Univerzita Palackého v Olomouci
Přírodovědecká fakulta
Katedra geoinformatiky

Markéta SOLANSKÁ

SYNCHRONIZACE A REPLIKACE GEODAT V PROSTŘEDÍ ESRI PLATFORMY

Diplomová práce

Vedoucí práce: doc. RNDr. Vilém Pechanec, Ph.D.

Olomouc 2014

Čestné prohlášení

Prohlašuji, že jsem závěrečnou práci magisterského studia oboru Geoinformatika vypracovala samostatně pod vedením RNDr. Viléma Pechance, Ph.D.

Všechny použité materiály a zdroje jsou citovány s ohledem na vědeckou etiku, autorská práva a zákony na ochranu duševního vlastnictví.

Všechna poskytnutá i vytvořená digitální data nebudu bez souhlasu školy poskytovat.

V Olomouci 19. dubna 2014 Markéta SOLANSKÁ

Ráda bych poděkovala doc. RNDr. Vilému Pechancovi, Ph.D. za ochotné vedení této práce, pečlivé korekce a věcné připomínky.

Děkuji také konzultantu Tomáši Vondrovi, za jeho cenné rady a odborný vhled, který vnesl do této práce, stejně tak jako i jeho kolegovi Pavlovi Stěhule.

Dále děkuji konzultantům Boudewijn van Leeuwen a Zalan Tobak působích na Univerzitě v Szegedu v Maďarsku za inspirativní podněty při vypracování této práce.

zadání

Obsah

Seznam obrázků

Seznam tabulek

ÚVOD

Dnešní trend je ukládat a ponechávat stále více dat pouze v digitální podobě. Mnoho dokumentů už se vůbec netiskne do papírové podoby, tím spíš pokud dnes existují elektronické podpisy, díky kterým je tištěná verze naprosto zbytečná. S přibývajícím počtem dat je však třeba řešit komplikace, které počítačová data přinášejí. Počítačoví experti řeší například otázky, kam ukládat tak velké množství dat, jak data efektivně aktualizovat, jak zabránit poškození dat ať už způsobených lidským faktorem či fyzickým poškozením hardware. V případě, že se poškodí disk, můžeme často během okamžiku přijít o všechna data, někdy však pro ztrátu dat stačí pouze stisknout tlačítko na klávesnici. Určitě už se Vám nejednou stalo, že jste se nemohli přihlásit do svého účtu na internetu z důvodu přetížení serveru. I to je problém, který velké množství dat a velký počet uživatelů přináší. Jak tedy pracovat s těmito objemy, jak zabránit komplikacím, které mohou poškodit či zcela zničit celou dosavadní práci, a jak zrychlit celý proces práce s daty?

Řešením velkého počtu výše uvedených problémů může být ukládaní dat do databáze a jejich následná replikace. Replikací je myšlena pokročilá funkce, která zajišťuje kopii dat na více serverů. Nabízí ji většina dnešních databázových serverů, zajišťuje větší robustnost databáze a vysokou dostupnost dat. Replikaci lze využít ve všech odvětvích, které pracují s daty. Výjimkou tedy není ani geoinformatika, která pracuje s velkými objemy dat, které navíc nesou informaci o geografické poloze. Právě reprezentace geografické polohy, skrze textový zápis souřadnice daných bodů, může způsobit razantní zvýšení velikosti dat. Například u webových dat se navím musí řešit častý přístup k databázi, protože každé posunutí výřezu či přiblížiní, resp. oddálení výřezu mapy, je samostatným dotazem, který musí kapacita serveru zvládat. Při představě, že si uživatel bude posouvat výřez mapy po 50m, může to způsobit velkou zátěž pro server. V tomto případě je potřeba řešit replikaci z důvodu rozložení zátěže.

Z mého pohledu data středně velkého až velkého projektu je vhodnější ukládata do databáze než jiných formátů typu shapefile, Microsoft Access nebo obyčejného tabulkového procesoru. Nabízí nám to sofistikované uložení dat, snadné propojení jednotlivých vrstev, snadnou přenostitelnost dat, možnost relačního propojení dat nebo efektivní vyhledávání. Replikace samotná se poté využívá pro kopii dat a následnou aktualizaci změn, která v databázi nastanou.

Replikaci ocení uživatelé pracující na společném projektu, distribuovaná pracoviště i společnosti s velkým množstvím důležitých dat, jejichž kopie je rozhodující pro jejich fungování. Dobrým příkladem využitelnosti replikace je také nový trend využívání offline aplikací v mobilních telefonech. Databáze se vždy replikuje do mobilního telefonu, kde může fungovat offline a vždy, když se klient připojit na internetovou síť,

aplikace kontroluje zda není na serveru novější verze databáze a pokud je, zkopíruje pouze změny, které proběhly od posledního stahování. (Jako příklad z geoinformatického prostředí bych uvedla diplomovou práci Dalibora Janáka, který řeší replikaci databáze lezeckých cest do mobilní aplikace.)

Databázové systémy nabízí širokou škálu nastavitelnosti, která umožňuje přizpůsobit replikaci danému řešení.

1 CÍLE PRÁCE

Cílem diplomové práce je provést rešerši v oblasti dostupných replikačních řešení a na jejím základě prakticky otestovat proces synchronizace a replikace geodat s ohledem na možnosti kombinace s produkty ArcGIS. V rešerši budou diskutovány databázové servery SQL Server a PostgreSQL, oba podporované produkty ArcGIS, a na jejím základě pak bude vybrán jeden, na kterém bude proces replikace prakticky testován.

V teoretické části práce budou podrobně definovány pojmy týkající se zálohování dat, především však synchronizace a replikace, dále detailně rozebrána replikace ve všech možných variantách nastavení, tedy jednosměrná, dvousměrná, synchronní, asynchronní, kaskádová, logická a fyzická. Dále rozbor zahrne celé portfolio produktů od desktopového řešení, přes možnosti ArcGIS serveru až po cloudový ArcGIS online.

Praktická část se bude zabývat návrhem replikačního řešení zohledňujícího požadavky katedry na databázové řešení a bude brát v úvahu její možnosti a způsob využívání databáze. Bude připraveno testovací prostředí na základě vytvořeného návrhu, které bude následně prakticky vyzkoušeno. Bude porozováno, zda replikace probíhá bez chyb a jsou přenesena všechna data v relativně krátkém časovém horizontu. Na závěr budou sepsána doporučení pro efektivní využívání databázového řešení.

2 POUŽITÉ METODY A POSTUPY PRÁCE

3 TEORETICKÁ VÝCHODISKA

Databáze je strukturovaná kolekce dat, která slouží pro efektivní ukládání dat a jejich zpětně čtení (Oppel, 2009). V relační databázi jsou data ukládána ve formě tabulek, tedy entit a atributů, a jsou vzájemně propojeny logickými vazbami, které se nazývají *relace* (Connolly, 2005). Toto logické uložení vazeb mezi tabulkami umožňuje efektivní manipulaci s daty, rychlé vyhledávání i komplexní analýzu (Momjian, 2001).

Základy *relační databáze* položil v roce 1970 matematik E. F. Codd, který relačnímu modelu přidal i srozumitelné příkazy vycházejících z běžné angličtiny, které jsou dnes známy jako jazyk *SQL* (Structured Query Language)(Žák, 2001). V dnešní době je možné setkat se také s pojmy objektová a objektově-relační databáze, které přebírají řadu vlastností z oblasti objektového programování.

Obvykle se rozlišují pojmy databáze, který odkazuje na obecný koncept, a pojem databázový systém nebo přesněji *systém řízení báze dat* [1], což je konkrétním počítačovým program, který zajišťuje fyzické uložení dat. Moderní SŘBD jsou navrženy na principu klient/server, kdy databáze běží jako služba na pozadí a čeká na dotazy od klientů. Server uživatelům umožňuje skrze jazyk SQL přístupovat k databázi, vytvářet a aktualizovat data, stejně jak jako vyhledávat či analýzovat (Connolly, 2005).

Pro uložení dat malého projektu je samozřejmě možno použít i jiného formátu určeného pro ukládání dat, například soubory formátu XLS, XML, CSV či moderního JSON. Pro komplexní správu dat velkého projektu je však databáze pro své relace více než vhodná.

Prostorová databáze, někdy také zvaná *geodatabáze*, není nic jiného než databáze obohacená o datový typ určený pro ukládání prostorové informace o prvku, prostorové indexy a sadu funkcí vhodných pro správu prostorových dat. Více informací o prostorových databázích viz kapitola 3.4.1 PostgreSQL 9.x (+ PostGIS) a 3.4.2 Microsoft SQL Server.

Prostorová data, také zvaná *geodata*, jsou z pohledu společnosti Esri prvky, které nesou informaci o geografické poloze, zakódovanou informaci o tvaru (bod, line, polygon) a popis geografického jevu. Tato geodata jsou uložená ve formátu, který je možno použít v geografickém informačním systému (Esri, 2006). Příkladem takového formátu může být rastrový Erdas Image, Esri grid, GeoTIFF, PNG, JPEG2000 nebo vektorový Esri shapefile, Esri coverage, GML, KML, DFX, DGN, GeoJSON nebo GeoHash.

Dalšími formáty, používanými právě pro ukládání dat do databáze, jsou Well-Known Binary (WKB) a Well-Known Text (WKT) pro reprezentaci vektorových dat.

[1] angl. Database Management System (DBMS)

Jejich tvar je dán standardem OGC[2] *Simply Features for SQL 1.2.1*, který specifikuje model pro uložení prostorových dat v digitální podobě.

Dle standardu OGC jsou funkce pro získání geometrie z databáze:

- `ST_AsBinary(geometry)` pro bitový zápis WKB a
- `ST_AsText(geometry)` pro WKT text.

Simply Features je založen na 2D geometrii s možností lineární interpolace mezi lomovými body. Základní prvky[3], které je možno vkládat ve formátu druh prvku a v závorce souřadnice lomových bodů, jsou:

- bod - `POINT(0 0)`,
- linie - `LINESTRING(0 0, 1 1, 1 2)`,
- polygon - `POLYGON ((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1))`,
- série bodů - `MULTIPOINT((0 0),(1 2))`,
- série linií - `MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))`,
- série polygonů - `MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)), ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1)))` a
- geometrická kolekce, která může obsahovat geoprvky různých typů (body, linie, polygony) - `GEOMETRYCOLLECTION(POINT(2 3),LINESTRING(2 3,3 4))`.

3.1 Vymezení pojmů

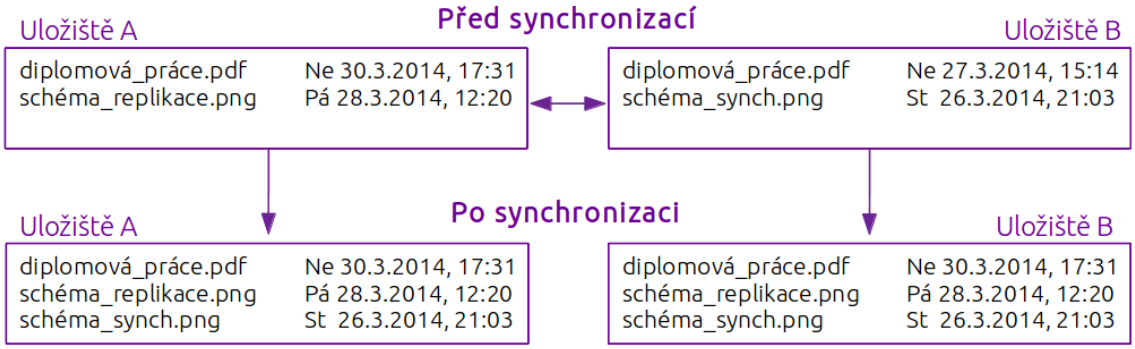
Pro lepší porozumění textu této práce je potřeba definovat pojmy replikace, synchronizace a verzování včetně popisu toho, jak jsou dané pojmy chápány v produktech ArcGIS. Výše zmíněné procesy jsou v literatuře často popisovány velmi odlišně. Některé zdroje pojmy replikace a synchronizace rozlišují, jiné je naopak považují za synonyma. Všechny zmíněné pojmy souvisí se zálohováním dat, tedy kopírovaním dat mezi dvěmi a více uložišti a se liší konkrétním důvodem pro použití daného procesu.

O synchronizaci souborů či datových složek je možno mluvit v případě, že existují dva datové zdroje, které je potřeba v daný okamžik sjednotit. Jde tedy o o proces, který probíhá jednorázově na vyžádání uživatele. U souborů se shodným názvem se

[2]OGC standardy jsou kontrolované konsorciem Open Geospatial Consortium, zdroj `http://www.opengeospatial.org/ogc`

[3]zdroj `http://postgis.net/docs/manual-2.1/using_postgis_dbmanagement.html#RefObject`

porovnává čas posledního zápisu, velikost nebo obsah souboru, naopak soubory, u kterých nebyla nalezena shoda, jsou jednoduše zkopírovány. Důvodem pro synchronizaci je většinou porování dvou a více datových uložišť a potřeba dostat je totožného stavu. To může například přispět snazší spolupráci více uživatelů nad stejnými daty nebo pomoct uživateli, který pracuje na více počítačích.



Obrázek 1: Příklad obousměrné synchronizace dat mezi dvěmi datovými uložišti

Replikace je průběžný průběžný, který soustavně hlídá, zda ve zdrojových datech nedošlo ke změny a pokud ano, dané změny zkopíruje na jiná datové uložiště. Často je tento proces používán právě ve spojitosti s databázemi, kdy jsou data kopírována z důvodu snížení zátěže serveru, či zvýšení ochrany dat. Replikace je tedy často vyžadována z jiných důvodů než synchronizace, začíná s daty existujícími pouze na jednom uložišti a pro zajištění konzistence dat používá jiných technologií. Více se replikací zabývá kapitola 3.2 Replikace.

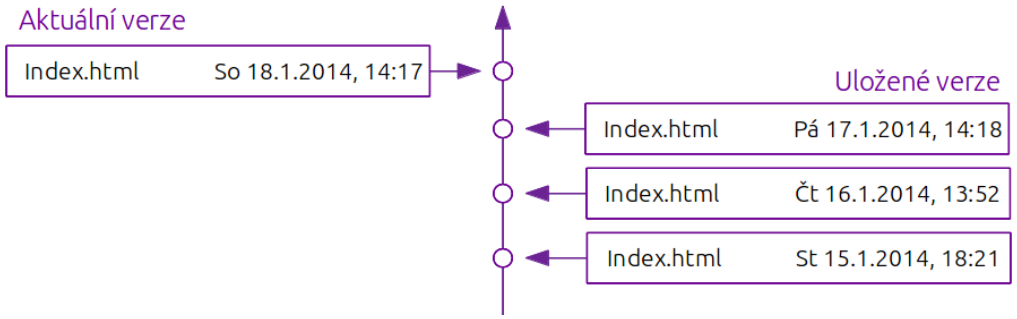
Oba procesy je možno použít jednostranně, tedy kopírovat data pouze z jednoho uložiště na druhé a nikoliv opačně, nebo oboustraně, kdy se data kopírují navzájem mezi sebou.

Specifickým způsobem zálohy dat je verzování, kdy se data na záložním datovém uložišti nepřepisují, ale systematicky ukládající v takzvaných verzích tak, aby se uživatel mohl snadno kdykoliv vrátit k předchozím stavům souborů. Smyslem verzování je zachovat všechny zvolené stavy práce, čímž se verzování liší od zálohování, kde stačí mít aktuální kopii daných dat. To, co je zde popsáno jako verzování, se v produktech ArcGIS nazývá archivování dat (Law, 2008).

Verzování může probíhat ručně, poloautomatizovaně či plně automatizovaně díky speciálním nástrojům pro správu verzí, kterých je na internetu dostupná celá řada. Oblíbeným verzovacím systémem programátorů je Git[4], open-source nástroj pro správu verzí, který pomáhá při práci s malými i velkými projekty a podporuje tý-

[4]více na `http://git-scm.com/`

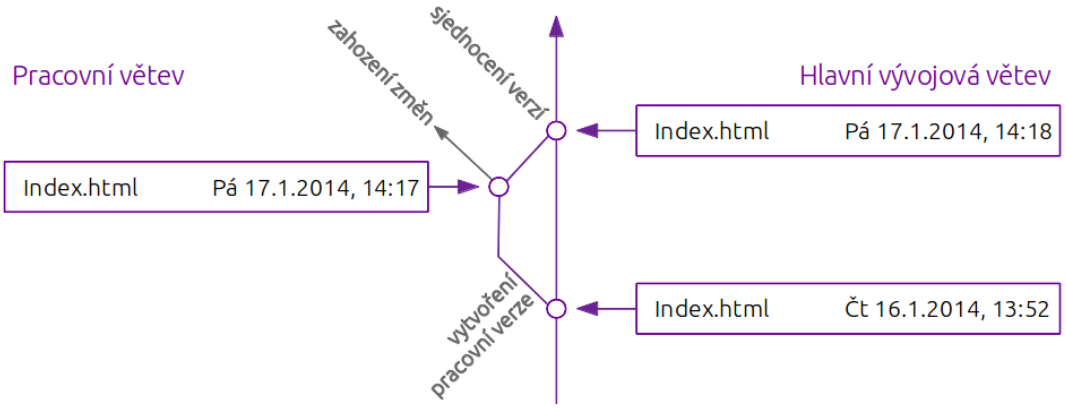
movou spolupráci. Umožňuje vrátit jednotlivé soubory nebo celý projekt do předchozího stavu, porovnávat změny provedené v průběhu času, zjistit, kdo naposledy upravil něco, co nyní možná způsobuje problémy, kdo vložil jakou verzi a mnoho dalšího (Chacon, 2009). Git je vhodný zejména pro textové soubory, protože dokáže analyzovat části textu, či programového kódu a zvýraznit místa, která se změnila.



Obrázek 2: Příklad verzování souboru

Samotná databáze přímo neposkytuje verzování dat. Nejsnazší způsob, jak získat verzi dat, je export databáze do souboru. Takový soubor je poté možno verzovat podobným způsobem jako jakýkoliv jiný soubor. Totéž platí i pokud v databázi ukládáme geodata. Podobně jako pro textová data, byl pro prostorová data vytvořen verzovací systém GeoGIT, který vychází z již zmiňovaného systému Git. Umožňuje uživatelům uchovávat změny v souborech shapefile, SpatialLite a z databáze PostGIS (PostgreSQL) a stejně tak jako Git se vrátit ke kterékoli předchozích verzí nebo sledovat konkrétní změny, které v datech nastaly od posledního záznamu.

Verzování může být chápáno také jako vytvoření pracovní verze. V případě, že jdou data bezchybná, ale je potřeba je aktualizovat, testovat či jinak měnit, pak je vhodné vytvořit tzn. pracovní verzi, aby nedošlo k jejich poškození. Jedná se tedy o kopii aktuálního stavu, na které je možno pracovat a zkoušet. V případě, že práce nedopadne podle představ, je možno změny zahodit, pokud je tomu naopak, je možno pracovní verzi sjednotit s platnou verzí. Tento způsob verzování umožňuje Git i GeoGIT a takto chápe pojem verzování i společnost Esri. V případě databáze takto používáný způsob verzování navíc umožňuje více uživatelům editovat jednu databázi. V případě, že by si nevytvořili pracovaní verzi, ale pracovali přímo s daty uloženými v databázi, databázový systém by data zamknul a žádný jiný uživatel by je nesměl editovat. Verzování v produktech ArcGIS zajišťuje technologie ArcSDE, která bude více diskutována v kapitole 3.4.3.



Obrázek 3: Příklad verzování souboru s použitím pracovní větve

3.2 Replikace

Replikace je proces, u kterého jsou data a databázové objekty kopírovány z jednoho databázového serveru na druhý a poté synchronizovány pro zachování souladu obou databází. Synchronizací v tomto případě myslíme kopírování všech změn, které v databázi nastanou. Použitím databáze je možno data distribuovat na různě vzdálená místa nebo mezi mobilní uživatele v rámci počítačové sítě a internetu (Microsoft, 2013).

Mnohé moderní aplikace se musí zabývat velkým počtem současných přístupů do databáze, což může v některých případech způsobovat problémy. Buď je server přetížen počtem připojení a data tedy přicházejí k uživateli pomalu, nebo dokonce úplně vypadne.

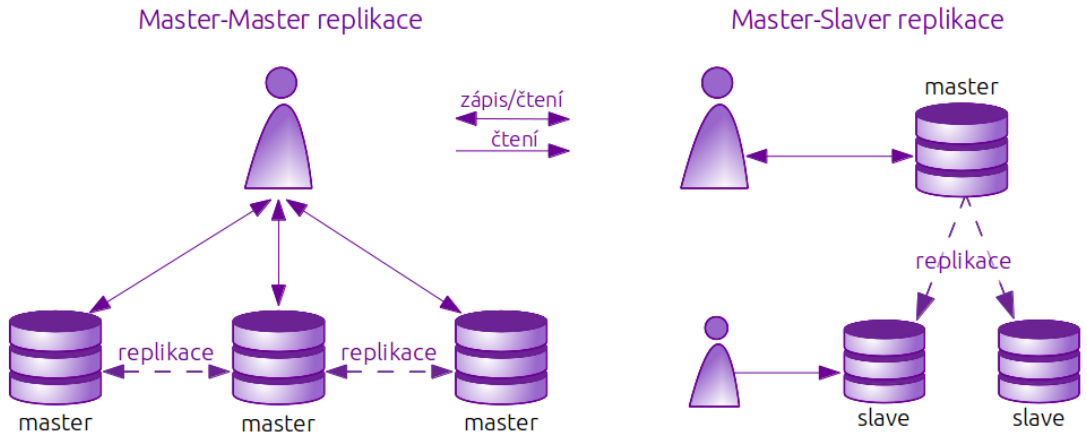
Mezi časté důvody použití databázové replikace tedy patří zajištění dostupnosti dat[5], resp. snížení pravděpodobnosti, že data nebudou dostupná, což může být způsobeno již zmíněným výpadkem serveru nebo například fyzickou ztrátou dat (Obe a Hsu, 2012). Další důvodem je rozložení zátěže přístupů do databáze mezi více serverů, takže nebude docházet ke zpomalení výkonu hlavního serveru ani k situaci, že data nebudou dostupná kvůli jeho výpadku (Bell et al., 2010). Databáze je často zálohovaná, například skriptem dump a i to může server zpomalit. Vhodným řešením je tedy nejdříve vytvořit kopii dat na jiný datový server a až poté proces zálohování spustit.

Všechny databáze zapojené do procesu replikace jsou v odborné literatuře nazývané uzly, angl. node. Tyto uzly dohromady tvoří replikační cluster[6]. Při správně nastavené replikaci, jejímž cílem je zajištění vysoké dostupnosti dat (HA), by v clus-

[5]angl. High Availability
[6]volně přeloženo jako skupina serveru zapojených do replikace

teru nikdy neměly být méně než tři uzly. Může se totiž stát, že vypadne jeden ze dvou uzlů, čímž dojde, ikdyž jen na krátkou chvíli, k situaci, že data nebudou v daný okamžik zálohovaná.

Uzly v replikačním clusteru mohou mít jednu ze dvou základních rolí, nejčastěji nazývaných master a slave. Master server nebo pouze master je server, který poskytuje data k replikaci, má práva na čtení i zápis a probíhají tedy na něm veškeré aktualizace. Je možno se setkat také s pojmenováním Primary server, Provider, Sender, Parent nebo Source server. Naprosto jiný pojem zavádí SQL Server, který tento zdrojový server nazývá Publisher (česky Vydavatel). Druhý databázový server je nejčastěji nazýván slave, Standby, Reciever, Child nebo Subsciber (česky Odběratel). Poslední pojem je také používán SQL Serverem. Na tento server, který je dostupný vždy jen pro čtení dat, se data a aktualizace kopírují, není však možné na něj změny zapisovat (Riggs a Krosing, 2010).

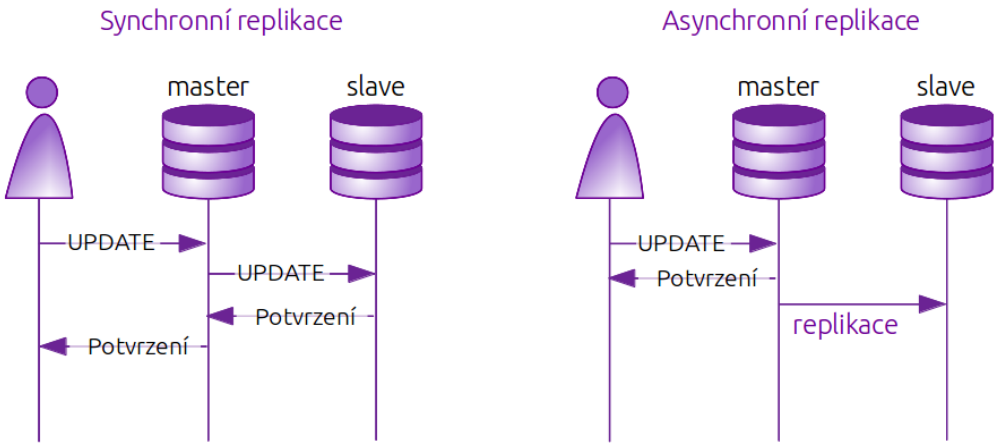


Obrázek 4: Srovnání Master-Master a Master-Slave replikace

Podle počtu master a slave serverů v replikačním clusteru, se rozlišuje zda se jedná o jednosměrnou nebo obousměrnou replikaci. Tzv. master-master replikace umožňuje zapisovat do všech uzlů v replikačním clusteru, což může být praktické například při použití databáze offline (viz obr. 4). Změny se tedy synchronizují mezi všemi databázovými uzly. Tento způsob však nese značné komplikace, je potřeba řešit konflikty změn ve stejných datech a je relativně náročný na údržbu. Tato práce se zabývá použitím druhé způsobu, tzv master-slave replikace. Tato replikace používá vždy jen jeden master server v clusteru a dva a více slave servery. Kopie dat tedy probíhá jednosměrně, vždy z master na slave servery. Podle Bella a kol. (2010) mají moderní aplikace často více čtenářů než zapisovatelů, proto je zbytečné, aby se všichni čtenáři připojovali na stejnou databázi jako zapisovatelé a zpomalovali tím jejich práci (Bell et al., 2010). Z toho důvodu je tedy použití master-slave replikace více než vhodné.

Při návrhu replikace je potřeba se zamyslet také nad tím, zda bude synchronní či asynchronní. Synchronní replikace neumožní potvrzení transakce modifikující data, dokud všechny změny nejsou přeneseny na slave server (Böszörmenyi a Schönig, 2013). Tento přístup zajistí, že žádná data nebudou v průběhu transakce ztracena. V některých případech tento způsob může zbytečně zpomalit rychlost přístupu do databáze, protože je nutno čekat na každou nedokončenou transakci. Zároveň může způsobit snížení dostupnosti databáze, protože v případě, že se například přeruší spojení mezi servery, nemůže být na master serveru potvrzena žádná další transakce. Ale jistě si najde své opodstatění například při bankovních transakcích, kde je potřeba, aby všechny operace proběhly na obou stranách. V tomto případě je užití tohoto způsobu zcela nezbytné.

Druhým způsobem je asynchronní replikace, při které se nová data mohou zapisovat na master server, přestože ještě nedošlo k replikaci stávajících dat na slave server (Obe a Hsu, 2012). To je sice za běžného provozu rychlejší, v některý případech však může způsobit nekonzistenci dat, například když proběhne transakce na master serveru, který však spadne dřív, než se změna zapíše na slave. V takovém případě se slave změní na master server, ale zároveň se nikdy nedozví o transakci, o které má uživatel informace, že proběhla v pořádku.



Obrázek 5: Rozdíl mezi synchronní a asynchronní replikací

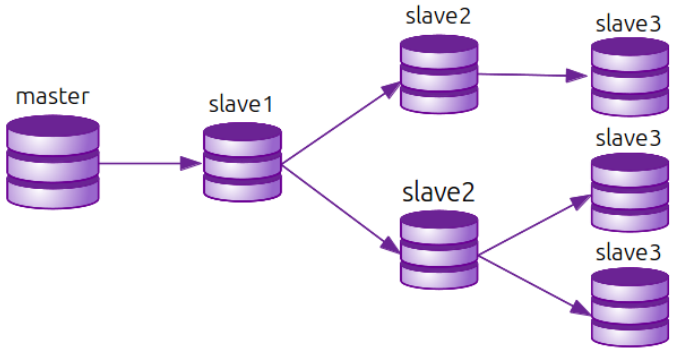
Replikace v PostgreSQL umožňuje plnou kopii dat z databáze i pouze výběr některých tabulek. Více o možnostech a způsobech nastavení replikace v kapitolách 4.4 Příprava prostředí pro konfiguraci a 4.5 Konfigurace replikace.

Dále je možno rozlišovat replikaci pole toho, zda je logická nebo fyzická. Fyzická replikace na druhý server kopíruje bloky binárních datových souborů bez znalosti jejich struktury (sloupce, řádky, ...), čímž se zajistí identická replika. Pro tento způsob kopírování dat, která mají jasně danou strukturu, je potřeba mít na obou

serveru stejnou platformu a architekturu. Tento způsob je velice spolehlivý a často snazší na konfiguraci.

Naopak logická přenáší data v textové podobě, která nese informace o příkazu a struktuře. Tento způsob je více flexibilní, umožňuje výběr jen několika databází nebo tabulek a není závislý na architektuře ani operačním systému (Böszörmenyi a Schönig, 2013).

Posledním diskutovaným pojmem je kaskádová replikace, která umožňuje připojit repliku k jinému slave serveru místo k hlavnímu master serveru. Tento způsob může být výhodných především z těchto dvou důvodů. Řekněme, že se kaskádová replikace používá při existenci většího počtu slave serverů v clusteru, třeba sta. V případě, že by se všechny repliky připojovaly k hlavnímu serveru, došlo by u něj k razantnímu zpomalení jeho výkonu. Kaskádová replikace může být praktická také v okamžiku, kdy se data přenáší na velkou vzdálenost, třeba do Číny. V případě, že mají v Číně dvě repliky, je zcela zbytečné, aby se obě kopie přenášely na tak velkou vzdálenost, když druhá replika se může připojit k první a mít data s mnohem menším zpožděním.



Obrázek 6: Ukázka kaskádové replikace

Každý databázový server (myšleno SŘDB) si volí terminologii a konkrétní nastavení mírně odlišně. Tato kapitola se snaží popsat chápání replikace co v největší míře obecně s ohledem na použití tohoto pojmu v PostgreSQL. Zcela jinou terminologii, ikdyž založenou na stejných principech, zavádí SQL Server, který pro export databáze do souboru používá pojem snímková replikace, pro master-slave replikaci pojem transakční replikace a pro master-master replikaci slučovací replikace.

3.3 ArcGIS produkty

V názvu práce se objevuje spojení Esri platforma, čímž jsou chápány produkty americké společnosti Esri založené v roce 1969 manželi Dangermondovými, a zabývající

se vývojem software zaměřeného na geografické informační systémy [7].

Z hlediska chápání Esri má GIS tři roviny. První je to GIS jako prostorová databáze reprezentující geografické informace, dále sada map zobrazující prvky a vztahy mezi prvky na zemském povrchu a zároveň i software pro GIS jako sada nástrojů pro odvozování nových informací ze stávajících. Esri tyto tři pohledy na GIS propojuje v software ArcGIS jakožto kompletní GIS, který se skládá z katalogu (kolekce geografický datových sad), map a sad nástrojů pro geografické analýzy.

Esri vytváří integrovanou sadu softwarových produktů ArcGIS, které poskytují nástroje na kompletní správu GIS a přizpůsobují produkty různým úrovním nasazení. Výběr produktu záleží na tom, zda zákazník požaduje jedno nebo více uživatelský systém, zda se má jednat o stolní systém nebo server, popř. zda má být dostupný prostřednictvím internetu. Nabízí také produkty vhodné pro práci v terénu (Esri, 2006).

Základními produkty[8] jsou stolní systémy ArcGIS for Desktop ve verzích Basic, Standard, Advanced[9], dále serverové verze ArcGIS for Server (pro Linux a Windows) ve třech úrovních funkcionality (Basic, Standard, Advanced) a dvou úrovních kapacity serveru (Workgroup a Enterpise). Další produkt ArcGIS for Mobile, ve verzích ArcPad, ArcGIS for Windows Mobile a ArcGIS for Smartphone and Tablet, je určený především pro práci v terénu. A v neposlední řadě verze dostupná skrze internet ArcGIS Online. K tomu všemu Esri přidává velké množství extenzí a dalších verzí[10].

Tabulka 1: Varianty programu ArcGIS platné od verze 10.1.

Produkt	**Verze**		
ArcGIS for Desktop	Basic	Standard	Advanced
ArcGIS for Server	Basic	Standard	Advanced
ArcGIS for Mobile	ArcGIS for Windows Mobile	ArcPAD	ArcGIS for Smartphone and Tablet
ArcGIS Online			

Dle Law (2008) je nativním formátem produktů ArcGIS geodatabáze a jsou rozlišovány tři druhy geodatabáze. Ani v jednom případě se však nejedná o databázi v pravém slova smyslu, tak jako ji chápame v **??** a 3.4.2. V každém případě však tyto

[7] více informací `http://www.esri.com/about-esri/history`

[8] Názvy jednotlivých produktů použitých v tomto odstavci jsou platné od verze ArcGIS 10.1. Starší verze ArcGIS používají jiné názvy, jejichž přehled je možný na stránkách firmy ARCDATA Praha `http://www.arcdata.cz/produkty-a-sluzby/software/arcgis/prejmenovani-arcgis/.`

[9] zdroj `http://www.esri.com/software/arcgis/about/gis-for-me`

[10] kompletní seznam na oficiálních webových stránkách Esri `http://www.esri.com/products` nebo `http://www.arcdata.cz/produkty-a-sluzby/software/arcgis/`

způsoby umožňují uložení, přístup a správu dat. U prvních dvou typů, personální a souborové geodatabáze, se data ukládají do jednoho binárního souboru, kde jsou však ukládaná ve stejné struktuře jako v plnohodnotném databázovém serveru. Do takového geodatabáze můžeme uložit více než jednu vrstvu, což je výrazný rozdíl oproti formátu shapefile. Výhodou je dále možnost použití relací, sofistikované dotazování a v neposlední řadě i snadná přenostitelnost, protože takováto databáze bude vždy jen jeden soubor obsahující několik vrstev. Oproti tomu shapefile, který obsahuje jen jednu vrstvu, je tvořen minimálně 4 soubory. Oba tyto typy podporují pouze jednoho editujícího uživatele a mnoho uživetelů s právem čtení. Nepodporují dlouhé transakce ani verzování.

Tabulka 2: Přehled rozdílů personální a souborové geodatabáze v ArcGIS

databáze	**souborová .gdb**[1]	**personální .mdb**[1]
datové uložiště/ databázový server	lokální souborový systém	MS Access
licence	ArcGIS for Destop (všechny verze)	ArcGIS for Destop (všechny verze)
operační systém	Windows (možná i jiné)	Windows
požaduje ArcSDE	ne	ne
vlastní datový typ	ne	ne
víceuživatelská editace	ano, ale s limity	ne
počet editorů	1 pro každý dataset nebo tabulku[2]	1[2]
počet čtenářů	více než 1[2]	více než 1[2]
master server[3]	ne[1]	ne[1]
slave server[3]	ano	ano

[1] http://www.esri.com/software/arcgis/geodatabase/singlex-user-geodatabase

[2] http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#//003n00000007000000

[3] je možno použít jako master/slave server

Tato práce se více zaměřuje na třetí typ, technologii ArcSDE, kterou v některých materiálech nazývají *geodatabáze ArcSDE*. Nejedná se o geodatabázi, ale spíše o zprostředkovatele komunikace mezi programem ArcGIS a databázovým server. Umožňuje víceuživatelský přístup, verzování i replikaci (Esri, 2006). Tato technologie využívá jako datové uložiště některý z již existujících databázových serverů, např. níže popsané PostgreSQL nebo SQL server. Touto technologií se více bude zabývat kapitola 3.4.3 ArcSDE geodatabase.

3.4 Vybrané programové prostředky

3.4.1 PostgreSQL (+ PostGIS)

PostgreSQL je objektově-relační databázový systém s otevřeným zdrojovým kódem dostupný na většině platforem. Je volně k dispozici pro použití, modifikaci a šíření způsobem, který si sami zvolíme. Jedná se o robustní, výkonný, bezpečný, kompatibilní a interoperabilní software s podporou a dobře komentovaným zdrojovým kódem. Vyhovuje standardům SQL od verze SQL 2008 a nabízí velké množství pokročilých funkcí. PostgreSQL je založen na architektuře klient-server, to znamená, že server pořád běží a čeká na dotazy klienta (Momjian, 2001).

S vývojem databázového serveru PostgreSQL začala University of California v Berkley již více než před 20 lety. Nyní je vyvíjen a udržován velkou komunitou nezávislých vývojářů. Používá licenci TPL (The PostgreSQL Licence), která je mírně odlišná od open-source licence BSD (Berkeley Distribution Software), ze které vychází (Riggs a Krosing, 2010).

Řadí se mezi nejpokročilejší databáze díky schopnosti pracovat s velkými objemy dat, díky své rychlosti a funkcionalitě může soupeřit i s populárními komerčními systémy jako je Oracle, IBM DB2, Microsoft SQL Server 2008 a dalšími (PostgreSQL, 2012).

Samotné PostgreSQL neobsahuje datové typy ani funkce vhodné pro správu prostorových dat. K tomu je nutné přidat nástavbu PostGIS, která implementuje specifikaci Simple Features for SQL konsorcia OGC a rozšiřuje tak databázi PostgreSQL o podporu geografických dat. PostGIS umožňuje ukládání geometrických objektů (bod, linie, polygon), použití prostorových funkcí pro určení vzdáleností, délky linií, výměr a obvodu ploch, výběr indexu při spojení prostorových a atributových dotazů a mnoho dalších.

PostGIS umožňuje práci s rozšířenými XML formáty GML, KML, GeoJSON a SVG, jejichž funkce pro získání geometrie jsou:

- `ST_AsGML(geometry)`,
- `ST_AsKML(geometry)`,
- `ST_AsGeoJSON(geometry)` a
- `ST_AsSVG(geometry)`.

PostGIS používá dva základní prostorové datové typy *geography* a *geometry*. Typ geography ukládá souřadnice v kartézských rovinných souřadnicích, kterým odpovídá

souřadnicový systém WGS84. Je zejména vhodný pro malá území, protože při výpočtu vzdálenosti dvou bodů tento datový typ vrátí jako výsledek nejkratší vzdálenost v kilometrech v rovině. Typ geometry data ukládá v polárním rovinném systému a umožňuje nastavit souřadnicový systém dle potřeb. Výsledkem dotazu na vzdálenost dvou bodů tedy bude úhel ve stupních, které po převodu do metrické soustavy určí nejkratší vzdálenost na kouli. Při výběru datového typu může být rozhodující například počet funkcí, kterých typ geometry poskytuje mnohem více než geography, nebo velikosti daného území (OpenGeo, 2012b).

Existuje také další nástavba PostGIS Raster, která rozšiřuje ukládání a manipulaci s rastrovými daty, nástavba PostGIS Topology pro topologickou správu vektorových dat a pgRouting pro síťové analýzy. PostGIS je podporován velkou řadou softwarových produktů zabývajících se správou geografických dat, což také umožňuje snadnou přenositelnost a použitelnost jednotlivých nástaveb (příklad software podporujících PostGIS: QGIS, GvSIG, GRASS, ArcGIS).

PostGIS implementuje mnoho běžně používaných knihoven jako GEOS (Geometry Engine Open Source) pro implementaci jednoduchých prostorových prvků a metod pro topologii, PROJ4 pro převod mezi kartografickými projekcemi nebo GDAL/OGR (Geospatial Data Abstraction Library) pro převod mezi různými vektorovými i rastrovými formáty (Obe a Hsu, 2011). Nástavba PostGIS 1.5. obsahovala přes 800 funkcí, typů a prostorových indexů (Obe a Hsu, 2012). Aktuální verze PostGIS[11] je 2.1.

Tabulka 3: Možné kombinace verzí PostgreSQL (+ PostGIS) a ArcGIS

PostgreSQL	PostGIS	ArcGIS	podporovaná architektura
9.3	PostgreSQL 9.3 není zatím podporováno produkty ArcGIS		
9.1 (64-bit)	2.0 (64-bit)	10.1 SP1	Linux 64-bit (x86_64), Windows 64-bit
9.0 (64-bit)	1.5* (64-bit)	10.1 SP1	Linux 64-bit (x86_64), Windows 64-bit
9.0 (64-bit)	1.5* (64-bit)	10.1	Linux 64-bit (x86_64), Windows 64-bit
8.3/8.4	1.4	10.0	Linux 64-bit (x86_64), Windows 64-bit

*není podporováno ve verzi Windows 64-bit

zdroj: http://support.esri.com/en/knowledgebase/techarticles/detail/40553

Od verze ArcGIS 9.3. je PostgreSQL oficiálně podporovanou databází pro ukládání geodat v produktech ArcGIS. Při instalaci je pouze potřeba zajistit kompatibilitu verzí jednotlivých nástrojů, viz tab. 3. Pro verzi ArcGIS 10.1 jsou podporované verze PostgreSQL 9.0 a PostGIS 1.5., pro ArcGIS 10.1 SP1[12] lze použít novější PostgreSQL 9.1

[11] aktuálně na http://postgis.refractions.net/

[12] Service Pack 1

a PostGIS 2.0 (OSGeo, 2013)[13]. Na stránkách ArcGIS Resources[14] jsou dále popsána další doporučení, například že je podporovaná pouze 64-bitová verze PostgreSQL.

Databázi PostgreSQL lze v ArcGIS produktech použít dvojím způsobem. Buď jen jako uložiště dat bez přidání geografického datového typu, nebo včetně datového typu, tedy včetně PostGIS knihovny. ArcSDE podporuje pouze datový typ PostGIS Geometry a přidává vlastní datový typ Esri St_Geometry. Výhodou používání Esri St_Geometry je nezávislost na zvoleném databázovém systému, tedy snazší přenostitelnost celého řešení.

3.4.2 Microsoft SQL Server

Microsoft SQL Server (dále SQL Server) je relační databázový systém vyvíjený společností Microsoft dostupný pro různé verze operačního systému Windows. Dodává se v mnoha verzích, které lze nainstalovat na různé hadrwarové platformy na základě odlišných licenčních modelů (Whalen, 2008). Podle Leitera (2009) SQL Server nabízí 8 základních verzí: Enterprise, Standard, Workgroup, Web, Express, Express Advanced Edition, Developer Edition a Compact Edition. Enterprise edition podporuje naprosto vše, co SQL Server nabízí, naopak verze Express, která je dostupná zdarma, obsahuje omezení některých funkcí a proto je vhodná spíše pro malé nebo začínající projekty (Leiter, 2009).

Prostorová data jsou implementována jako CLR rozšíření a přidávají databázovému serveru dva prostorové datové typy geometry a geography, jejichž rozdíl je podobný jako u PostgreSQL. První jmenovaný slouží k reprezentaci dat (bodů, linií, polygonů) v rovině, naproti tomu datový typ geography slouží ukládání stejných dat na povrchu zeměkoule. Oba typy pracují ve dvou dimenzích, nebere se tedy v potaz výška. Podporuje také indexování dat, index je tvořen standardním B stromem (Činčura, 2009). SQL Server podporuje OGC standardy pro prostorová data.

SQL Server je podporován a používán ArcGIS produkty od začátku jeho vývoje[15]. Verze ArcGIS Enterprise může být propojena s jakoukoliv uživatelem zvolenou a zakoupenou licencí databázového systému. Verze ArcSDE Desktop a Workgroup používají verzi Express, která je dostupná zdarma a podporuje většinu základních funkcí. Replikaci plně podporuje verze Enterprise, ostatní verze ji podporují pouze s ome-

[13] zdroj a další informace na stránkách PostgreSQL `http://trac.osgeo.org/postgis/wiki/UsersWikiPostgisarcgis` nebo ArcGIS Resources `http://resources.arcgis.com/en/help/system-requirements/10.1/index.html#//015100000075000000`

[14] `http://resources.arcgis.com/en/help/system-requirements/10.1/index.html#//015100000075000000`

[15] pro přehled kompatibilních verzí ArcGIS a SQL Server viz `http://resources.arcgis.com/en/help/system-requirements/10.1/index.html#/Microsoft_SQL_Server_Database_Requirements/015100000070000000/`

zenými funkcemi. Avšak již zmiňovaná verze Express, která je podporávána ArcSDE Desktop a Workgroup, může být použita pouze jako slave server, tedy odběratel replikovaných dat, a do takovéto databáze připojené do replikačního clusteru tedy není možné zapisovat. Nemůže být tím, kdo poskytuje data k replikaci (Whalen, 2008). Stejně jako u PostgreSQL platí, že si uživatel může zvolit, zda použije datový typ, který je součastí ArcSDE, nebo ten, který je implementován do SQL Serveru.

3.4.3 ArcSDE geodatabase

ArcSDE je technologie firmy Esri pro správu geoprostorových dat uložených v relačních databázových systémem. Jedná se o otevřenou a interoperabilní technologii, která podporuje čtení a zápis mnoha standardů. Využívá jako své nativní datové struktury standard konsorcia OGC Simple Feature a prostorový typ ISO pro databázové systémy Oracle, IBM DB2 a Informix. Poskytuje vysoký výkon a je přizpůsobena velkému počtu uživatelů (Esri, 2006).

ArcSDE je prostředník pro komunikaci mezi klientem (např. ArcView) a SQL databází (př. PostgreSQL). Umožňuje přístup a správu dat v databázi, současnou editaci jedné databáze více uživateli, archivování dat, , dlouhé transakce, zajišťuje integritu a poskytuje vlastní prostorový datový typ (St_Geometry) (Law, 2008).

Vytváří vlastní databázové schéma, tedy databázi s jasně danou strukturou, která definuje jednotlivé funkce, datové typy a indexy a obsahuje data ve formě tabulky, například aktuální změny v databázi. To se používá v případě, že více uživatelů edituje jednu tabulku a k daným datům vytváří tzv. pracovní verzi, kterou po dokončení připojí ke stávajícím datům. Tyto verze jsou právě uchovávány ve schématu ArcSDE ve formě tabulek.

Technologie ArcSDE vyžaduje dvě úrovně: databázovou a aplikační, která se skládá z ArcObjects a ArcSDE. Databázová úroveň zajišťuje jednoduchý, formální model pro uložení a správu dat ve formě tabulek, definici typů atributů (datových typů), zpracování dotazů či víceuživatelské transakce (Law, 2008). ArcSDE podporuje databázové systémy IBM DB2, IMB Informix, Oracle, Microsoft SQL, PostgreSQL (Esri, 2013a).

Existují tři úrovně ArcSDE databáze: desktop (ArcSDE Desktop), skupinová (ArcSDE Workgroup) a podniková (ArcSDE Enterprise). Každá verze má jiné parametry a umožňuje různou úroveň editace (viz tab. 4).

Tabulka 4: Přehled verzí ArcSDE, jejich parametrů a možností

databáze	ArcSDE		
	Desktop[1]	Workgroup[1]	Enterprise[1]
databázový server	SQL Server Express	SQL Server Express	PostgreSQL, Oracle, SQL Server a další
licence	ArcGIS for Destop	ArcGIS for Server Workgroup	ArcGIS for Server Enterprise
operační systém	Windows	Windows	multiplatformní
požaduje ArcSDE	ano	ano	ano
vlastní datový typ	ne	ne	ano
víceuživatelská editace	ne	ano	ano
počet editorů	1	10	bez limitu
počet čtenářů	3	10	bez limitu
master server[2]	ne	ne	ano
slave server[2]	ano	ano	ano
verzování	ano	ano	ano
závislost na sítích	lokální síť	lokální síť, internet	lokální síť, internet
velikostní limity	10GB	10GB	záleží na velikosti serveru

[1] http://www.esri.com/software/arcgis/geodatabase/multi-user-geodatabase

[2] pozn. je-li možno použít jako master/slave server

Od verze ArcGIS 9.2 je ArcSDE Desktop spolu s databázovým systémem SQL Server Express součástí licence produktů ArcGIS for Desktop Standard a Advanced. Takovou databázi mohou současně používat 4 uživatelé, z toho jen jeden může databázi editovat, jsou však omezeni velikostí databáze.

Součastí licence ArcGIS for Server Workgroup je ArcSDE Workgroup, která se liší od verze Desktop především tím, že počet uživatelů, kteří mohou současně editovat nebo prohlížet databázi, je zvýšen na deset.

Nejvyšší úroveň, ArcSDE Enterprise, je možno získat s licencí ArcGIS for Server Enterprise, která uživatelům přináší nejméně omezení. Mohou si vybrat z několika komerčních i nekomerčních databázových systémů, počet uživatelů není omezen, stejně jako velikost databáze.

Replikaci a synchronizaci dat umožňují pouze ArcSDE Enterprise a Workgroup (Esri, 2013b). Jak už bylo zmíněno v předchozí kapitole 3.4.2 Microsoft SQL Server Express 2008, SQL Server Express je možný použít v replikačním clusteru pouze jako slave server. Vzhledem k tomu, že proces replikace je implementován přímo do ArcObjects a ArcSDE, nezáleží na konkrétním databázovém systému (Law, 2008).

3.5 Nástroje pro replikaci v PostgreSQL

PostgreSQL nabízí hned několik nástrojů pro řešení replikace. Je možno použít zabudovanou streaming replikaci, která je dostupná od verze PostgreSQL 9.0 nebo některou z extenzí, například Slony-I, pgpool, Londiste, Bucardo nebo Postgres-XC, pro jejichž srovnání viz tab. 5. Tato kapitola se dále bude zabývat a porovnávat nativní streaming replikace s extenzí Slony-I a pgpool.

Tabulka 5: Srovnání různých typů dostupných replikačních řešení

nástroje	typ	technika	M/M	M/S	sync	async
PostgreSQL 9.1*	fyzická	xlog	ne	ano	ano	ano
pgpool-II	logická	proxy	ano	ne	ano	ne
slony-I	logická	triggers	ne	ano	ne	ano
Londiste	logická	triggers	ne	ano	ne	ano
Bucardo	logická	triggers	ano	ano	ne	ano
Postgres-XC	cluster	-	ano	ne	ne	ano

*streaming replikace zdroj: Tomáš Vondra, 2011

3.5.1 Slony-I

Jak píší Böszörmenyi a Schönig (2013) je Slony-I jeden z nejrozšířenějších externích nástrojů pro replikaci pro PostgreSQL. Zároveň se také řadí mezi nejstarší, plně používán je v PostgreSQL již od verze 7.3. a je velmi dobře podporován i dalšími externími řešeními pro PostgreSQL, například programem PgAdmin3, který nabízí správu dat pomocí grafického rozhraní (Böszörmenyi a Schönig, 2013).

Jedná se o trigger-based replikaci, což znamená, že je ke každé exitující tabulce přidán trigger, který zajistí replikaci každé změny, která v databázi nastane. Z toho také vyplývá, že se jedná o logickou replikaci, kdy je možné replikovat pouze změny v datech, tzv. DML změny, tedy SQL příkazy INSERT a UPDATE, nikoli strukturu databáze, příkazy typu CREATE/DROP TABLE, ALTER TABLE. Každá změna struktury se tedy musí provést ručně, což se může jevit jako nevýhodné. Nese to ale i své klady, například možnost výběru pouze některých tabulek. Vytváří si totiž tzv. *replikační set*, do kterého se zapíší pouze ty tabulky, které je potřeba replikovat.

Další výhodou, a to zvlášť v porování se streaming replikací, je možnost replikace dat mezi různými verzemi PostgreSQL bez ohledu na platformu a architekturu. Naopak spíše za nevýhodu je považováno, že si vytváří ke každé tabulce vlastní schéma, do kterého se ukládají replikovaná data, což způsobuje redundanci dat.

Slony-I replikace je z principu asynchronní, zpoždění je v řádu jednotek či desítek

vteřin. Umožňuje Hot Standby mode, kdy je možno použít repliku na dotazy, i kaskádovou replikaci. Slony-I má vlastní konfigurační nástroj a samotná replikace funguje díky vlastnímu replikačnímu démonu, který běží stále, registruje změny a kopíruje je na slave servery.

3.5.2 Streaming replikace

Streaming replikace je nativní řešení do PostgreSQL implementované od verze 9.0. Jedná se o log-shipping replikaci, což znamená, že jsou na slave servery posílány transakční logy, v PostgreSQL nazývané WAL (Write Ahead Log). Do nich jsou změny nejdříve zaznamenávány přímým zápisem na disk a až poté potvzeny jako úspěšné. Tento způsob zajišťuje datům naprosté bezpečí, protože kdyby došlo k chybě a změny se nezapisovaly na disk, ale pouze do cache, mohlo by dojít k jejich ztrátě. Zároveň to zajišťuje kopii jak dat, tak i struktury databáze. Existuje pouze jeden transakční log pro jednu instalaci PostgreSQL, proto se replikují vždy všechny databáze a není možné výběru jen několika tabulek, tak jako u Slony-I (Böszörmenyi a Schönig, 2013). Protože replikace probíhá pomocí transakčního logu, je nutné použití stejné verze PostgreSQL, stejné platformy i architektury na všech uzlech replikačního clusteru.

Streaming replikace umožňuje jak synchronní, tak asynchronní replikaci, dále Hot standby mode i kaskádovou replikaci.

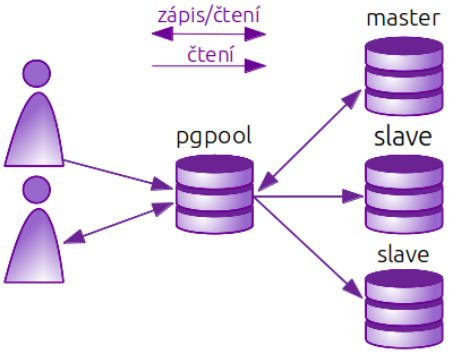
3.5.3 pgpool

Nástroj pgpool, který je stejně jako Slony-I extenzí pro PostgreSQL, je dalším z nástrojů, který je možno použít pro replikaci dat, umožňuje však i další pokročilé funkce jakými jsou sdílení spojení klienta s databází (angl. connection pooling), paralelní uložení dat (angl. parallel query) a rozložení zátěže mezi více servery (angl. load balancing) . pgpool umožňuje sdílení spojení klienta s databází, což v praxi znamená, že se vytvoří několik spojení se serverem, která i po skončení dotazu zůstanou otevřená a připravená pro další použití. Nemusí se tedy navazovat spojení při každém požadavku ze strany klienta, což velice zrychlí provoz a zajistí plynulost užívání databáze. Je vhodným nástrojem pro správu velkých tabulek díky distribuovanému způsobu ukládání dat (pgpool Global Development Group, 2013).

Zároveň je nástrojem pro zjednodušení nastavení a provozu replikace a umožňuje rozložení zátěže mezi více serverů v replikačním clusteru, aby nedocházelo k přetíže jednotlivých uzlů a celkově se zvýšila rychlost a efektivita práce s databází. V případě rozložení zátěže se pgpool stává prostředníkem pro komunikaci mezi klientem a serverem. Aby nebylo potřeba dát každému uživateli přístup k jinému slave serveru, nebo přístupy do databáze manuálně rozkládat skrze složité programové řešení, nabízí se

možnost použití pgpool, který se navenek jeví jako jakákoliv jiná databáze, do které se uživatelé připojí bez ohledu na jejich práva a požadavky. pgpool pak sám rozdělí dotazy mezi uzly v replikačním clusteru dle aktuální zátěže (Böszörmenyi a Schönig, 2013). Zároveň, pokud má uživatel přístup k zápisu i čtení, umí na základě jeho aktuálního SQL příkazu, rozhodnout, zda jej připojí k master nebo slave databázi, (viz obr. 7). Tento způsob velmi zjednoduší jak administraci databáze, tak nastavení pro běžného uživatele, který se připojí pouze do jedné databáze a víc se nezajímá, zda z databáze pouze čte, nebo do ní i zapisuje.

Na základě vybraných funkcí je možno použít jeden ze čtyř základních módů, které pgpool poskytuje[16]: základní, replikační, master/slave a paralelní. V návrh databázového řešení byl použit mód master/slave, který je dále popisován v kapitole 3.5.3.



Obrázek 7: Zjednodušené schéma pgpool v módu master/slave

[16]kompletní přehled na `http://www.pgpool.net/docs/latest/pgpool-en.html#config`

4 NÁVRH A KONFIGURACE REPLIKACE

Tato kapitola se zabývá hodnocením současného stavu správy dat na katedře geoinformatiky, návrhem databázového řešení dle požadavků a možností katedry a podrobně popisuje vytvoření testovacího prostředí na serverech katedry dle vytvořeného návrhu. Do hloubky popisuje konfiguraci vybraných nástrojů, včetně jejich praktického spuštění.

4.1 Aktuální stav správy dat

Katedra aktuálně provozuje tři servery, konkrétně virtus.upol.cz, gislib.upol.cz a geohydro.upol.cz. Poslední z jmenovaných byl poskytnut jako testovací server pro tuto práci a v budoucnu s ním počítá jako s master serverem pro zde popisované databázové řešení. První dva zmíněné servery jsou aktivně používány, hostují například geoportál publikovaný skrze ArcGIS Server, který je důležitým prostředkem pro prezentaci projektů a dat, která na katedře vznikají. Data ke geoportálu i dalším aplikacím běžícím na těchto serverech jsou ukládána do MS SQL Serveru, každý ze serverů momentálně obsahuje jiné datové sady, které nejsou pravidelně zálohovány, protože aktualizace dat není příliš častá. Aktuální řešení nepoužívá replikaci dat, což může způsobovat nedostupnost dat z důvodu výpadku serveru.

Databáze aktuálně obsahují data například z projektů BotanGIS[17], Virtuální studovna CHKO Litovelské Pomoraví[18], dále data metadatového systému Micka[19], data ze senzorové sítě KGI, data ke studentským pracím a také ukázková data určená pro výuku. Je založeno přibližně 10 účtů, které mají přístup pro zápis, a řádově v desítkách účtů s právem čtení. V současné situaci není do databází příliš často zapisováno.

Současný stav, kdy se přenášejí data přes různá hardwarová zařízení nebo kopírují po síti, není plně vyhovující z několika důvodů. Často jedná o velké objemy dat, jejichž kopie může trvat řádově až desítky minut. Studenti si musejí dělat kopie dat při každém cvičení, což velice zdržuje výuku. Data jsou poté fyzicky uložena na počítačích v učebnách, což mimo jiné dovoluje, aby se k datům například z různých projektů dostal kdokoliv, kdo má přístup na učebnu. Při každé aktualizaci dat je navíc potřeba data opět zkopírovat, což je další časové omezení, k tomu může dojít k nekonzistenci dat různých datových zdrojů.

[17] `http://botangis.upol.cz/botangis/mapa`
[18] http://virtus.upol.cz/
[19] `gislib.upol.cz/metadata`

4.2 Požadavky na databázové řešení

Katedra má zájem využít potenciál databázového řešení a plánuje využít tento návrh k uložení dalších datových sad, které má k dispozici a které jsou momentálně dostupné ve formátech shapefile nebo geodatabáze, ale zatím nejsou uložená v databázi, kterou je možno sdílet. Jedná se například o datové sady ArcČR500 verze 2.0 a 3.0, Data200 (ČUZK), CEDA ČR 150, data, která byla uvolněna pro podporu pro Krajinotvorný program MŽP, nebo data dostupná k produktům Esri nebo Idrisi. Data uložená v databázi pak budou mnohem snáze využitelná jak pracovníky, tak i studenty katedry, kteří data využijí nejen ve výuce, ale také v jejich odborných prácích. Při kopírováním dat na různá datová uložiště je navíc těžké udržet licenční podmínky, se kterými jsou data pořizována.

4.3 Návrh replikačního řešení

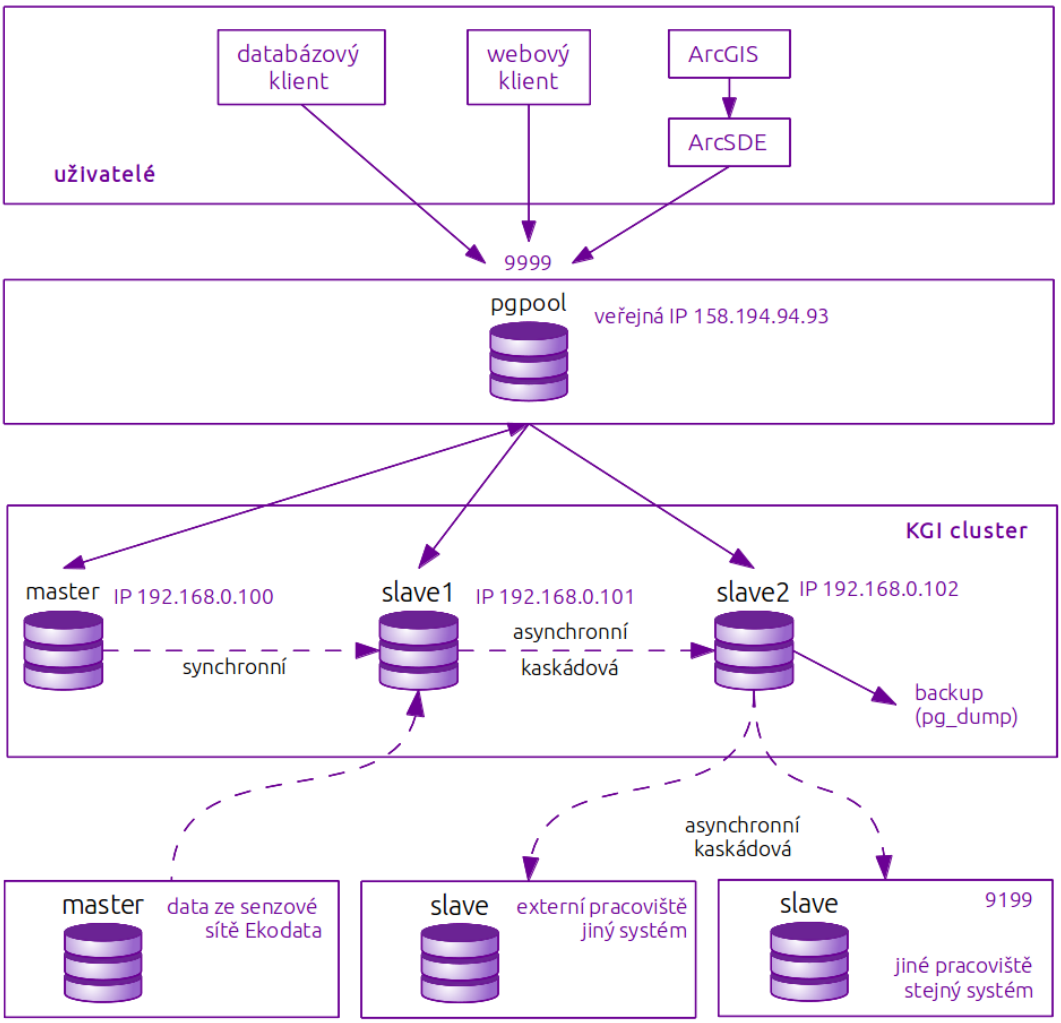
Po provedení rešeršní části a zohlednění všech podmínek, požadavků a možností katedry, byl sestaven návrh pro kompletní databázové řešení založené na procesu replikaci. Z databázových serverů, diskutovaných v kapitole 3.4, byl vybrán server PostgreSQL hned z několika důvodů. Jedná se o plnohodnotný databázový systém dostupný zdarma se všemi nástroji, je široce používaný v oblasti geoinformačních technologií, je multiplatfomní a od verze ArcGIS 9.3 plně podporováný produkty ArcGIS.

Byl navržen replikační cluster s nejméně třemi servery z důvodů, které již byly diskutovány v kapitole 3.2. Celý cluster poběží na stejné platformě a proto bude možno použít streaming replikaci, která jakožto nativní řešení PostgreSQL, nabízí větší stabilitu a bezpečnost, díky přenosu transakčních logů, než jiná diskutovaná řešení. Byla zvolena jednosměrná master-slave replikace, cluster tedy bude obsahovat jeden master a dva (popř. více) slave serverů. Aby nedošlo ke ztrátě dat v případě, že by master server spadl dřív, než se data zkopírují na slave server, pro první slave (slave1) byla zvolena varianta synchronní replikace. Je vhodné, aby servery běžely v lokální síti, protože se tím snižuje pravděpodobnost, že by došlo k výpadku spojení mezi master a slave1 server a nebylo by tak možno na master zapisovat.

Druhý server (slave2) bude replikovat asynchronně a zároveň, aby nedocházelo k přetížení master serveru, bude replikace probíhat ze slave1 na slave2, tedy kaskádově. Ze slave2 lze dále tvořit pravidelnou, například denní nebo týdenní, zálohu pomocí ulitily pg_dump, která je více popsána v kapitole 4.4. Záloha přes pg_dump tak nebude zatěžovat master server a sama o sobě bude probíhat rychleji, než by tomu bylo na master serveru, který je již tak velmi vytížen dalšími procesy.

Uživatelé, kteří budou mít právo do databáze číst i zapisovat, se budou připojovat

skrze pgpool, jehož výhody a možnosti byly popsány v kapitole 3.5.3. Uživatelům to usnadní práci, protože si nebudou muset hlídat, ke kterému ze serverů se připojit na základě jejich aktuálního dotazu. pgpool se bude tvářit jako jakákoli jiná databáze, ke které se klienti přihlásí bez ohledu na typ jejich dotazu a on sám pak rozhodne, ke kterému ze serverů klienta přihlásí. Tím bude mít zároveň možnost rozložit zátěž na dostupné uzly v clusteru dle počtu konkrétních dotazů. Pro ještě větší efektivitu provozu databáze bude pgpool uchovávat databázová spojení a při novém dotazu využije stávajícího spojení, místo aby vytvářel spojení nové. Tímto se zajistí plynulost a zvýší rychlost provozu databáze.



Obrázek 8: Návrh replikačního řešení

Vzhledem k tomu, že se klienti k databázi budou přistupovat skrze pgpool, není potřeba aby jednotlivé uzly v clusteru měly veřejnou IP adresu. Plně dostačuje, že servery poběží na lokální síti a pouze pgpool bude na serveru s veřejnou IP, čímž se

zajistí, že data budou přístupná i skrze internet.

Návrh počítá také s externími pracovišti, která budou často přistupovat do databáze s právem čtení, a budou mít zájem o zrychlení přístupu k datům tím, že se slave server přesune na jejich pracoviště, tedy na hardware, který bude připojen do jejich lokální sítě. Typ replikace se zvolí podle jejich operačního systému a jeho architektury. Pokud se bude jednat o shodný systém, jaký bude použit ve výše popsaném clusteru, pak bude možno použít asynchronní streaming replikaci, naopak pokud se bude bude jednat o systém jiný, bude použita Slony-I replikace.

4.4 Příprava prostředí pro konfiguraci replikace

Na začátku je potřeba připravit prostředí hned s několika závislostmi. Hlavním používaným software je PostgreSQL s extenzemi PostGIS, Slony-I a pgpool. Informace o instalacích jednotlivých komponent jsou dostupné na jejich webových stránkách. Ve Windows si stačí stáhnout pouze instalační balík pro PostgreSQL, který umožňuje instalaci databázového systému včetně všech výše zmíněných extenzí. Pro grafickou administraci databáze je doporučený, ale nepovinný, program PgAdminIII[20], který je taktéž multiplatformní. Většina příkazů je zde popisována skrze příkazový řádek, mají však i své ekvivalentní použití skrze grafické rozhraní.

Všechny technologie byly testovány na operačním systému Debian-based Linux, tedy některé příklady použité v této kapitole, především pak ukázky absolutních cest k souborům, odpovídají struktuře tohoto systému. Slony-I bylo navíc vyzkoušeno také na systému Windows XP. Bylo používáno databázového systému PostgreSQL ve verzích PostgreSQL 9.1 a 9.3, Postgis verze 1.5 a 2.1, Slony-I verze 2.1 a pgpool verze 3.1 a 3.3.

Pro databázové servery byla zvolena tři datová uložiště, pro jejichž přehled viz tab. 6. IP adresy byly pro větší názornost upraveny na rozsah běžné lokální sítě. Vzhledem k tomu, že se do databáze bude přistupovat skrze pgpoool, není potřeba, aby kterýkoli z níže vypsaných serverů, měl veřejnou IP adresu. Všechny servery běží na defaultním portu 5432, který je standardem pro PostgreSQL.

[20] http://www.pgadmin.org/

Tabulka 6: Přehled databázových serverů

název serveru	IP adresa	port
master	192.168.0.100	5432
slave1	192.168.0.101	5432
slave2	192.168.0.102	5432

Aby bylo možné pracovat s databází, je nejdříve nutné chápat význam jednotlivých konfiguračních souborů a mít přehled o souborové struktuře PostgreSQL. Vzhledem k tomu, že si ji každý systém uzpůsobuje podle sebe, nezbývá než po instalaci PostgreSQL nastudovat, kde se jaký soubor nachází. Existuje tabulka `pg_settings`, která uchovává veškeré informace o nastavení databáze. SQL příkazem volajícím tuto tabulku je možno vypsat absolutní cestu k datům (`data_directory`) a cestu k souboru, který uchovává PID (process-ID) běžícího procesu (`external_pid_file`). Také popisuje tři hlavní konfigurační soubory:

- `postgres.conf`, který definuje obecné nastavení databáze,
- `pg_hba.conf`, který povoluje konkrétním uživatelům přístup z určitých IP adres,
- `pg_ident.conf`, který slouží k mapování uživatel operačního systému na uživatele PostgreSQL (Obe a Hsu, 2012).

Příklad SQL příkazu, spuštěného na serveru master serveru, který vypíše umístění jednotlivých souborů a složek:

```
SELECT name, setting FROM pg_settings WHERE category = '
   File Locations';
```

```
name                 settings
-------------------  +  ----------------------------------------
data_directory       |  /var/lib/postgresql/9.1/main
external_pid_file    |  /var/run/postgresql/9.1-main.pid
hba_file             |  /etc/postgresql/9.1/main/pg_hba.conf
config_file          |  /etc/postgresql/9.1/main/postgresql.conf
ident_file           |  /etc/postgresql/9.1/main/pg_ident.conf
```

U všech typů replikace je potřeba mít vytvořeného databázového uživatele s právem pro replikaci, pod kterým bude daný proces probíhat. Je možné vytvořit nového

uživatele a nastavit mu tato práva nebo použít již existující účet `postgres`, který jako `SUPERUSER` obsahuje také práva pro replikaci. Je však potřeba mu hned na začátku změnit heslo.

Příklad změny hesla uživatele `postgres` na master serveru:

```
ALTER ROLE postgres PASSSWORD 'kgigis';
```

Příklad vytvoření nového uživatele `replikator` s přidáním práv pro replikaci na master serveru:

```
CREATE ROLE replikator REPLICATION ENCRYPTED PASSWORD '
    kgigis';
```

Pokud se začíná databázovým systémem, který ještě neobsahuje žádná data, je vhodné replikaci spustit ještě před přidáváním dat a to z důvodu rychlosti přenosu dat na druhý databázový server. V případě, že již databáze naplněná daty je, není problém replikaci spustit, jen je třeba počítat s delším časem kopírovaní dat a větší opatrností při konfiguraci.

Oba typy replikace, vyžadují lehce odlišné přístupy při přípravě dat před spuštěním samotné replikace. Slony-I replikace vyžaduje mít předem vytvořenou strukturu databáze včetně tabulek a poté zajistit existenci totožné kopie na všech serverech v clusteru. Je možné toho dosáhnout použitím utility `pg_dump`, která data exportuje, na master servery, a `pg_restore`, která data importuje, na slave serveru. Tímto způsobem lze převádět jak strukturu databáze, tak data, a zároveň to umožňuje přenášet pouze vybrané části databáze.

Příklad exportu a importu dat z databáze do databáze dat skrze:

```
pg_dump > /tmp/dump.sql
pg_restore /tmp/dump.sql
```

Streaming replikace, která funguje pouze mezi dvěmi a více databázemi běžícími na stejném systému a architektuře, vyžaduje kopii celé datové složky s daty `data_directory`. Je mnoho způsobů, jak toho dosáhnout, například klasickým kopírováním skrze utilitu `cp`, resp. `scp` u vzdálených složek, nebo utilitou `rsync`. Kopírování dat za běhu databáze navíc vyžaduje použití příkazu `SELECT pg_start_backup`, který zajistí, že bude archivovat transakční log po celou dobu kopírovná, tedy až do té doby než proběhne příkaz `SELECT pg_stop_backup`. Tím nepřijdeme o žádné změny, které nastanou v průběhu kopírování dat.

Příklad možného způsobu zkopírování dat master serveru na repliku slave1 spuštěného z slave1:

```
SELECT pg_start_backup('backup', true)
```

```
scp -rv root@192.168.0.100:/var/lib/postgresql/9.1/
   main /var/lib/postgresql/9.1/main
rm /var/lib/postgresql/9.1/main/postmaster.pid

SELECT pg_stop_backup()
```

Stejný způsobem je možno provést i kopii na slave2.

Alternativou výše zmíněných nástrojů je utilita přímo určená pro zálohování dat v PostgreSQL `pg_basebackup`. Tento příkaz mimo jiné umožňuje kopírování dat za běhu replikace bez nutnosti použití `pg_start/stop_backup`.

Použití `pg_basebackup` pro vytvoření repliky:

```
pg_basebackup -D /var/lib/postgresql/9.1/main/ -U
   replikator -h 192.168.0.100
```

Kopírování dat je velice důležitý krok pro správný chod replikace. V případě, že se data nesprávně zkopírují, není možné replikaci zprovoznit.

Pří kopírování celé datové struktury je vhodné zajistit jednotlivým souborům a složkám správa. Vzhledem k tomu, že databázový systém zapisuje do složky s daty (`data_directory`), musí mít postgres, i po zkopírování celé datové struktury na jiný server, práva pro zápis.

A v neposlední řadě je potřeba zajistit konektivitu obou, resp. všech serverů v replikačním clusteru. S tím souvisí i nutnost nastavení povolení přístupů z IP adres slave serverů, kterou je možno zajistit skrze konfigurační soubor `pg_hba.conf`. Následující příklad ukazuje možné nastavení souboru `pg_hba.conf` na master serveru. Povoluje uživetelům market a replication, přihlášených z dané IP adresy, přistupovat na master server a číst, resp. replikovat data.

```
#host  DATABASE     USER         ADDRESS          METHOD
host   all          market       80.188.74.1/32   md5
host   replication  replication  80.188.74.1/32   md5
```

4.5 Konfigurace replikace

4.5.1 Streaming replikace

Jak bylo nastíněno v kapitole 4.3, databázové řešení staví na streaming replikaci a skládá se ze tří uzlů v clusteru, jednoho master serveru a dvou slave serverů. Pokud je správně provedena příprava dle kapitoly 4.4, samotné nastavení replikace není nijak

náročné. V první fázi je potřeba konfigurace souboru `postgresql.conf` na master serveru. Pro asynchonní replikaci stačí editace parametrů:

- `wal_level`, který určuje, kolik informací má být zapsáno do transakčního logu (WAL) a
- `max_wal_senders`, který odpovídá maximálnímu počtu připojených slave serverů.

Hodnota `wal_level`, stanovena na `hot_stadby`, zajistí, že na slave serveru bude umožněno dotazování. Vzhledem k tomu, že se bude na master server připojovat pouze `slave1` a všechny další slave servery se poté budou připojovat k němu, hodnota `1` zcela dostačuje. Je však možné hodnotu rovnou navýšit, aby se soubor v budoucnu nemusel znovu editovat z důvodu připojení dalšího serveru.

Tyto dva parametry stačí pro achynchronní replikace, pro nastavení synchronní je ještě potřeba přidat `synchronous_standby_names`, jehož hodnota může být libovolné slovo.

Konfigurace `postgres.conf` na master serveru:

```
wal_level = hot_standby
max_wal_senders = 1
synchronous_standby_names = 'gis'
```

Stejně tak je potřeba konfigurovat `postgresql.conf` na slave serverech. Hodnoty `wal_level` a `max_level_sender` můžou a nemusí zůstat stejné jako na masteru. Pokud však má být slave připraven zastoupit master server v případě jeho výpadku, pak je vhodné, aby hodnoty byly nastaveny shodně. Na slave serveru je dále potřeba editovat:

- `hot_standby`, který určuje, zda je na slave serveru umožněno dotazování a
- `hot_stadby_feedback`, který udává, zda bude replika informovat master server o příkazech, které na ní byly provedeny.

Konfigurace `postgresql.conf` shodně na obou slave serverech (na `slave1` a `slave2`):

```
wal_level = hot_standby
max_wal_senders = 5
hot_standby = on
hot_standby_feedback = on
```

Posledním krokem je vytvoření souboru `recovery.conf` na slave serveru ve složce s daty, který definuje parametry:

- `standby_mode`, který povoluje či zakazuje použití serveru jako slave a
- `primary_conninfo`, který nastavuje informace o serveru, ze kterého budou data replikována. Parametr nastavuje IP adresu serveru, ze kterého se data budou replikovat, název replikačního uživatele a jeho heslo a v případě synchronní replikace ještě klíčové slovo, které musí být shodné s hodnotou, která byla nastavená na master serveru v souboru `postgresql.conf` v parametru `synchronous_standby_name`.

Ukázka konfigurace `recovery.conf` uloženého ve složce s daty na `slave1`, který je připojován na master server a běží jako synchronní:

```
standby_mode='on'
primary_conninfo='host=192.168.1.100 user=replikator
   password=kgigis application_name=gis'
```

V návrhu je počítáno s kaskádovou replikací, tedy s tím, že se `slave1` bude připojovat na `slave2` místo na master server. To lze nastavit úpravou souboru `recovery.conf`, kde IP adresa parametru `host` bude odpovídat IP adrese `slave1` serveru.

Ukázka konfigurace `recovery.conf` uloženého ve složce s daty na `slave2`, který běží jako asynchronní a je kaskádově připojován ke slave1:

```
standby_mode='on'
primary_conninfo='host=192.168.1.101 user=replikator
   password=kgigis'
```

To, že je replikace správně nastavená, lze zkontrolovat několika způsoby. Připojené repliky lze vypsat pomocí SQL příkazu `pg_stat_replication`, kde poslední parametr udává, zda se jedná o synchronní nebo asychronní replikaci.

Spuštění SQL příkazu `pg_stat_replication` z master serveru:

```
SELECT usename, application_name, client_addr, state,
   sync_state FROM pg_stat_replication ;
```

```
-[ RECORD 1 ]--   +  -------
usename           |  replikator
application_name  |  gis
client_addr       |  192.168.1.101
state             |  streaming
sync_state        |  sync
```

Spuštění stejného SQL příkazu `pg_stat_replication` ze `slave1`:

```
-[ RECORD 1 ]--   +  -------
usename           |  replikator
application_name  |  walreceiver
client_addr       |  192.168.1.102
state             |  streaming
sync_state        |  async
```

Stejně tak lze na slave serveru zjistit, zda běží jako replika či nikoli pomocí SQL příkazu `pg_is_in_recovery()`:

```
postgres=# select pg_is_in_recovery();

 pg_is_in_recovery
-------------------
 t
(1 row)
```

Zároveň na slave server nesmí být možné zapsat žádná data:

```
INSERT INTO student (jmeno) VALUES('Jan Vlasovec');

ERROR:  cannot execute INSERT in a read-only transaction
```

Připojené repliky lze vypsat pomocí SQL příkazu `pg_stat_replication`, kde poslední parametr udává, zda se jedná o synchronní nebo asychronní replikaci:

V případě, že master server spadne, je možné během pár minut vyměnit role, určit jako master jeden ze slave serverů. Lze to udělat několik způsoby, jedním z nich je sledování existence souboru, který je definován v souboru `recovery.conf` na kterémkoli slave serveru:

```
trigger_file = '/tmp/trigger.txt'
```

Název souboru může být zvolen libovolně a může být zcela prázdný. Slave server pouze hlídá jeho existenci a jen to, že se soubor objeví v dané složce, způsobí, že se ze slave serveru stane master. Obsah souboru můžou tvořit další instrukce, které se můžou ovlivnit další chod databáze.

4.5.2 Slony-I

Jak už bylo zmíněno v kapitole 3.5.1, u Slony-I není možné replikovat strukturu databáze. Výhodou však je, že lze vybrat pouze některé tabulky k replikaci a zároveň, že se nemusí shodovat názvy databází. V tomto ohledu tedy nabízí velkou variabilitu propojení. Jak bylo nastíněno v kapitole 4.4, nastavení replikace začíná přípravou uživatele, pod kterým bude proces probíhat, vytvořením datové struktury a zajištění shodné kopie dat na všech uzlech v clusteru.

Pro názornost byla vytvořena databáze *studenti* pomocí příkazu `CREATE DATABASE` a tabulky `student` a `rodne_mesto`, u kterých byl nastaven `primary key` jakožto podmínka Slony-I replikace, pomocí SQL příkazu `CREATE TABLE`:

```
CREATE DATABASE studenti;
studenti=# CREATE TABLE student (id int, jmeno varchar,
   id_rodne_mesto int, primary key(id));

NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit
    index "student_pkey" for table "student"
CREATE TABLE

studenti=# CREATE TABLE rodne_mesto (id int, jmeno
   varchar, umisteni geometry, primary key(id));

NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit
    index "rodne_mesto_pkey" for table "rodne_mesto"
CREATE TABLE
```

Dále tento způsob obnáší vytvoření konfiguračních skriptů, které zajistí inicializaci replikace. Slony-I pro to používá vlastní konfigurační jazyk, pomocí kterého se nastavují konkrétní požadavky na replikaci. Na začátku se díky němu sestaví konfigurační soubor k inicializaci replikace, později se používá pro jakýkoli zásah do replikace, například přidání další tabulky do replikačního setu nebo změny struktury databáze. Tento vzniklý konfigurační skript se provede pomocí utility `slonik`, která se vždy spouští jednorázově a vykonává požadavky definované v konfiguračním souboru.

Ukázka konfiguračního skriptu nazvaného `init_master.txt` uloženého na master serveru pro inicializaci replikačního clusteru (init cluster):

```
$master = 'dbname=studenti host=192.168.1.100 user=
   replikator password=kgigis'
$slave1 = 'dbname=studenti host=192.168.1.101 user=
   replikator password=kgigis'
$slave2 = 'dbname=studenti host=192.168.1.102 user=
   replikator password=kgigis'

# nazev clusteru
cluster name = gis_cluster;

# definice jednotlivych uzlu v clusteru
node 1 admin conninfo=$master;
node 2 admin conninfo=$slave1;
node 3 admin conninfo=$slave2;

# inicializace clusteru
init cluster (id=1, commnet = 'master');
store node   (id=2, comment = 'slave1', event node=1);
store node   (id=3, comment = 'slave2', event node=1);

# vytvoreni replikacniho setu
create set (id=1, origin=1, comment='Tabulky k replikaci
   ');
# pridani tabulek do setu
# prvni id odpovida id setu
# druhe id odpovida id uzlu masteru
# treti id je id nove pridane tabulky
set add table (set id=1, origin=1, id=1, fully qualified
    name = 'public.student', comment='seznam studentu');
set add table (set id=1, origin=1, id=2, fully qualified
    name = 'public.rodne_mesto', comment='seznam mest');

store path (server=1, client=2, conninfo=$slave1);
store path (server=1, client=3, conninfo=$slave2);
store path (server=2, client=1, conninfo=$master);
store path (server=2, client=3, conninfo=$slave2);
store path (server=3, client=1, conninfo=$master);
```

```
store path (server=3, client=2, conninfo=$slave1);

store listen (origin=1, provider=2, receiver=1);
store listen (origin=1, provider=1, receiver=2);
store listen (origin=1, provider=2, receiver=3);
```

Skript Slonik volá příkazy:

- `cluster name` představující jedinečný název pro daný cluster,
- `node ival/číslo admin conninfo` definující všechny uzly v clusteru a parametry jejich připojení k databázi,
- `init cluster`, který inicializuje cluster a nastavuje master server jako uzel s id 1,
- `store node`, který vytváří další uzly,
- `create set` vytvářející soubor tabulek určených k replikaci,
- `set add table` přidávající vždy jednu tabulku do replikačního setu s identickým id,
- `store path`, který nastavuje cesty mezi jednotlivými uzly a
- `store listen` nastavující naslouchání jednotlivých uzlů.

Slony-I rozlišuje tři druhy serverů:

- `origin` odpovídá master serveru, tedy jedinému uzlu, kterému je povoleno zapisování,
- `subscriber` je ekvivalentem slave serveru s právy čtení a
- `provider` je poskytovatel dat, může to být master server, ale při kaskádové replikaci, kterýkoliv ze slave serverů.

Konfigurační skript pro inicializaci clusteru se spustí ze stejné složky, ve které je uložen daný soubor, příkazem `slonik` a názvem souboru, viz příklad:

```
# slonik init_master.txt
```

Pomocí dalších skriptu se mohou slave servery přihlásit odběru replikačního setu. Příklad vytvoření skriptu `subscribe_slave1.txt` pro přidání serveru do existujícího clusteru (subscribe set):

```
# nazev clusteru
cluster name = gis_cluster;

# definice jednotlivych uzlu v clusteru
node 1 admin conninfo=$master;
node 2 admin conninfo=$slave1;
node 3 admin conninfo=$slave2;

subscribe set (id=1, provider=1, receiver=2, forward=yes
   );
```

Spuštění skriptu `slonik` pro přidání slave1 do clusteru:

```
slonik subscribe_slave1.txt
```

Stejný způsobem se připojít k odběru i slave2.

To, že se vytvořit cluster a tabulka do něj byla přidána, lze zkontrolovat ve výpisu, kde nově přibyly triggery, které sledují změny, které v tabulce nastanou:

```
studenti=# /d student
            Tabulka "public.student"
Column          |  Type               |  Modifiers
--------        +  ----------         +  -------
id              |  integer            |  not null
jmeno           |  character varying  |
id_rodne_mesto  |  integer            |
Indexes: "student_pkey"PRIMARY KEY, btree (id)
Triggers: _gis_cluster_logtrigger AFTER INSERT OR DELETE
OR UPDATE ON repl_test FOR EACH ROW EXECUTE PROCEDURE
_is_cluster.logtrigger('_gis_cluster', '1', 'k')
Disabled triggers: _gis_cluster_denyaccess BEFORE INSERT
OR DELETE OR UPDATE ON repl_test FOR EACH ROW EXECUTE PROCEDURE
_gis_cluster.denyaccess('_gis_cluster')
```

Běh replikace je zajištěn vlastním démonem, který je možnost spustit v okamžiku, kdy je vytvořen cluster a všechny repliky jsou do něj přidány. Démon `slon`, který je potřeba spustit na všech uzlech, v parametrech přebírá název clusteru a hodnoty připojení daného uzlu k databázi. Je důležité, aby log po spuštění nevypisoval žádné chyby, jinak je potřeba zkontrolovat všechny příkazy konfiguračních souborů.

Příklad spuštění démona na master serveru s názvem clusteru a parametry připojení k serveru:

```
slon gis_cluster 'host=192.168.1.100 dbname=student
   user=replikator'
```

Obdobně je démona spuštěn i na obou slave serverech:

```
slon gis_cluster 'host=192.168.1.101 dbname=student
   user=replikator'
slon gis_cluster 'host=192.168.1.102 dbname=student
   user=replikator'
```

Podoboně jako u streaming replikace, lze zkontrolovat, že replikace běží správně, přidáním nového záznamu na slave server. Pokud nepovolí přidání a vypíše následující chybu, znamená, že je replikace správně nastavená a funkční.

```
studenti=# INSERT INTO student (jmeno) VALUES ('Josef
   Kraus');

ERROR:  Slony-I: Table repl_names is replicated and
   cannot be modified on a subscriber node - role=0
```

Přidání další tabulky či jakákoli jiná změna struktury databáze probíhá v několika krocích. Nejdříve je potřeba vytvořit soubor, který bude obsahovat SQL příkaz provádějící zvolenou změnu databáze. Poté se spustí program slonik, který zavolá soubor s SQL příkazem a vykoná jej na všech uzlech clusteru.

Vytvoření souboru `createTable.sql` ve slozce `/tmp/` s SQL příkazem `CREATE TABLE`:

```
CREATE TABLE predmet (id int, jmeno varchar, primary
   key(id));
```

Vytvoření skriptu `ddlZmena.txt`, který umožní přidání tabulky za chodu replikace:

```
# nazev clusteru
cluster name = gis_cluster;

# definice jednotlivych uzlu v clusteru
node 1 admin conninfo=$master;
node 2 admin conninfo=$slave1;
node 3 admin conninfo=$slave2;
```

```
execute script (
  SET ID = 1,
  filename = '/tmp/createTable.sql',
  event node = 1
);
```

Spuštění programu `slonik`, který spustí skript a vykoná daný SQL příkaz na všech uzlech:

```
slonik ddlZmena.txt

DDL script consisting of 2 SQL statements
DDL Statement 0: (0,67) [ CREATE TABLE predmety (id int,
    nazev varchar, primary key(id));]
slony_ddl.txt:6: NOTICE:  CREATE TABLE / PRIMARY KEY
   will create implicit index "predmety_pkey" for table
   "predmety"
DDL Statement 1: (67,69) [ ]
Submit DDL Event to subscribers...
```

Podrobný výpis informuje, že se změnilo DDL[21] schéma, které provádí příkazy CREATE, ALTER, DROP, tedy příkazy měnící strukturu databáze. Posledním řádkem potvrzuje, že se schéma zapsalo také na slave servery. Takto se tabulka přidá do databáze, nikoliv však do replikačního clusteru. K tomu je potřeba vytvořit další slonik skript.

Vytvoření skriptu `add_to_set.txt` pro přidání tabulky do replikačního clusteru:

```
# nazev clusteru
cluster name = gis_cluster;

# definice jednotlivych uzlu v clusteru
node 1 admin conninfo=$master;
node 2 admin conninfo=$slave1;
node 3 admin conninfo=$slave2;

# definice noveho setu (id 2)
create set (id=2, origin=1, comment='Dalsi tabulky k
   replikaci');

# pridani nove vytvorene tabulky
```

[21] Data Definition Language

```
set add table (set id=2, origin=1, id=3, fully qualified
    name = 'public.predmet', comment='seznam predmetu');

# pridani noveho setu id=2
subscribe set (id=2, provider=1, receiver=2);

# spojeni setu id2 se setem id1
merge set(id=2, add id=1, origin=1);
```

Spuštění skriptu `slonik`, které příkazy vykoná

```
slonik add_to_set.txt
```

Podobně je možné provést také smazání tabulky pomocí parametru `DROP SET`.

4.6 Rozložení zátěže mezi replikační servery

Doposud je samotné replikační rešení nastaveno tak, že se uživatelé musí přihlašovat vždy ke konkrétnímu serveru v clusteru. To není úplně vhodné řešení, protože v případě, že se na jeden uzel, například v rámci cvičení, připojí velký počet klientů, může se stát, že se daný server přetíží, zatímco druhý slave server nezaznamená žádnou zátěž. Zároveň by pro plynulý běh databázového clusteru bylo vhodné, aby se uživatelé s dotazy SELECT, připojovali pouze na slave servery, zatímco příkazy, které modifikují data (CREATE, INSERT, DELETE), byly vykonány na master serveru.

Oba výše diskutované problémy řeší nástroj `pgpool`, který bude použit v módu `master/slave` a který zajistí, že se všechny příkazy provádějící změnu v databázi pošlou na master server a ostatní dotazy budou rozloženy mezi slave servery. Dotazy na master server bohužel nemohou být rozloženy, protože v replikačním clusteru smí být vždy jen jeden server s právem pro zapis. Toto řešení zajistí také zvýšení dostupnosti dat, protože tím, že bude mít uživatel přístup ke všem serverům místo pouze jednoho, nebude vůbec omezen výpadkem kterékoli uzlů na straně serveru.

Konfigurace pgpool se skládá ze tří hlavních souborů:

- `pcp.conf`, který nastavuje přístupové jméno a heslo pro administrátora pgpool,
- `pool_hba.conf`, který povoluje přístupy k pgpool pro konkrétní uživatele, soubor je podobný jako pg_hba.conf v PostgreSQL a
- `pgpool.conf`, který zajišťuje obecné nastavení.

Nejdříve je potřeba nastavit heslo pro administrátora, který bude moct měnit nastavení a sledovat statistiky. Heslo lze vytvořit pomocí ulitity `pg_md5`, která vrátí

zadané zašifrované heslo, které je poté potřeba zkopírovat do souboru `pcp.conf` a doplnit jej o uživatelské jméno.

Příklad zašifrování hesla `kgigis` pomocí `pg_md5` s vypsaným výsledkem:

```
pg_md5 kgigis
eea831dcf9dc85ace5836024f3a253e7
```

Přidání přihlašovacích údajů pro administrátora do souboru `pcp.conf`:

```
#username:[password encrypted in md5]
kgi:eea831dcf9dc85ace5836024f3a253e7
```

Zvolený master/slave mód počítá s již nastavenou replikací a podporuje jak streaming replikaci, tak Slony-I. Pro usnadnění konfigurace pgpool poskytuje příklady všech konfiguračních souborů včetně různých typů nastavení. Pro master/slave mód jsou připraveny hned dva příkladové soubory, `pgpool.conf.sample-master-slave` pro Slony-I a `pgpool.conf.sample-stream` pro streaming replikaci. V případě použití této šablony, je nejdříve potřeba ji přesunout do složky s konfiguračními soubory a poté přejmenovat na `pgpool.conf`. Šablona zajistí základní konfiguraci pro mód master/slave a streaming replikaci, nastaví parametry:

- `replication_mode`, který povoluje replikaci, výchozí hodnota je off,
- `load_balance_mode`, který umožňuje rozložení zátěže, východí hodnota je off
- `master_slave_mode`, který povoluje propojení k master a slave serverů,
- `master_slave_sub_mode`, který nastavuje hodnotu na 'stream' v případě streaming replikace a 'slony' v případě Slony-I,
- `sr_check_period = 10`, který nastavuje jak často má systém zkontrolovat pozici v XLOGu, aby zjistil, jestli je zpoždění příliš vysoké, či nikoli,
- `delay_threshold`, která definuje maximální možné zpoždění slave za master serverem, menší zpozdeni je možno nastavit v případě, ze je potřeba, aby replikace proběhla velice rychle (hodnota je určena v bytech) a

Část konfiguračního souboru `pgpool.cong` s výpisem důležitějších parametrů a jejich hodnot:

```
replication_mode = off
load_balance_mode = on
master_slave_mode = on
master_slave_sub_mode = 'stream'
```

```
sr_check_period = 10
log_standby_delay = 'if_over_threshold'
delay_threshold = 10000000
```

Další část konfiguračního souboru přidává konkrétní uzly, ke kterým bude po spuštění pgpool možno přistupovat právě přes pgpool. Pro uživatele se v podstatě nic nezmění, k databázi se připojí stejně, jako by se přihlašovali přímo, s jediným rozdílem, že použijí port definovaný v tom souboru parametrem `port`. Číslo za parametrem začínajícím `backend` vždy značí číslo daného nodu přidaného do pgpool. V tomto případě jsou přidány tři uzly, který byla přiřazena čísla 0 pro master, 1 pro slave1, 2 pro slave2. Parametr:

- `listen_addresses` definuje IP adresy, na kterých pgpool naslouchá,
- `port` definuje port, kterým se uživatelé budou přihlašovat k databázovému clusteru (místo nejčastnějšího 5432),
- `pcp_port` určuje port, kterým se bude přihlašovat administrátor,
- `backend_hostname` nastavuje hosta nebo IP adresu daného uzlu,
- `backend_port0` určuje port, na kterém naslouchá daný uzel,
- `backend_weight` umožňuje zvýšit danému uzlu zátěž, čím vyšší číslo, tím více datazů bude směřováno na tento uzel místo,
- `backend_data_directory` určuje, kde jsou uložená data daného uzlu a
- `backend_flag` povoluje nebo zakazuje daný uzel použít jako master v případě výpadku master servera.

Čast konfigurace souboru `pgpool.conf`, která nastavuje jednotlivé uzly:

```
listen_addresses = '*'
port = 9999
pcp_port = 9898

# node 0 - master server
backend_hostname0 = '192.168.1.100'
backend_port0 = 5432
backend_weight0 = 1
backend_data_directory0 ='/var/lib/postgresql/9.3/main'
backend_flag0 = 'ALLOW_TO_FAILOVER'
```

```
# node1 - slave1
backend_hostname1 = '192.168.1.101'
backend_port1 = 5432
backend_weight1 = 1
backend_data_directory1 = '/var/lib/postgresql/9.1/main'
backend_flag1 = 'ALLOW_TO_FAILOVER'

# node2 - slave2
backend_hostname2 = '192.168.1.102'
backend_port2 = 5432
backend_weight2 = 1
backend_data_directory2 = '/var/lib/postgresql/9.1/main'
backend_flag2 = 'ALLOW_TO_FAILOVER'
```

Pro správný běh pgpool je potřeba, aby slave server, který má hodnotu `backend_flag2 'ALLOW_TO_FAILOVER'` měl v souboru `recovery.conf` přidán parametr `trigger_file`, již popisovaný v kapitole 4.5.1.

Než dojde k samotnému spuštění pgpool, je možno zkontrolovat, zda jsou všechny uzly přidány a běží, tak jak mají. Lze použít utilitu `pcp_node_count`, který vypíše počet aktuálně přidaných uzlů. Zadání dále vyžaduje difinici parametrů v pořadí:

- `timeout`, který učí maximální čas, po který se má snažit o vykonání příkazu v sekundách,
- `hostname`, který definuje pgpool IP,
- `port` definovaný pro administrátora,
- `username` odpovídá uživatelskému jménu zadanému v pcp.conf a
- `password` odpovídá heslu definovanámu v pcp.conf a

Spuštění nástroje `pcp_node_count` s maximálním časem provedení 30 sekund, pgpool běžícím na localhostu, portem 9898, uživatel kgi a heslem kgigis:

```
pcp_node_count 30 localhost 9898 kgi kgigis
3
```

Obdobně lze získlad informace o konkrétních uzlech utilitou `pcp_node_info`, která navíc přidává parametr `nodeID`, který odpovídá ID uzlu, o kterém chceme získat informace.

Spuštění nástroje `pcp_node_info` pro uzly 0, 1 a 2 s maximálním časem provedení 30 sekund, pgpool běžícím na IP adrese 127.0.0.1, portem 9898, uživatel kgi a heslem kgigis:

```
pcp_node_info 30 192.169.1.100 9898 kgi kgigis 0
192.168.1.100 5432 1 0.333333
pcp_node_info 30 192.169.1.101 9898 kgi kgigis 1
192.168.1.101 5432 1 0.333333
pcp_node_info 30 192.169.1.102 9898 kgi kgigis 2
192.168.1.102 5432 1 0.333333
```

pgpool nabízí ještě další nástroje, pomocí který lze získat informace o nastavení pgpool (`pcp_pool_info` nebo `pcp_promote_node` pro změnu uzlů z master na slave a opačně[22].

Posledním krokem je spuštění démona pgpool. /etc/init.d/pgpool2 start|status|stop

[22]kompletní seznam nástrojů pgpool na `http://www.pgpool.net/docs/latest/pgpool-en.html#pcp_command`

5 DISKUZE

6 ZÁVĚR

LITERATURA

BELL, C., KINDAHL, M., THALMANN, L. *MySQL High Availability.* Vyd. 1. Sebastopol, CA: O'Reilly Media, Inc, 2010. ISBN 978-059-6807-306.

BöSZöRMENYI, Z., SCHöNIG, H.-J. *PostgreSQL Replication: Understand basic replication concepts and efficiently replicate PostgreSQL using high-end techniques to protect your data and run your server without interruptions.* Birmingham: Packt Publishing, 2013. ISBN 978-1-84951-672-3.

CHACON, S. *Pro Git.* Edice CZ.NIC. Praha: CZ.NIC, 2009. ISBN 978-80-904248-1-4.

CONNOLLY, T. *Database Systems: A Practical Approach to Design, Implementation, and Management.* Vyd. 4. Harlow: Addison-Wesley, 2005. ISBN 03-212-1025-5.

ESRI. *ArcGIS 9: Co je ArcGIS 9.2?* United States: ESRI Press, US, 2006. ISBN 15-894-8166-6.

ESRI. A quick tour of working with databases in arcgis. *ArcGIS Help 10.1 [online]*, 2013a. Dostupné z: `http://resources.arcgis.com/en/help/main/10.1/index.html#/A_quick_tour_of_working_with_databases_in_ArcGIS/019v00000008000000/`.

ESRI. Preparing data for replication. *ArcGIS Help 10.1 [online]*, 2013b. Dostupné z: `http://resources.arcgis.com/en/help/main/10.1/index.html#/Preparing_data_for_replication/003n000000z5000000/`.

LAW, D. Enterprise geodatabase 101: A review of design and key features for gis managers and database administrators. *Esri: Understanding our world. [online]*, 2008. Dostupné z: `http://www.esri.com/news/arcuser/0408/entergdb_101.html`.

LEITER, C. *Beginning Microsoft SQL Server 2008 Administration.* Indianapolis, IN: Wiley Pub., 2009. ISBN 978-047-0440-919.

MICROSOFT. SQL server - replication. *Microsoft [online]*, 2013. Dostupné z: `http://technet.microsoft.com/en-us/library/ms151198(v=sql.100).aspx`.

MOMJIAN, B. *PostgreSQL: Introduction and Concepts.* Boston, MA: Addison-Wesley, 2001. ISBN 02-017-0331-9.

OBE, R., HSU, L. *PostGIS in Action.* London: Pearson Education [distributor], 2011. ISBN 19-351-8226-9.

OBE, R., HSU, L. *Postgresql: Up and Running.* Sebastopol, CA: O'Reilly, 2012. ISBN 978-144-9326-333.

OPENGEO. Introduction to postgis [online]. *Section 17: Geography*, 2012b. Dostupné z: `http://workshops.opengeo.org/stack-intro/openlayers.html`.

OPPEL, A. J. *Databases: A Beginner's Guide*. New York: McGraw-Hill, 2009. ISBN 00-716-0846-X.

OSGEO. Postgis and arcsde/arcgis articles. *PostGIS Tracker and Wiki [online]*, 2013. Dostupné z: `http://trac.osgeo.org/postgis/wiki/UsersWikiPostgisarcgis`.

GLOBAL DEVELOPMENT GROUP. What is pgpool-ii? *Pgpool Wiki*, 2013. Dostupné z: `http://www.pgpool.net/docs/latest/pgpool-en.html`.

POSTGRESQL. Faq. *PostgreSQL wiki [online]*, 2012. Dostupné z: `http://wiki.postgresql.org/wiki/FAQ`.

RIGGS, S., KROSING, H. *PostgreSQL 9 Administration Cookbook: Solve real-world PostgreSQL problems with over 100 simple, yet incredibly effective recipes*. Birmingham: Packt Publishing, 2010. ISBN 978-1-849510-28-8.

WHALEN, E. a. k. *Microsoft SQL Server 2005: velký průvodce administrátora*. Vyd. 1. / *Edice Administrace (Computer Press)*. Brno: Computer Press, 2008. ISBN 978-80-251-1949-5.

ČINčURA, J. MS SQL 2008 – prostorová data poprvé. *Databázový svět [online]*, 2009. Dostupné z: `http://www.dbsvet.cz/view.php?cisloclanku=2009101201`.

ŽáK, K. Historie relačních databází. *Root.cz*, 2001. Dostupné z: `http://www.root.cz/clanky/historie-relacnich-databazi/`.

SUMMARY

There is summary of all aims, methods and results in this chapter. Summary is not only translation of chapter Závěr. There is more information from chapters Cíle, Výsledky and Diskuze. Number of pages of Summary chapter is two at least. The style is Normalni Summary. Language is set to Angličina(Velká Británie) for automatic spell check. Do not use language Angličtina(USA).

PŘÍLOHY

SEZNAM PŘÍLOH

Volné přílohy

Příloha 1 CD

Popis sktruktury CD

Adresáře a soubory:

- `skripty/` - složka se skripty

- `web/` - webové stránky jako doplněk k diplomové práci

- `Solanska_DP.pdf` - text diplomové práce