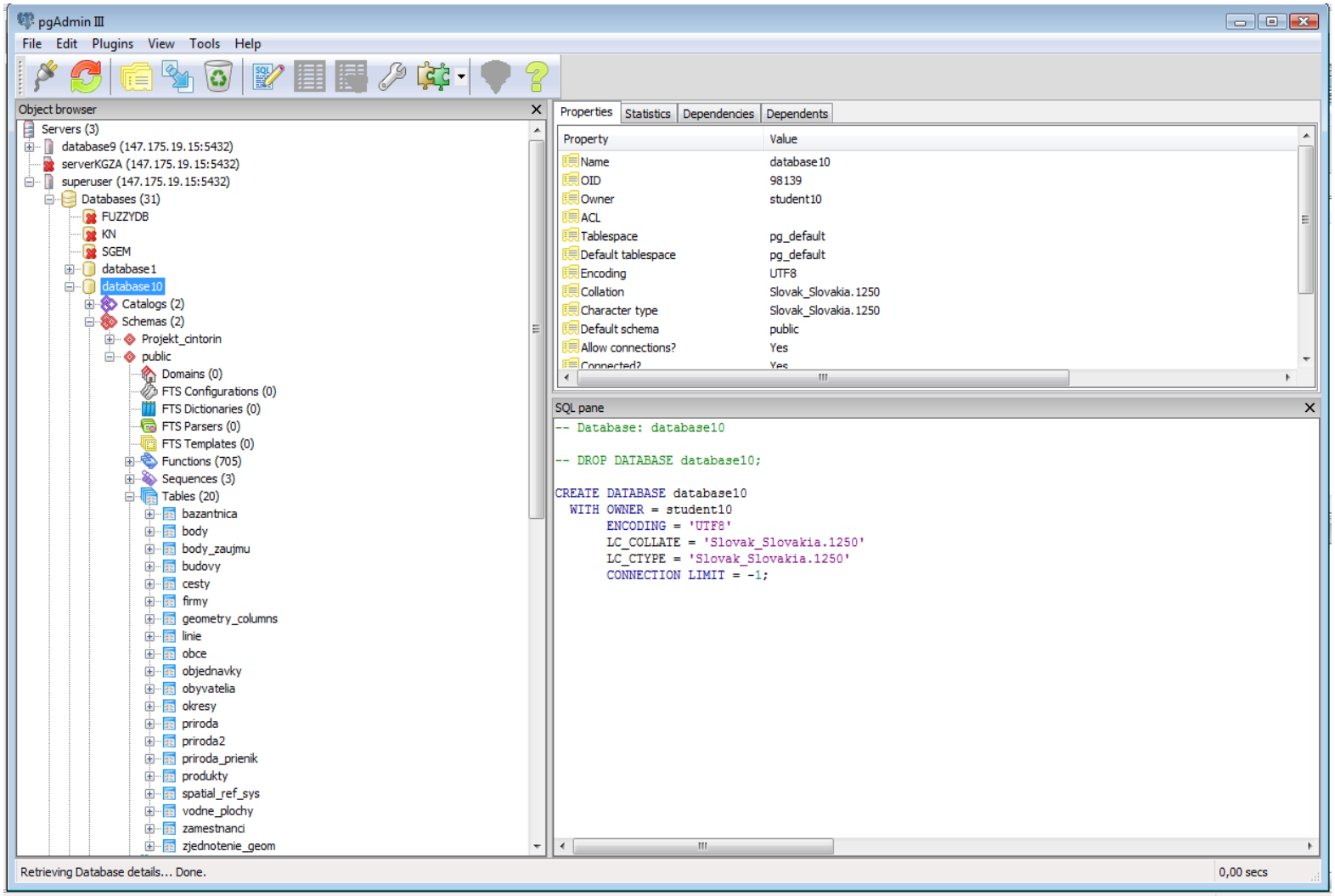
Príloha č. 1
(súbor Vlastnictvo_parciel_MSA.zip)



Príloha č. 2
(súbor Vlastnictvo_parciel_PSQL.zip)



Príloha č. 3
(súbor BA_kraj.zip)



Príloha č. 4
(súbor Priestorove_data_shp.zip)