Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra matematiky

DIPLOMOVÁ PRÁCE

Kontrola údajů a vazeb stavebních objektů vedených v RÚIAN

Plzeň, 2015 **Jindra Marvalová**

Zadání práce

Prohlášení

Předkládám tímto k posouzení a následné obhajobě diplomovou práci, kterou jsem zpracovala na závěr studia oboru Geomatika na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracovala samostatně a všechny použité literární prameny jsou uvedeny v seznamu literatury.

V Plzni, dne 20. 5. 2015 .....

Jindra Marvalová

Poděkování

Na tomto místě bych ráda poděkovala panu Ing. Karlu Janečkovi, Ph.D. za odborné vedení diplomové práce a panu Ing. Jiřímu Formánkovi, vedoucímu oddělení správy RÚIAN na ČÚZK v Praze, za poskytnutí konzultací.

Abstrakt

Cílem diplomové práce bylo přispět ke zvýšení kvality dat Registru územní identifikace, adres a nemovitostí (RÚIAN) se zaměřením na kontrolu údajů a vazeb stavebních objektů.

Celkem byly realizovány čtyři kontroly:

1. Vyhledání stavebních objektů, u kterých jsou totožné (blízké) definiční body.
2. Vyhledání stavebních objektů, které mají podezřelá čísla (popisná, evidenční).
3. Vyhledání stavebních objektů, které mají vazbu na stejnou parcelu katastru nemovitostí (KN).
4. Vyhledání stavebních objektů, které mají definiční bod stavebního objektu a definiční bod parcely vzdálený více jak definovaná max. vzdálenost.

U každé z kontrol bylo potřeba detailně popsat všechny případy, které mohou nastat. V některých případech bylo možné řešení úlohy převést na úlohu výpočetní geometrie, např. v případě kontroly 1. V takovém případě byla provedena rešerše dostupných algoritmů a na základě jejich porovnání provedena implementace vybraných algoritmů. Ve většině případů bylo pro každou z kontrol navrženo více vlastních řešení, jejichž funkčnost byla testována na vybraných vzorcích dat RÚIAN, konkrétně nad daty obce s rozšířenou působností (ORP) Plzeň a následně nad daty všech krajských měst ČR.

Byla provedena diskuze a porovnání jednotlivých navržených řešení, a to především z pohledu rychlosti výpočtu a úplnosti výsledků. Zásadním výstupem práce jsou doporučení jaká konkrétní řešení pro uvedené kontroly použít v produkčním prostředí RÚIAN a zdrojový kód pro jejich realizaci.

Klíčová slova

Registr územní identifikace, adres a nemovitostí; stavební objekt; kontroly dat; základní registry.

Abstract

The aim of the thesis was to contribute to improving the quality of data Register of territorial identification, addresses and real estate (RTIARE), which are focused on the control and data links of buildings.

A total of four controls were carried out:

1. Searching buildings which have identical (near) definition points.
2. Searching buildings which have suspicious numbers (house registration).
3. Search buildings which have a common bond to the same parcel in cadastre.
4. Searching buildings which have a definition point of a building and a definition point of a parcel far more than the max. distance.

For each of controls it is necessarily to describe in detail all cases that may occur. In some cases the solving of the task was possible to convert to the problem of computational geometry, e.g., in the case of control 1. In this case the recherche of available algorithms was created and on the basis of the comparison selected algorithms was implemented. In most cases, for each of the controls was designed more custom solutions whose functionality was tested on selected samples of data RTIARE, particularly over data municipality Plzeň and then over the data of all the regional capitals of the Czech Republic.

It was conducted discussion and comparison of proposed solutions, particularly in terms of the computation speed and the completeness of results. A crucial outcome of this work is recommendations which specific solutions for such controls used in a production environment RTIARE and a source code for their implementation.

Keywords

Register of territorial identification, addresses and real estate; building construction; building; inspection of data, basic registers.

Obsah

Seznam obrázků

Seznam grafů

Seznam tabulek

Seznam příkladů

1 Úvod

Evidence dat veřejné správy prošla zásadními změnami v závislosti na rozvoji informačních a komunikačních technologií a jejich postupným zaváděním do oblasti administrativy. Papírové kartotéky nahradily efektivnější databázové systémy, ale i tento způsob evidence dat se ukázal být nedostačující a to především proto, že většina dat nebyla mezi sebou propojena, velké množství dat bylo vedeno duplicitně v mnoha databázích na různých úřadech a docházelo k nekonzistenci dat. Tento problém řeší zavedení základních registrů. Základní registry veřejné správy představují hlavní pilíř elektronizace veřejné správy. Jejich účelem je zefektivnění a zrychlení činnosti veřejné správy, snížení byrokratické zátěže, rozšíření funkcionality a kvality služeb veřejné správy, urychlení přenosu informací mezi orgány veřejné správy a standardizace procesů ve veřejné správě.

Jedním ze čtyř základních registrů je Registr územní identifikace, adres a nemovitostí (RÚIAN). Jeho data jsou k dispozici zdarma a jsou bez registrace poskytována i pro komerční účely formou otevřených dat. Vzhledem k podrobnosti dat RÚIAN se tedy jedná o naprosto unikátní a jedinečný projekt.

Problémem, který může způsobovat komplikace při rozhodovacích procesech nad daty RÚIAN, je chybovost evidovaných dat. Většina chyb vznikla již při prvotním naplnění databáze, kdy byly do RÚIAN importována data z několika datových zdrojů, jež nebyly vždy konzistentní. Český úřad zeměměřický a katastrální (ČÚZK), správce RÚIAN, tento problém aktivně řeší. Pracuje na čištění a kontrole dat, a tím zvyšuje jejich kvalitu.

Kontrolami dat RÚIAN se zabývá i tato práce. Konkrétně se zaměřuje na kontrolu stavebních objektů evidovaných v RÚIAN. Cílem práce je navrhnout postupy, jak vyhledat konkrétní chyby, ke kterým může docházet u evidovaných stavebních objektů, tyto postupy optimalizovat, otestovat je a vyhodnotit jejich výsledky. Konkrétně se jedná o:

- vyhledání stavebních objektů, u kterých jsou totožné definiční body,

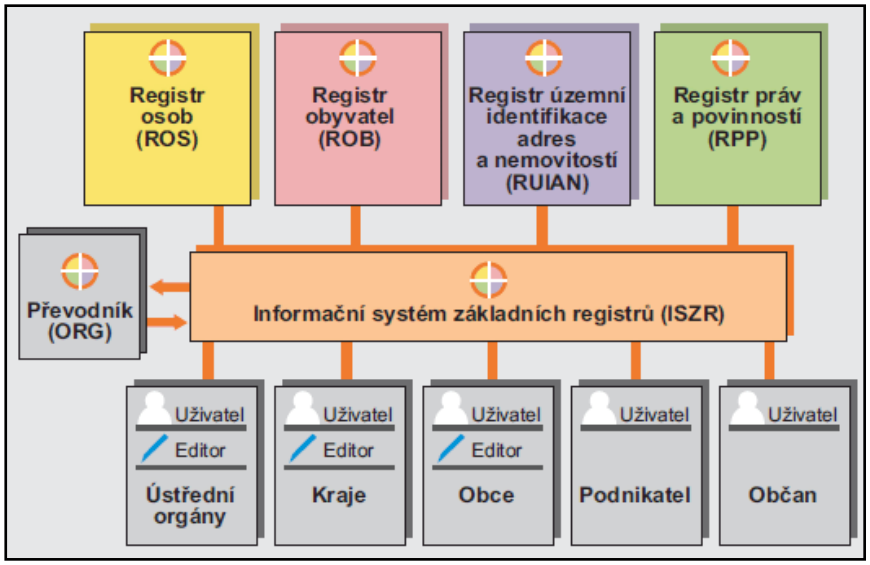
- vyhledání stavebních objektů, které mají podezřelá čísla domovní (popisná, evidenční),
- vyhledání stavebních objektů, které mají definiční bod stavebního objektu a definiční bod parcely vzdálený více jak definovaná max. vzdálenost,
- vyhledání stavebních objektů, které mají vazbu na stejnou parcelu KN.

Práce je členěna do deseti kapitol. Následující kapitola popisuje základní registry a strukturu Informačního systému základních registrů. Ve třetí kapitole je podrobně popsán RÚIAN, jeho vznik, struktura a údaje, které jsou v něm vedeny. Třetí popisuje testovací data a použitý software. V kapitolách 5 – 8 jsou popsány jednotlivé kontroly, možnosti jejich řešení a výsledky testování navrhovaných řešení. V kapitole 9 je popsáno testování kontrol nad datovou sadou krajských měst České republiky. V závěru je pak shrnutí práce a dosažených výsledků.

2 Informační systém základních registrů

Informační systém základních registrů (ISZR) je pojem, který je definován Zákonem o základních registrech jako informační systém veřejné správy, jehož prostřednictvím je zajišťováno sdílení dat mezi jednotlivými základními registry navzájem a základními registry a agendovými informačními systémy (AIS, Informační systémy Orgánů veřejné moci, které mohou komunikovat se základními registry), správa oprávnění přístupu k datům a další činnosti podle zákona o základních registrech. ISZR je definován jako jediné referenční rozhraní pro přístup k základním registrům. Pomocí tohoto rozhraní jsou poskytovány služby nad základními registry, zajišťuje aktuálnost dat, umožňuje výměnu dat se zeměmi EU a efektivní sdílení dat [1].

V rámci ISZR, který spravuje Správa základních registrů, fungují čtyři základní registry a Převodník identifikátorů fyzických osob (ORG). Ten spravuje Úřad pro ochranu osobních údajů. Činnost ORG je v ISZR zásadní pro ochranu osobních údajů. ORG totiž nahrazuje rodné číslo jako jednoznačný identifikátor fyzické osoby systémem identifikátorů bez významu. Tyto identifikátory se v jednotlivých registrech liší. Jsou uloženy právě v systému ORG bez dalších údajů o fyzické osobě [1]. Schéma fungování ISZR je znázorněno na obrázku 1.



Obr. 1: Schéma fungování ISZR [1].

2.1 Základní registry

Základním prvkem systému základních registrů je tzv. referenční údaj. Referenční údaj definuje aktuální právně platnou hodnotu příslušného údaje. Pokud není referenční údaj zpochybněn, je považován za platný a aktuální bez nutnosti dalšího ověřování.

Tyto referenční údaje jsou vedeny ve čtyřech registrech, jež jsou navzájem propojeny. Jedná se o Registr obyvatel, Registr osob, Registr územní identifikace adres a nemovitostí a Registr práv a povinností. Jednotlivé registry budou stručně popsány v následujících podkapitolách, které vychází především z [1], [2] a [3].

2.1.1 Základní registr obyvatel (Registr obyvatel, ROB)

Gestorem Registru obyvatel je Ministerstvo vnitra České republiky. Obsahuje aktuální referenční údaje o všech občanech ČR, cizincích s povolením k pobytu v ČR, cizincích, kterým byl na území ČR udělen azyl nebo doplňková ochrana a jiných fyzických osobách, o nichž jiný právní předpis stanoví, že budou vedeny v registru obyvatel. Vedenými údaji jsou: příjmení; jméno; adresa (ve formě vazby – kódu adresního místa - na referenční údaj o adrese v registru územní identifikace); datum, místo a okres narození; datum, místo a okres úmrtí; státní občanství, čísla elektronicky čitelných identifikačních dokladů a záznam o zřízení datové schránky a její identifikátor.

2.1.2 Základní registr právnických osob, podnikajících fyzických osob a orgánů veřejné moci (Registr osob, ROS)

Gestorem registru je Český statistický úřad (ČSÚ). Eviduje právnické osoby a organizační složky právnických osob, podnikající fyzické osoby, podnikající zahraniční osoby a organizační složky zahraničních osob, organizace s mezinárodním prvkem, organizační složky státu a orgány veřejné moci. Všechny osoby vedené v registru jsou identifikovány jednoznačným identifikátorem (IČ). O přidělování IČ dříve rozhodoval ČSÚ. Dnes o jeho přidělování rozhodují editoři ROS, kterými jsou obchodní rejstřík, rejstřík živnostenského podnikání

a informační systémy vybraných ministerstev, ústředních orgánů veřejné správy, obcí nebo krajů [4].

2.1.3 Základní registr územní identifikace, adres a nemovitostí (Registr územní identifikace, RÚIAN)

Správcem registru je Český úřad zeměměřický a katastrální. Slouží k evidenci údajů o územních prvcích, údajů o územně evidenčních jednotkách, adresách, územní identifikaci a k evidenci údajů o účelových územních prvcích. Jednotlivé prvky jsou zobrazovány na mapách státního mapového díla a digitálních mapách veřejné správy. Jako jediný ze systému základních registrů je veřejný, protože neobsahuje žádné zákonem chráněné osobní údaje [1]. RÚIAN bude podrobněji popsán v kapitole 2.

2.1.4 Základní registr agend orgánů veřejné moci a některých práv a povinností (Registr práv a povinností, RPP)

Správcem registru je Ministerstvo vnitra. Registr obsahuje údaje o orgánech veřejné moci, a to o agendách, o orgánech veřejné moci, které je vykonávají, o informačních systémech, které pro výkon agend používají a o rozsahu oprávnění přístupu k referenčním údajům. Dále obsahuje referenční údaje o právech a povinnostech osob. To jsou údaje o rozhodnutích, na jejichž základě dochází ke změně referenčním údajů v základních registrech. Registr slouží jako zdroj údajů pro informační systémy základních registrů při řízení přístupu uživatelů k údajům v jednotlivých registrech a AIS. To znamená, že kdykoliv se někdo pokusí získat z registrů nějaký údaj nebo ho dokonce změnit, bude systém posuzovat, zda to bude dovolené a jestli má na to právo ze zákona. Díky tomuto registru má každý občan možnost se dozvědět, kdo, kdy a za jakým účelem o něm vedená data v základních registrech měnil nebo upravoval [1].

3 RÚIAN

Správcem registru územní identifikace je Český úřad zeměměřický a katastrální. RÚIAN je veřejný seznam. Jeho data jsou přístupná pomocí aplikace Veřejný dálkový přístup (VDP), jež umožňuje prohlížení dat nebo stahování dat ve výměnném formátu RÚIAN (VFR). Formát VFR je textový formát typu XML. Více o VDP v [3], [5].

RÚIAN je veden v souladu se dvěma právními předpisy, kterými jsou Zákon o základních registrech [2] a Vyhláška o základním registru územní identifikace, adres a nemovitostí [6].

3.1 Údaje vedené v RÚIAN

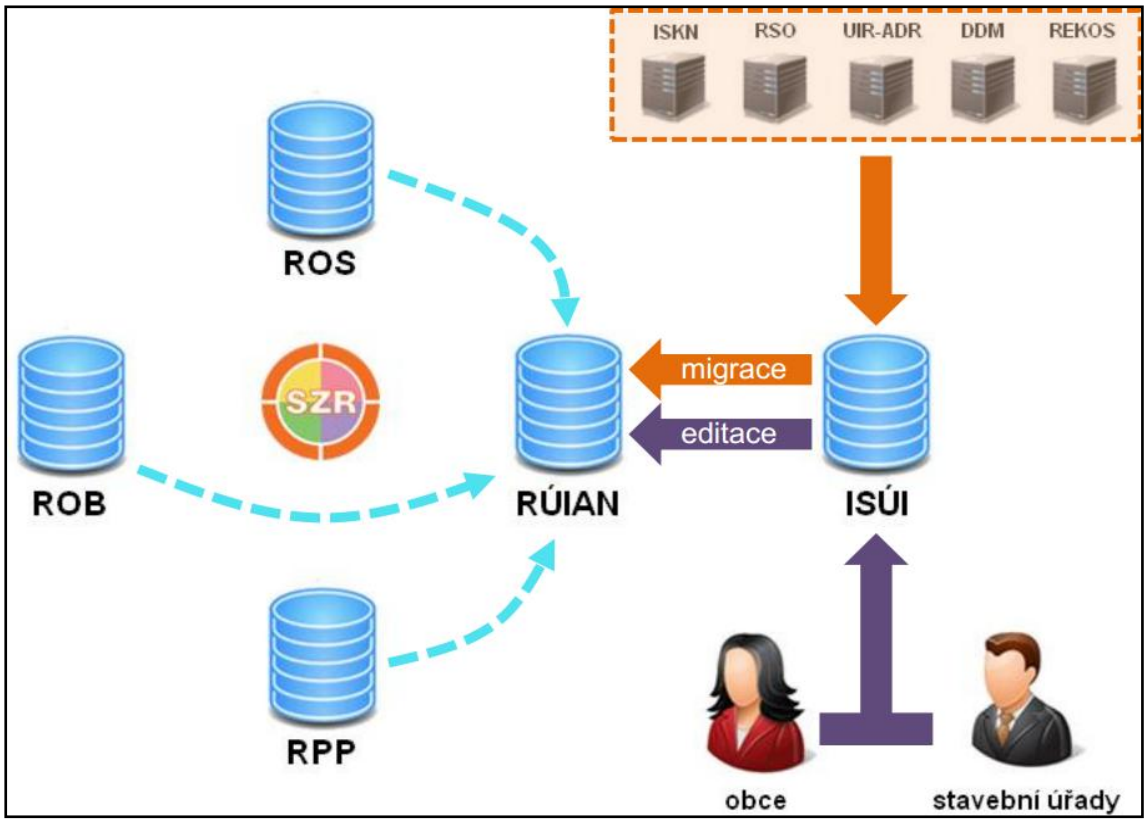
RÚIAN eviduje údaje o základních územních prvcích, kterými jsou území státu, území regionu soudržnosti podle jiného právního předpisu, území vyššího územního samosprávného celku, území kraje, území okresu, správní obvod obce s rozšířenou působností (ORP), správní obvod obce s pověřeným obecním úřadem, území obce, území vojenského újezdu, správní obvod v hlavním městě Praze, území městského obvodu v hlavním městě Praze, území městské části v hlavním městě Praze, území městského obvodu a městské části územně členěného statutárního města, katastrální území, území základní sídelní jednotky, stavební objekt, adresní místo a pozemek v podobě parcely. Registr územní identifikace dále obsahuje údaje o územně evidenčních jednotkách, kterými jsou část obce a ulice nebo jiné veřejné prostranství.

O územních prvcích RÚIAN eviduje identifikační údaje, kterými jsou kód a název, lokalizační údaje (definiční bod, případně hranice) a údaje o vazbách na jiné územní prvky, případně na územně evidenční jednotky. O každé územně evidenční jednotce se v registru územní identifikace vedou identifikační údaje (kód a název), lokalizační údaje, kterými jsou u části obce definiční bod a u ulice definiční čára a údaje o vazbách na územní prvky. Identifikačními údaji

pro pozemek jsou kód a název katastrálního území, ve kterém pozemek leží, a parcelní číslo [2].

3.2 Zdroje RÚIAN a jeho editace

Prvotní naplnění databáze RÚIAN proběhlo pomocí importu dat Informačního systému katastru nemovitostí (ISKN), který spravuje ČÚZK. Dalším zdrojem je Registr sčítacích obvodů a budov (RSO), spravovaný ČSÚ, ze kterého pocházejí některé stavební objekty a základní sídelní jednotky. Územně identifikační registr adres (ÚIR-ADR), který spravuje Ministerstvo práce a sociálních věcí, je základním zdrojem adresních údajů. Z Registru komunálních symbolů (REKOS), který vede Poslanecká sněmovna Parlamentu ČR, byly do RÚIAN vloženy erby a vlajky územních samospráv. Další data dodaly například obce a stavební úřady nebo Databáze dodacích míst České pošty (DDM) [7].

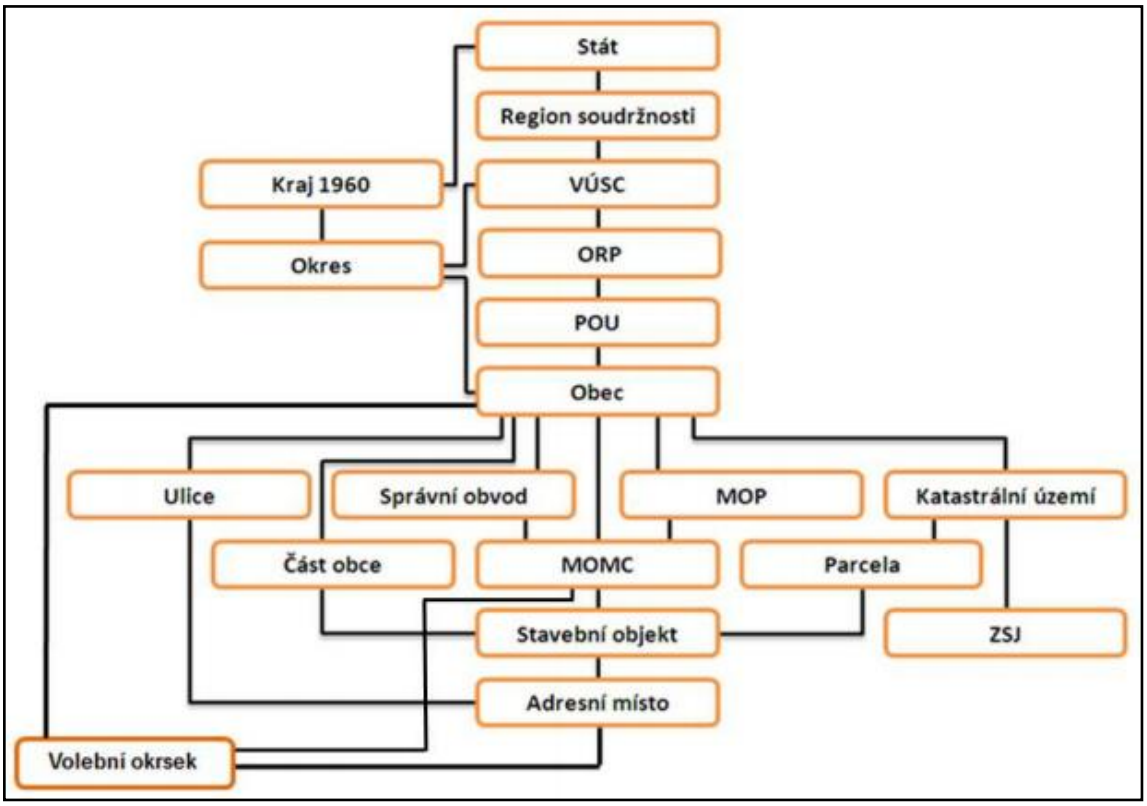


Obr. 2: Zdroje a naplnění databáze RÚIAN [41].

Základními AIS, které umožní editaci dat v RÚIAN, jsou ISKN a Informační systém územní identifikace (ISÚI). ISÚI je nový informační systém, který spravuje všechna data, která RÚIAN podle zákona musí vést, ale která nejsou obsažena v ISKN. Jedná se zejména o územní identifikaci a adresy. Po prvotním naplnění jsou data aktualizována přes editační rozhraní. Editorem údajů ISKN je ČÚZK, editorem údajů ISÚI je ČÚZK, ČSÚ, obce a stavební úřady [7]. Proces naplnění RÚIAN daty a jejich aktualizace je zobrazen na obrázku 2.

3.3 Struktura RÚIAN

Data RÚIAN jsou spravována pomocí databázového systému Oracle. Struktura datového modelu, ve kterém jsou data uložena, je znázorněna na obrázku 3.



Obr. 3: Struktura datového modelu RÚIAN [42].

Databáze RÚIAN je rozsáhlá, její velikost je 250 GB, aktuálně je využito cca 190 GB [8]. Počty některých prvků vedených v RÚIAN jsou uvedeny v tabulce 1. Data jsou aktuální k 27. 4. 2015.

Tabulka 1: Počty prvků evidovaných v RÚIAN.

Prvek	Počet	Prvek	Počet
Obce	6 253	**Adresní místa**	2 899 695
Části obcí	15 072	**Parcely**	21 728 625
Katastrální území	13 094	**Základní sídelní jednotky**	22 429
Stavební objekty	4 075 916	**Ulice**	81 211

3.4 Stavební objekty v RÚIAN

Prvotním zdrojem stavebních objektů pro RÚIAN byly ISKN a RSO. Stavební objekty jsou dále doplňovány stavebními úřady. Zde ale dochází ke konfliktu, neboť legislativa stavebních úřadů, katastru nemovitostí a RÚIAN se řídí různými právními předpisy. Tyto předpisy se totiž liší terminologií i vymezením objektů, které zákon o základních registrech označuje jako stavební objekty.

3.4.1 Stavební objekt v zákoně o základních registrech

Podle Zákona o základních registrech [2] je stavebním objektem zapisovaným do RÚIAN dokončená budova zapisovaná do katastru nemovitostí České republiky nebo jiná dokončená stavba, která se do katastru nemovitostí nezapisuje, ale bylo jí přiděleno číslo popisné nebo evidenční, pokud slouží k ubytování lidí nebo k podnikání nebo jiné ekonomické činnosti. Identifikačními údaji o stavebním objektu jsou identifikační údaje pozemku, na kterém je stavební objekt postaven a údaj o jeho čísle popisném nebo evidenčním, pokud se přiděluje, a údaj o části obce, ke které stavební objekt přísluší, nebo údaj o tom, že se jedná o stavební objekt, kterému se popisné ani evidenční číslo nepřiděluje. O stavebním objektu se dále vedou údaje o typu stavebního objektu, způsobu jeho využití a o typu a způsobu jeho ochrany a technickoekonomické atributy tohoto objektu, kterými jsou měsíc a rok dokončení, počet bytů u stavebního objektu s byty, zastavěná plocha v m^2, obestavěný prostor v m^3, podlahová plocha v m^2, počet

nadzemních a podzemních podlaží, druh svislé nosné konstrukce, připojení na vodovod, připojení na kanalizační síť, připojení na rozvod plynu, způsob vytápění a vybavení výtahem [2].

V ISÚI jsou vedeny údaje o stavebních objektech evidovaných v RÚIAN, a to ty údaje, které nejsou vedeny v ISKN. Nad rámec RÚIAN je v ISÚI veden další technickoekonomický atribut, a to měsíc a rok odstranění stavebního objektu.

Identifikační údaje stavebního objektu, definiční bod, typ stavebního objektu a technickoekonomické atributy zapisuje do RÚIAN stavební úřad. V případě, že objekt nevyžaduje stavební povolení nebo ohlášení, nahrazuje funkci stavebního úřadu obec. Kód stavebnímu objektu přiděluje ČÚZK. ČÚZK dále zapisuje údaje o hranici stavebního objektu a údaje o pozemku. Změny stavebních objektů v RÚIAN mohou být provedeny na základě kolaudačního rozhodnutí a zpracování oznámení o dokončení/užívání stavby, případně odstranění stavby. Postup při zápisu těchto prvků je stanoven ve Vyhlášce o základním registru územní identifikace, adres a nemovitostí [6], povinnost zápisu konkrétních údajů o stavebních objektech rozděluje mezi jednotlivé orgány Zákon o základních registrech [2].

Z katastrálního zákona vychází i ČSÚ a budovy v RSO jsou tedy chápány ve smyslu tohoto právního předpisu.

3.4.2 Budova v katastrálním zákoně

Zákon o katastru nemovitostí (katastrální zákon) [9] na rozdíl od Zákona o základních registrech o stavebních objektech nemluví. Prvkem obdobným stavebnímu objektu v RÚIAN, který je evidovaný v ISKN, je budova. Budovou se podle katastrálního zákona rozumí nadzemní stavba spojená se zemí pevným základem, která je prostorově soustředěna a navenek převážně uzavřena obvodovými stěnami a střešní konstrukcí. Katastr nemovitostí (KN) eviduje budovy s číslem popisným nebo evidenčním, pokud nejsou součástí pozemku, budovy, kterým se číslo popisné ani evidenční nepřiděluje, pokud nejsou součástí pozemku nebo práva stavby a jsou hlavní stavbou na pozemku. KN neeviduje

drobné stavby. Obvod budovy, která je hlavní stavbou a je součástí pozemku nebo práva stavby, je obsažen v polohopisu katastrální mapy. Ten zároveň obsahuje i obvody vedlejších staveb [9].

Údaje, které jsou evidovány v KN o budovách, jsou geometrické a polohové určení, číslo popisné nebo evidenční, údaj o tom, zda se jedná o dočasnou stavbu a vybrané údaje o způsobu využití a ochraně nemovitosti. Tyto údaje blíže specifikuje Vyhláška o katastru nemovitostí (katastrální vyhláška) [10]. Vklad a editaci budov v ISKN provádí výhradně katastrální úřady na základě geometrického plánu a dokladu o způsobu užívání budovy.

3.4.3 Stavba ve stavebním zákoně

Podle stavebního zákona [11] se stavbou rozumí veškerá stavební díla, která vznikají stavební nebo montážní technologií, bez zřetele na jejich stavebně technické provedení, použité stavební výrobky, materiály a konstrukce, na účel využití a dobu trvání. Za stavbu se považuje také výrobek plnící funkci stavby. Stavba, která slouží reklamním účelům, je stavba pro reklamu. Pro účely stavebního zákona se stavbou rozumí i její část nebo změna.

Stavby mohou vznikat na základě stavebního povolení, ohlášení stavby, ale i bez stavebního povolení a ohlášení. U staveb, které vyžadují stavební povolení, případně ohlášení stavebnímu úřadu, musí stavební úřady zapisovat jejich vznik, zánik nebo změnu do ISÚI a editovat tak stavební objekt vedený v ISÚI.

3.4.4. Porovnání pojmů budova, stavební objekt a stavba

Jaký je tedy rozdíl mezi budovou, stavebním objektem a stavbou? Stavba je jakékoli stavební dílo bez ohledu na jeho umístění a způsob využití. Budova je nadzemní stavba, která je prostorově uzavřena a má střechu. Obecně platí, že stavby určené pro bydlení lidí jsou vždy budovami. Stavebním objektem jsou pak budovy zapisované do katastru nemovitostí nebo jiné budovy s číslem popisným nebo s číslem evidenčním, pokud slouží k bydlení nebo podnikání. Tyto pojmy je tedy

možné hierarchicky seřadit. Budova je pojem nejspecifičtější, stavební objekt pak rozšiřuje budovy o další objekty a nejobecnějším pojmem jsou stavby.

Z výše uvedeného vyplývá, že ne všechny stavební objekty vedené v RÚIAN jsou evidovány v ISKN jako budovy. Naopak všechny budovy evidované v ISKN by měly být evidovány i v množině stavebních objektů RÚIAN. Mezi stavební objekty evidované v RÚIAN a neevidované v ISKN patří například podzemní stavby, které mají číslo popisné nebo číslo evidenční. To můžou být například stánky nebo obchody v podchodech nebo v prostorách metra, mobilní domy nebo energetické domy, které jsou pod povrchem. RÚIAN na rozdíl od ISKN neeviduje rozestavěné budovy.

Zajímavým nesouladem v ISKN a zároveň v RÚIAN je evidence chladicích věží jaderných elektráren. V České republice existují dvě jaderné elektrárny, Jaderná elektrárna Temelín v katastrálním území Křtěnov a Jaderná elektrárna Dukovany v katastrálním území Skryje nad Vltavou. Parcely, na kterých stojí chladicí věže elektrárny Temelín, jsou v ISKN evidovány s druhem pozemku *ostatní plocha* a není na nich evidovaná žádná budova. Parcely, na kterých stojí chladicí věže elektrárny Dukovany, jsou v ISKN evidovány s druhem pozemku *zastavěná plocha a nádvoří*. Součástmi těchto parcel jsou budovy bez čísla popisného nebo evidenčního, kterými jsou chladicí věže. Chladicí věže ale nemají střešní konstrukci, takže by podle katastrálního zákona budovou být neměly. Tento nesoulad se dále promítá i do RÚIAN, kde chladicí věže Dukovan jsou vedeny jako stavební objekty bez čísla popisného nebo evidenčního, ale chladicí věže Temelína zde evidovány nejsou.

3.5 Kontroly RÚIAN

Kvalita a správnost dat vedených v RÚIAN je průběžně kontrolována a z RÚIAN jsou postupně systematicky odstraňovány chyby a nepřesnosti. Naprostou většinu

těchto kontrol a oprav iniciuje ČÚZK. Kontroly, které doposud provedl, jsou zveřejněny na webových stránkách ČÚZK[1].

Na základě těchto prováděných kontrol jsou vytvářeny chybové a rozdílové sestavy, jež jsou dále předávány editorům RÚIAN spolu s návody, jak postupovat při vyhodnocování sestav a opravě chyb. Editoři na základě sestav posoudí, zda se v jednotlivých případech jedná o chyby či nikoliv a případně je odstraní.

Namátkově mohou chyby v RÚIAN odhalit orgány veřejné moci, jež mají povinnost nesoulady vedených údajů a skutečného stavu neprodleně ohlásit přes rozhraní ISZR editorům příslušných údajů. Pro hlášení chyb od externích uživatelů byla nově zřízena webová aplikace Interní reklamace správce registru. Tato služba umožňuje příjem reklamací pro prvky adresní místo, stavební objekt a ulice. U stavebních objektů je možné zaslat reklamaci pro doplnění nebo zrušení stavebního objektu, změnu čísla domovního, městské části, části obce, definičního bodu, parcely, změnu způsobu využití, hranice a duplicitně vedené stavební objekty. Reklamace od externích uživatelů zasílá ČÚZK příslušnému editorovi, který ji vyřídí změnou, doplněním údajů, případně zamítnutím. Externí žadatel, který reklamaci zadal, je o průběhu jejího řešení informován prostřednictvím e-mailu, který vyplnil do formuláře [12].

ČÚZK dále spolupracuje s akademickým prostředím a při zvyšování kvality dat ISKN a RÚIAN pomáhají studenti vysokých škol v rámci svých závěrečných prací. Jednou z takových prací, která vznikla ve spolupráci s ČÚZK, je disertační práce pana Karla Janečky, Modelování konzistentní báze geodat na úrovni datového modelu katastru nemovitostí, z roku 2009 [13]. Dalším příkladem je disertační práce Jiřího Bartoše Geodata a metadata ISKN v prostředí INSPIRE [14] z roku 2010, jež vznikla ve spolupráci s ČÚZK a zabývá se mimo jiné zvyšováním kvality dat ISKN. Zde je zmiňován problém duplicitních bodů v ISKN omezený na totožné body.

[1] http://www.cuzk.cz/Uvod/Produkty-a-sluzby/RUIAN/3-Overeni-uplnosti-a-spravnosti-udaju-ISUI-RUIAN/Kontroly-dat-ISUI-RUIAN.aspx

V roce 2014 vznikly dvě diplomové práce zabývající se kvalitou dat RÚIAN. Jednou z nich je Kontrola vybraných územních prvků a evidenčních jednotek v RÚIAN [3]. Autorka Jana Hejdová se v ní zabývá adresními místy a ulicemi v RÚIAN a řeší kontrolu vazeb adresních míst na ulice a jejich odlehlosti, kontrolu duplicitních a velmi blízkých adresních míst a kontrolu nespojitosti ulic. Tyto kontroly jsou řešeny v prostředí Oracle Spatial pomocí procedurálního jazyka PL/SQL. Druhou prací je diplomová práce Simony Karochové, Analýza uliční sítě v obcích České republiky [15]. Autorka zde podrobně analyzuje uliční síť na datech RÚIAN v prostředí ArcGIS.

4 Příprava dat pro praktické řešení kontrol

K testování vytvořených postupů jsou použita data RÚIAN, stažená ve formátu VFR pomocí aplikace Veřejný dálkový přístup[2]. Data jsou aktuální k 28. 2. 2015. Data byla pomocí nástroje VFR Import Tool[3] 10.3.10 nejprve naimportována do geodatabáze systému ArcGIS[4] a odtud pak byly jednotlivé prvkové třídy exportovány do souborů formátu shapefile.

Vyexportované shapefile soubory byly následně pomocí nástroje GeoRaptor[5] verze 3.2.3.0470 importovány do databáze Oracle, verze 11g Enterprise Edition Release 11.2.0.2.0, GeoRaptor však nelze přidat jako doplněk pro správu prostorových dat do použité verze vývojového prostředí Oracle SQL Developer 4.0.0.12. Při použití GeoRaptoru pro načtení souborů shapefile bylo tedy nutné použít nižší verzi Oracle SQL Developer 3.2.10.09, se kterou je GeoRaptor kompatibilní.

Pro uložení dat v databázi Oracle byl vytvořen datový model, založený na datovém modelu RÚIAN (obrázek 3) [3].

Další možností importu dat do databáze je využití možností knihovny GDAL. Podrobný postup importu VFR souborů do databáze Oracle pomocí knihovny GDAL je uveden v [16]. K importu dat lze použít i komerční software, například program FME Desktop[6], jedná se o software určený ke konverzi dat mezi formáty. Systém řízení báze dat (Database management systems, DBMS) Oracle je pro realizaci kontrol zvolen proto, že tento systém používá ČÚZK pro správu dat RÚIAN. Výsledné kontroly jsou implementovány jako procedury v jazyce PL/SQL, což je procedurální nadstavba dotazovacího jazyka SQL.

Pro testování kontrol jsou použity dvě datové sady. První z nich je použita pro ladění kontrol. Jedná se o data ORP Plzeň. Počty vybraných prvků RÚIAN v této

[2] http://vdp.cuzk.cz/
[3] http://www.arcdata.cz/produkty-a-sluzby/software/arcdata-praha/vfr-import
[4] http://www.esri.com/software/arcgis/arcgis-for-desktop
[5] http://sourceforge.net/projects/georaptor/
[6] https://www.safe.com/fme/fme-desktop/

testovací sadě jsou uvedeny v tabulce 2. Data byla stažena v patnácti VFR souborech a velikost těchto souborů je 296 MB. Tabulka 2 dále uvádí, kolik procent celkových dat RÚIAN testovací data obsahují. Výsledky testování jednotlivých kontrol na této testovací sadě budou uvedeny u každé kontroly.

Tabulka 2: Počty vybraných objektů v testovacích datech ORP Plzeň.

PRVEK	POČET PRVKŮ	% Z CELKOVÝCH DAT
Stavební objekt	48 413	1,2
Parcela	148 247	0,7
Adresní místa	33 522	1,1
Obec	15	0,2
Část obce	46	0,3
Katastrální území	42	0,3

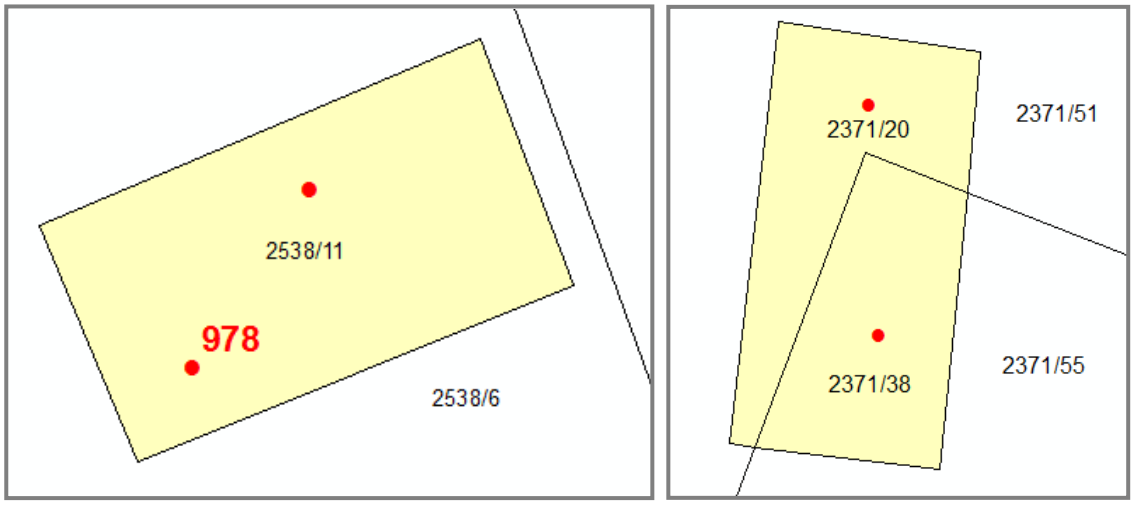
Druhá testovací sada obsahuje data všech krajských měst v ČR. Velikost VFR souborů, jež obsahují tato data, je 2,98 GB. Počty vybraných prvků v těchto datech a informace o tom, o jakou část dat RÚIAN se jedná, jsou uvedeny v tabulce 3. Nad touto testovací sadou jsou testovány pouze výsledné kontroly. Testování vytvořených kontrol na této testovací sadě je diskutováno v kapitole 9, kde jsou rovněž popsány výsledky jednotlivých kontrol a počty prvků, které byly kontrolou označeny za chybné.

Tabulka 3: Počty vybraných prvků v testovacích datech krajských měst.

PRVEK	POČET PRVKŮ	% Z CELKOVÝCH DAT
Stavební objekt	498 915	12,2
Parcela	1 372 476	6,6
Adresní místa	365 890	12,6
Obec	13	0,2
Část obce	405	2,7
Katastrální území	400	3,1

5 Vyhledání stavebních objektů, u kterých jsou blízké definiční body

Úkolem této kontroly je nalezení stavebních objektů, jež jsou v RÚIAN vedeny duplicitně. To znamená, že pro jeden reálný stavební objekt jsou v databázi dva či více záznamů. Jde tedy o nalezení duplicitních definičních bodů stavebních objektů a to nejen těch bodů, jež mají totožné souřadnice, ale i definičních bodů, které jsou příliš blízké. Příklad hledaných situací je na obrázku 4. Na obrázku jsou žlutou barvou zvýrazněny stavební objekty, červené body jsou definiční body stavebních objektů a červená čísla jsou čísla domovní. Černá čísla jsou pak parcelní čísla. Je zde znázorněn případ dvou blízkých definičních bodů, jež oba mají vazbu na jednu parcelu KN. Jeden evidovaný stavební objekt má číslo popisné, druhý je bez čísla. Reálně se ale jedná o jediný stavební objekt. Ve druhém případě stojí jeden stavební objekt na dvou parcelách KN a pro každou parcelu je objekt evidován, pokaždé bez čísla popisného nebo evidenčního.



Obr. 4: Příklady chyb, které jsou hledány kontrolou pro vyhledání stavebních objektů, u kterých jsou totožné definiční body. Vlevo: definiční body dvou stavebních objektů (jeden s číslem domovním, druhý bez čísla domovního), oba leží uvnitř polygonu jednoho stavebního objektu. Vpravo: stavební objekt ležící na dvou parcelách, v polygonu stavebního objektu leží dva definiční body stavebních objektů (oba bez čísla domovního).

Je potřeba vyřešit otázku, jak pouze na základě definičních bodů a jejich vzdáleností určit, zda se v případě dvou blízkých definičních bodů jedná o definiční

body různých stavebních objektů, které stojí blízko u sebe nebo zda se jedná o duplicitní definiční bod. Existují stavební objekty, jež mají zdroj v RSO, případně byly vytvořeny jako fiktivní z UIR-ADR. Zdrojem většiny stavebních objektů v RÚIAN byl ale ISKN. Z definice budovy v ISKN je zřejmé, že z ISKN by se do RÚIAN neměla dostat budova o ploše menší než $16 m^2$. To by znamenalo, že nejmenší evidované budovy se zhruba čtvercovým půdorysem by měly mít rozměr asi 4×4 m. Stavební objekty ale nemusí mít čtvercový tvar. Je tedy zřejmé, že přesnou vzdálenost, kdy se jedná o duplicitu, nelze stanovit. Dá se ale říci, že body do určité vzdálenosti od sebe jsou z duplicity podezřelé. Co se týče umístění definičního bodu, vyhláška o základním registru územní identifikace, adres a nemovitostí doporučuje umístění definičního bodu v těžišti stavebního objektu. To ale platí především u nově vkládaných objektů. U starých stavebních objektů toto pravidlo platit nemusí. Důvodem je nejčastěji to, že došlo k obnově operátu a vrstva katastru se změnila. Na základě tohoto vysvětlení je vytvořena kontrola, kterou je možné spustit pro různé prahové hodnoty. Jako vhodné se jeví provedení kontroly pro prahové hodnoty 2, 4 a 6 metrů, přičemž u 6 metrů se dá předpokládat množství nalezených dvojic blízkých bodů, u kterých se o duplicity nejedná.

Nalezené duplicitní stavební objekty se mohou lišit v různých atributech, například číslo domovní, část obce, způsob využití, počet adresních míst nebo katastrální území. Určení těchto atributů je důležité pro rozhodnutí, zda se skutečně jedná o duplicitní stavební objekt nebo o dva různé stavební objekty, jejichž definiční body jsou umístěny blízko sebe. Příkladem mohou být dva definiční body, vzdálené od sebe 1,5 metru. Oba tyto definiční body mají stejná číslo domovní, typ stavebního objektu, náleží stejnému katastrálnímu území, ale liší se v části obce. Pak je zde velká pravděpodobnost, že se skutečně jedná o duplicitní záznam a že jeden stavební objekt je evidován ve dvou částech obce. Druhým příkladem mohou být dva definiční body vzdálené 1,9 metru. Tyto body náleží stejnému katastrálnímu území a stejné části obce, oba nemají žádné adresní místo a oba jsou bez čísla domovního. Způsob využití obou stavebních objektů je garáž. V tomto případě je pravděpodobné, že se jedná o dva různé stavební

objekty, které stojí u sebe. Garáže jsou totiž typickým příkladem malých stavebních objektů, které často mívají obdélníkový tvar a jsou uspořádány v řadě.

Vyhledání totožných, popřípadě blízkých definičních bodů stavebních objektů, je možné převést na geometrický problém hledání blízkých bodů. Ten tvoří podmnožinu problému hledání nejbližšího souseda. Problém hledání nejbližšího souseda a možnosti jeho řešení, jsou teoreticky popsány v kapitolách 5.1 a 5.2. Vybrané možnosti řešení byly otestovány a porovnány z hlediska časové náročnosti a kvality výstupů. Testování a porovnání metod je popsáno v kapitole 5.3.

5.1 Problém hledání nejbližšího souseda

Obecným problémem, který zahrnuje hledání blízkých bodů, je problém hledání nejbližšího souseda. Hledání nejbližšího souseda (*nearest neighbor search*, NNS), nazývané také *proximity search* [17], *similarity search* [18] nebo *post office problem* [19], je dlouho řešený problém, který spočívá v nalezení objektů s podobnými charakteristikami. Je využíván v mnoha oblastech, jako například data mining, rozpoznávání řeči nebo obrazu, v oblasti genetiky pro nalezení podobných struktur DNA [20] nebo v oblasti výpočetní geometrie pro hledání nejbližších bodů (*closest pair of points problem*, CPP) [21].

5.1.1 NNS v metrickém prostoru

Pro zavedení problému NNS v metrickém prostoru je potřeba tento prostor definovat.

Metrický prostor je dvojice (X, ρ), *kde* X *je množina a* ρ *je reálná funkce na dvojicích prvků z* X, *tj.* $\rho : X \times X \to \mathbb{R}$ *splňující následující vlastnosti.*

1) ρ *je symetrická:* $\rho(x, y) = \rho(y, x)$

2) ρ *je nulová právě tehdy, když* $x = y$: $\rho(x, y) = 0 \equiv x = y$

3) ρ *splňuje trojúhelníkovou nerovnost:* $\rho(x, y) \leq \rho(x, z) + \rho(z, y)$.

Takovou funkci ρ *nazýváme metrikou, vzdálenostní funkcí nebo jen vzdáleností.*

Jedná se tedy o prostor, na kterém lze definovat vzdálenost. Nejpoužívanější metrikou, která je používána i v geografických informačních systémech, je klasická Euklidovská metrika [22].

Problém NNS je v metrickém prostoru definován následovně:

$$NNS(A, B) = \left\{\left(a_i, b_j\right): \forall\, a_i \in A, \exists\, b_j \in B, \exists\, b_k \in B \left\{dist(a_i, b_k) < dist\left(a_i, b_j\right)\right\}\right\},$$

kde A, B jsou dvě datové sady, $dist\,(a_i, b_j)$ počítá metrickou vzdálenost a a_i se označuje jako query point [23].

Jinými slovy, pro každý objekt z množiny A je nalezen nejbližší soused z množiny B. Problém NNS může mít různé modifikace, například vyhledávání *k* nejbližších sousedů (*k-nearest neighbor search*, KNNS) nebo hledání sousedů v pevné vzdálenosti (*fixed-radius near neighbors*, FRNN).

5.1.2 FRNN v metrickém prostoru

Definice problému FRNN v metrickém prostoru je následující:

$$FRNN(A, B) = \{r \in \mathbb{R}, \left(a_i, b_j\right): \forall\, a_i \in A, \forall\, b_j \in B, \{dist(a_i, b_j) \leq r\}\},$$

kde A, B jsou dvě datové sady a $dist(a_i, b_j)$ počítá metrickou vzdálenost a a_i se označuje jako query point.

Hledají se tedy všechny body, které jsou od query pointu do vzdálenosti menší než je zadaná vzdálenost *r*. Ve 2D euklidovském prostoru pak platí:

$$dist(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}, \qquad kde\ a = (a_1, a_2), \qquad b = (b_1, b_2)$$

Na problém FRNN ve 2D, případně 3D euklidovském prostoru narážíme při zpracování geografických dat a prostorových analýzách. Jedná se o hledání blízkých bodů, tedy modifikaci CPP problému. Úkolem je vyhledání dvojic bodů, jejichž vzdálenost je nižší než zadaná hodnota. Právě řešením tohoto problému ve 2D se bude zabývat následující část.

5.2 Řešení NNS ve 2D euklidovském prostoru

Příklady řešení problému NNS (i FRNN a CPP) jsou popsány například v [17], [19], [20], [21], [23], [24]. Vybrané postupy řešení jsou popsané v kapitolách 5.2.1 - 5.2.5. Kapitola 5.2.6 obsahuje shrnutí vybraných metod.

5.2.1 Použití hrubé síly

Nejjednodušší metoda řešení NNS a FRNN problému je použití hrubé síly (*brute force method, naive method, linear search*). Metoda pro NNS funguje tak, že pro každý query point z množiny A spočítá vzdálenost ke všem bodům z množiny B a při tom si pamatuje bod, ke kterému je vzdálenost nejkratší a případně velikost vzdálenosti. Modifikace pro FRNN je taková, že algoritmus spočte opět všechny vzdálenosti a pamatuje si všechny dvojice bodů, mezi kterými byly vzdálenosti nižší než daná hodnota. Algoritmus hrubé síly je časově a výpočetně náročný, jeho výpočetní složitost je $O(n^2)$. Algoritmus je tedy použitelný pouze pro malá data [25].

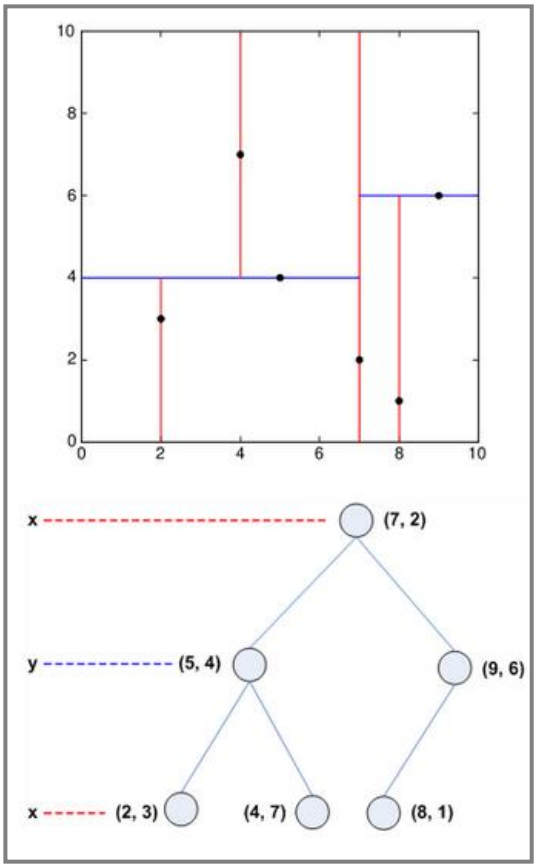
5.2.2 Použití prostorových indexů

Další možností řešení problému je použití prostorových indexů založených na stromových strukturách. Používanými indexy jsou například k-d strom (2-D strom), R-tree, R*tree. Složitost vyhledávání ve stromových strukturách bývá většinou $O(\log n)$ [18].

K-d strom

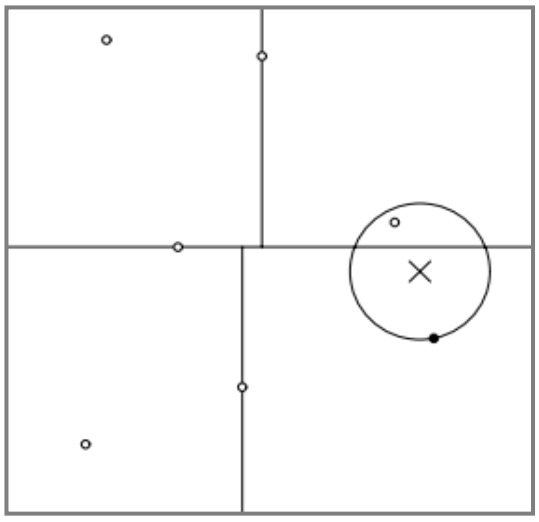
K-d strom je binární strom. Používá se pro ukládání k-dimenzionálních dat. Zúžením k-d stromu na 2D vzniká 2-D strom. Vybudování této datové struktury pro bodová data funguje tak, že body jsou seřazeny nejprve podle souřadnice X a je vybrán medián. Vybraným mediánem je vedena přímka, která dělí rovinu na dvě části, dva podstromy. Každý takto vzniklý podstrom obsahuje stejný počet bodů (v případě sudého počtu prvků leží v jedné polorovině o jeden bod více).

V následujícím kroku jsou body každého podstromu seřazeny podle souřadnice Y, je opět vybrán medián a vybraný medián je vrcholem dalšího vzniklého podstromu. Tento postup se rekurzivně opakuje pro levý i pravý podstrom až do okamžiku, kdy po rozdělení zůstane v každé skupině jeden prvek, jež se poté umístí na pozici listu stromu. Průměrná složitost vytvoření 2-D stromu je $O(\log n)$. Ukázka bodového datasetu, rozděleného pomocí 2-D stromu a výsledný 2-D strom jsou na obrázku 5 [26], [27].



Obr. 5: Ukázka 2-D stromu, který je vytvořen nad 6 body v rovině. Mediánem je bod o souřadnicích [7,2] [40].

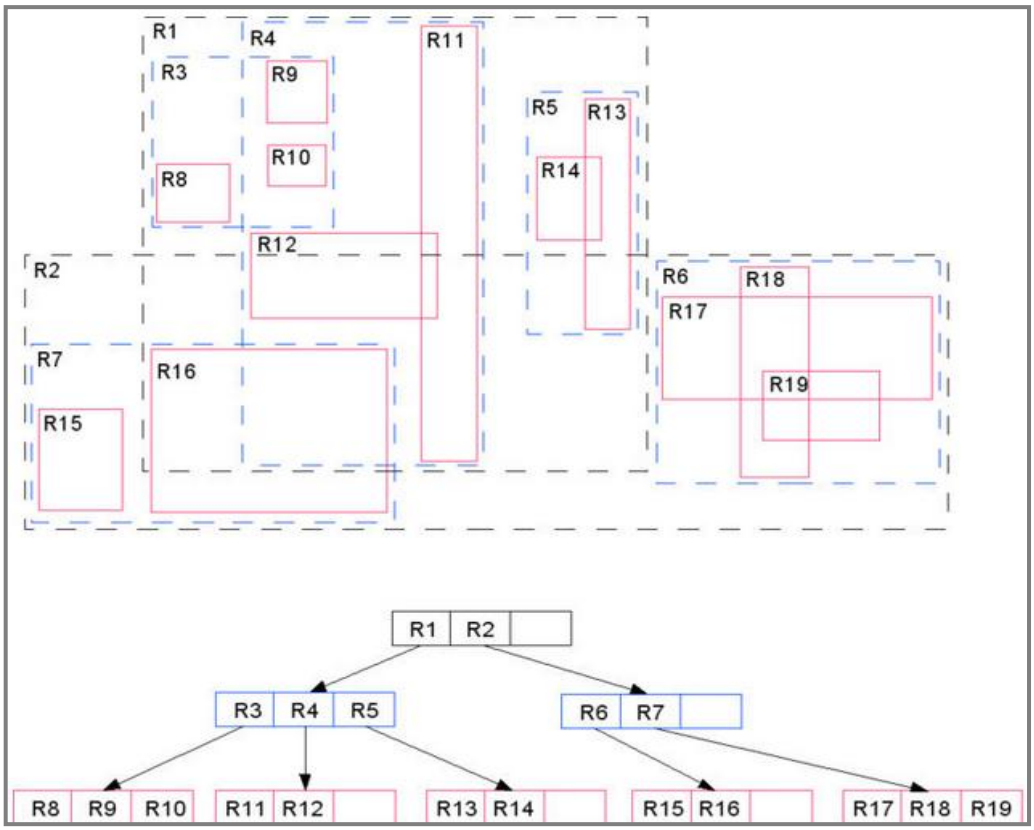
Algoritmus vyhledání nejbližšího souseda je znázorněn na obrázku 6. *Query point*, pro který je hledán nejbližší soused, je označen křížkem. V prvním kroku je nalezen listový bod oblasti, ve které leží zdrojový bod. To nemusí být nutně nejbližší soused, ale je zřejmé, že pokud by existoval ještě bližší bod, bude ležet uvnitř kružnice, která má střed ve *query pointu* a prochází listovým bodem oblasti, ve které leží zdrojový bod. Stačí tedy zkontrolovat ty listy, do jejichž oblastí zasahuje vytvořená kružnice. Pokud by oblast, ve které leží *query point*, neobsahovala listový bod, jako nejbližší by se vybral předek oblasti, ve které leží zdrojový bod. Celý algoritmus je podrobně popsán v [27]. V případě řešení


Obr. 6: Vyhledání nejbližších bodů ve 2D stromu. *Query point* je označen křížkem, listový bod oblasti, ve které leží query point, je označen černým bodem [28].

problému FRNN je modifikace algoritmu zřejmá. Kolem *query pointu* se vytváří kružnice o daném poloměru a opět se kontrolují listy, do jejichž oblastí kružnice zasahuje.

R-tree

R-tree je vyvážený binární strom. Vychází původně z B-stromu, ale liší se tím, že pro indexaci využívá vícerozměrnou informaci (pro geografická data dvourozměrnou). Každý uzel R-tree obsahuje odkazy na několik potomků, jejichž počet by se měl pohybovat mezi *m* a *M*, pro které platí $m<M/2$. Uzly, které nejsou listy, obsahují *bounding box* (minimální ohraničující obdélník) všech svých potomků. Oblasti jednotlivých uzlů se mohou překrývat. R-tree se používá k indexaci v mnoha databázových systémech, jedním z nich je i Oracle. Ukázka R-tree je na obrázku 7 [28].



Obr. 7: Ukázka R-tree. R-tree je sestaven nad dvanácti objekty v rovině [40].

Variantou R-tree je R*-tree. Jejich rozdíl je pouze v tom, že R*-tree používá více optimalizačních kritérií.

Algoritmus pro nalezení nejbližšího souseda pomocí R-tree začíná u kořene a hierarchicky postupuje nižšími úrovněmi stromu. Pro všechny potomky kořene je spočtena vzdálenost do query pointu (nejkratší vzdálenost mezi bodem a obdélníkem). Je vybrán ten podstrom, ke kterému je vzdálenost nejkratší. Algoritmus se opakuje pro vybraný podstrom, dokud nedojde k listům. Následně je pro všechny vybrané listy spočtena vzdálenost od query pointu a je vybrán nejbližší bod. Tento postup a jeho možnosti je podrobně popsán v [28]. Modifikace pro řešení FRNN problému je opět zřejmá, algoritmus kontroluje všechny podstromy, jejichž vzdálenost od query pointu je nižší než daná hodnota.

Cover tree

Cover tree pro množinu bodů S v rovině je hierarchická stromová struktura, která ve své nejvyšší úrovni obsahuje kořenový bod a v nejnižší úrovni obsahuje všechny body vstupní množiny S. Každá úroveň cover tree je tzv. „cover“ pro nižší úroveň. Každá úroveň je indexována celočíselnou hodnotou i, která klesá směrem od kořene k listům. Platí, že každý uzel stromu je spojen s bodem množiny S. Každý bod množiny S může být spojen s více uzly, ale uzel spojený s daným bodem se může vyskytovat pouze jednou v každé úrovni. Označme C_i množinu bodů v S, jež se nacházejí v úrovni *i.* Pak pro cover tree platí následující invarianty:

- $C_i \in C_{i-1}$. To znamená, že pokud je bod $p \in S$ spojen s uzlem obsaženým v úrovni C_i, pak musí každá nižší úroveň stromu obsahovat uzel spojený s bodem p
- Pro každý bod $p \in C_i$ existuje bod $q \in C_i$ takový, že platí $d(p,q) < 2^i$ a uzel v úrovni i spojený s bodem q je rodičem uzlu v úrovni $i-1$, který je spojený s bodem p.
- Pro každé $p, q \in C^i, p \neq q$ platí $d(p,q) > 2^i$.

Složitost vytvoření této stromové struktury je $O(c^6 n \ln n)$, kde c je konstanta expanze vstupní množiny S, závisí na její dimenzi. Ve 2D je možné složitost zjednodušit na $O(n \ln n)$ [29].

K nalezení nejbližšího souseda bodu p pomocí cover tree je potřeba procházet stromem sestupně, úroveň po úrovni a pamatovat si množinu uzlů $Q_i \in C_i$, které mohou mít nejbližšího souseda bodu p za potomka. Algoritmus iterativně vytváří množinu bodů Q_{i-1}tak, že rozšíří Q_i o potomky z úrovně C_{i-1}. Nejmenší vzdálenost bodu p od Q je $d(p, Q) = min_{q \in Q} d(p, q)$. Z Q_{i-1} jsou poté odstraněny všechny body q, které nemohou vést k nejbližšímu sousedovi bodu p, to znamená všechny body, které nesplňují podmínku nejmenší vzdálenosti $d(p, Q_i)$. Algoritmus končí v poslední úrovni stromu, kdy v množině Q zůstane nejbližší bod bodu p. Složitost algoritmu ve 2D můžeme zjednodušit na $O(\ln n)$ [29].

5.2.3 Locality sensitive hashing

Hashování je transformace dat pomocí hash funkce na adresu v tabulce, kde je uložen záznam. Hashovací tabulka je kombinací dvou přístupů, vyhledávání pomocí indexu, které má nízkou složitost, a vyhledávání v seznamu, které má nízké nároky na paměť. Hashovací tabulky ukládají indexy a pro každý index je položkou tabulky lineární seznam, do kterého se ukládají položky se stejným indexem.

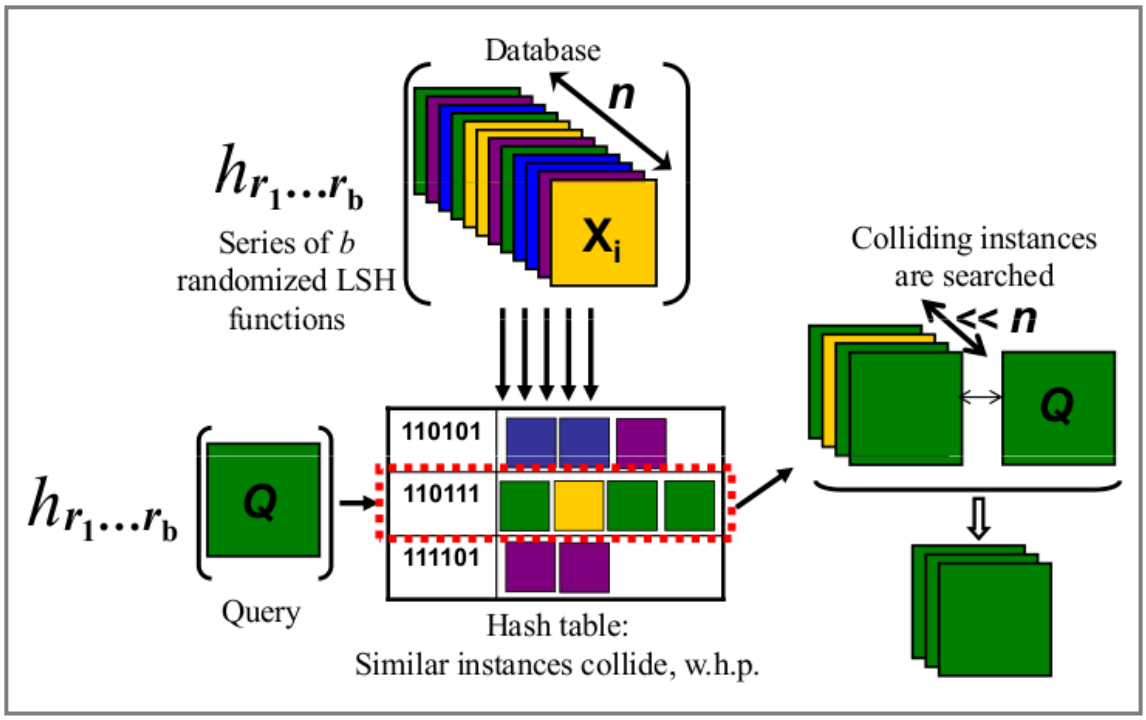
Funkce locality sensitive hashing (LSH) předpokládají navíc to, že prvky, které si jsou blízké, budou mít po aplikaci LSH funkce s větší pravděpodobností stejný index v hashovací tabulce než prvky, které jsou vzdálené. To znamená, že blízké body budou uloženy v hashovací tabulce v jednom řádku v jednom lineárním seznamu (dojde ke kolizi). Základní metoda LSH funguje následovně:

- pro přirozené číslo K definujeme množinu funkcí $G = \{g: S \rightarrow U^K\}$ a pro $g \in G, g(v) = (h_1(v), h_2(v) \ldots, h_K(v))$, kde h jsou příslušné hash funkce, S je množina vstupních bodů, v je bodem S,
- pro přirozené číslo L, které definuje počet skupin hash funkcí $g_1, \ldots, g_L$ z G. Každá funkce g_i je použita k vytvoření jedné hashovací tabulky,

- použitím vzorce $h_{a,b}(v) = \left[\frac{a \cdot v + b}{W}\right]$ sestrojíme L hashovacích tabulek pomocí K LSH funkcí. W je velikost seznamu, $b \in [0, W]$ a a je d-dimenzionální náhodný vektor.

Optimální volba parametrů K, L závisí na vzdálenosti nejbližších bodů ve vstupních datech [18]. Pro lepší představu je postup LSH pro jednu skupinu LSH funkcí na obrázku 8.

Po vytvoření hashovacích tabulek, ve kterých jsou kandidáti na blízké sousedy uloženy vždy pod jedním indexem (v jednom lineárním seznamu), následuje vyhledání nejbližších sousedů bodu q, které je provedeno ve dvou krocích. Nejprve jsou vyhledány všechny seznamy, ve kterých je q zatříděn. Druhým krokem je seřazení všech bodů z prvního kroku podle jejich vzdálenosti ke q a výběr nejbližších bodů. Tato metoda dokáže vyhledávat nejbližšího souseda se složitostí $O(1)$ [18].

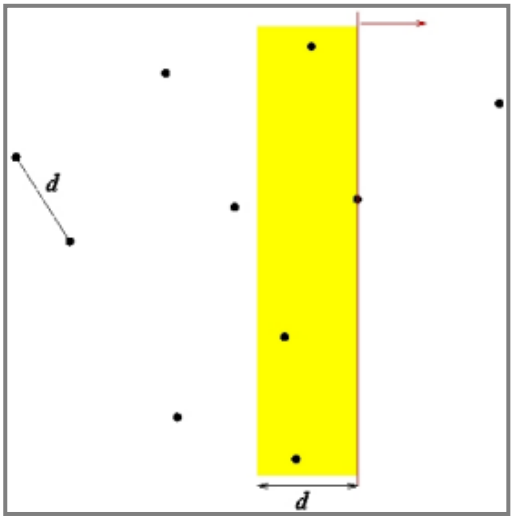


Obr. 8: Postup LSH. Vyhledávání pro jednu skupinu LSH funkcí [43].

5.2.4 Sweep-line

Sweep-line je podle [30] jednou z nejoblíbenějších technik používaných při řešení geometrických problémů ve 2D. Myšlenka sweep-line je jednoduchá. Geometrické elementy jsou seřazeny podle jedné ze souřadnic. Rovinou pak prochází svislá linie, která se zastaví v každém bodě, kde dochází k požadované události. Zde je vyřešena část problému a linie se může pohybovat dál [30].

Algoritmus hledání nejbližších bodů pomocí techniky sweep-line je znázorněn na obrázku 9. Funguje tak, že body vstupní množiny jsou seřazeny vzestupně například podle souřadnice X. Linie jimi prochází zleva doprava. V průběhu algoritmu je vždy ukládána informace o dvojici nejbližších bodů z dosud zkontrolované části vstupní množiny, vzdálenost d mezi nejbližšími body a všechny body, které se nacházejí v pásu o šířce d nalevo od linie (tyto body tvoří množinu D). Linie se zastaví v každém bodě p vstupní množiny a algoritmus z množiny D odstraní všechny body, které jsou od p ve větší vzdálenosti než d. Dále vybere z bodů, jež zbyly v D, nejbližší bod k p. Pokud je jeho vzdálenost menší než d, aktualizuje se dvojice nejbližších bodů a d. Po průchodu posledním bodem je nalezena dvojice nejbližších bodů a jejich vzdálenost. Algoritmus sweep-line pracuje se složitostí $O(n \cdot \log n)$ [13].

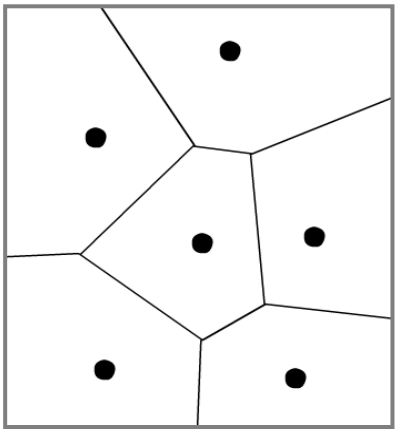


Obr. 9: Nalezení nejbližších bodů pomocí algoritmu sweep-line [13].

5.2.5 Voroného diagram

Voroného diagram ve 2D je způsob dělení roviny v závislosti na množině vstupních bodů, které se nazývají generující body. Pro každý generující bod vzniká oblast, nazývaná Voroného polygon, pro kterou platí, že každý bod této oblasti je ze všech generujících bodů nejblíže právě generujícímu bodu této oblasti. Ukázka Voroného diagramu pro šest generujících bodů je na obrázku 10. Voroného diagramy mají

několik zajímavých vlastností. Jedná se o planární grafy, jejichž každá hrana je sdílena dvěma Voroného polygony. Pro řešení CPP problému je nejdůležitější to, že Voroného polygon je konvexní polygon, pro který platí, že nejbližší soused každého bodu polygonu je generující bod tohoto Voroného polygonu [31].



Obr. 10: Ukázka Voroného diagramu pro množinu šesti bodů.

Řešení problému hledání blízkých bodů pomocí Voroného diagramu je jednoduché, složitější je ale problém konstrukce diagramu. Pro jeho konstrukci existuje řada metod různé složitosti. Jednou z nich je i použití techniky sweep-line, která byla zmíněna v kapitole 5.2.4.

5.2.6 Shrnutí popsaných metod

Jak již bylo řečeno, metoda hrubé síly je pro tak velká data, jako jsou data RÚIAN nevhodná, a to kvůli nutnosti provedení obrovského množství výpočtů a porovnání, a tím neúnosně dlouhé době trvání algoritmu.

Dobrým řešením, z pohledu výpočetní složitosti, je pak použití vhodného prostorového indexu. Zde je nutné zhodnotit spolu s časovou náročností vyhledání nejbližšího bodu i dobu potřebnou pro vytvoření prostorového indexu, popřípadě velikost indexu na disku. V systémech řízení báze dat tyto indexy slouží k urychlení prostorových dotazů. Drobnou nevýhodou indexů je zvětšení objemu dat, a tím větší paměťová náročnost dat. Podobné je to i s použitím Voroného diagramu. I zde je nutné spolu s časovou náročností vyhledávání blízkých bodů zhodnotit časovou náročnost vytvoření diagramu.

LSH se používá pro vyhledávání blízkých sousedů ve vícedimenzionálním prostoru. Aby bylo vyhledávání pomocí LSH spolehlivé, je potřeba použít velký počet hashovacích tabulek, to znamená velké množství skupin hashovacích fukcí. Podle [18] ukazují experimentální výsledky, že je potřeba použít přes sto hashovacích tabulek.

Z pohledu výpočetní složitosti se jako vhodná jeví i metoda sweep-line.

5.3 Testování vybraných metod, vhodných pro řešení kontroly

Nad testovacími daty (data ORP Plzeň) bylo provedeno testování vybraných metod, kterými lze kontrolu řešit, a to metody hrubé síly, prostorového indexu R-tree a metody sweep-line. Metoda hrubé síly byla k testování vybrána pro svoji jednoduchost a snadnou algoritmizaci. Použití R-tree indexu se přímo nabízí, protože tento index je součástí DBMS Oracle a lze předpokládat, že použití standardních funkcí Oracle bude efektivní. Metoda sweep-line pak byla vybrána pro svou oblíbenost.

5.3.1 Metoda hrubé síly

Metoda hrubé síly byla na testovací data aplikována v rámci katastrálních území. To znamená, že pro jednotlivá katastrální území byl porovnán každý bod s každým. Testovací data obsahují 42 katastrálních území. Jednotlivá katastrální území pak obsahují desítky až tisíce stavebních objektů. Nejvíce stavebních objektů obsahuje katastrální území Plzeň a to téměř 13 000, to znamená 169 000 000 porovnání. Pro celá testovací data se pak jedná cca o 250 000 000 porovnání.

Procedura vytvořená v PL/SQL má vstupní parametr minimální vzdálenost. Pokud vzdálenost dvou bodů ve vstupních datech bude menší než tato minimální vzdálenost, tyto body budou uloženy do tabulky blízkých definičních bodů. Jádro procedury je uvedeno v příkladu 1. Procedura funguje tak, že postupně načte do kurzoru *c_KatUz* kódy katastrálních území. Pro každé katastrální území jsou pak vybrány všechny kódy a geometrie definičních bodů stavebních objektů, které se v něm nacházejí, a jsou uloženy do dvou totožných polí *SO1* a *SO2* (v Oracle se jedná o datový typ *TABLE*). První pole je pak postupně procházeno a každý záznam je porovnán se všemi záznamy z druhého pole. Vzdálenost definičních bodů je počítána pomocí funkce *SDO_DISTANCE*. Tato funkce je součástí knihovny geometrických funkcí *SDO_GEOM*. Funkce *SDO_DISTANCE* vrací datový typ *NUMBER* a její parametry jsou:

- *GEOM1 (SDO_GEOMETRY)* geometrie prvního objektu,
- *GEOM2 (SDO_GEOMETRY)* geometrie druhého objektu,
- *TOL (NUMBER)* hodnota tolerance vzdálenosti,
- *UNIT (VARCHAR2)* jednotka délky.

```
OPEN c_KatUz; --otevreni kurzoru pro prochazeni katastralnich uzemi
LOOP
  FETCH c_KatUz INTO tmp_KatUz;
  EXIT WHEN c_KatUz%NOTFOUND;
  AktKatUz := tmp_KatUz;

FOR i IN (SELECT s.kod, s.geom --naplneni dvou totoznych poli
      FROM StavebniObjektDFB s, ParcelaDFB
      WHERE s.IdentifikacniParcela=ParcelaDFB.ID
      AND ParcelaDFB.KatastralniUzemi=AktKatUz
      ORDER BY s.geom.SDO_POINT.X) LOOP
  n := n + 1;
  SO1(n) := i;
  SO2(n) := i;
END LOOP;

FOR i IN SO1.FIRST..SO1.LAST LOOP
  FOR j IN SO2.FIRST..SO2.LAST LOOP
          vzdal := SDO_GEOM.SDO_DISTANCE (SO1(i).geom, SO2(j).geom, tol, NULL);
          IF (vzdal<=minvzdal) THEN
            INSERT INTO K3_bf_3m (StavebniObjekt1, StavebniObjekt2, Vzdalenost)
            --vlozeni nalezenych prilis blizkych bodu do tabulky chyb
              VALUES (SO1(i).kod, SO2(j).kod, vzdal);
          END IF;
  END LOOP;
END LOOP;
COMMIT;
END LOOP;
CLOSE c_KatUz;
```

Příklad 1: Vyhledání blízkých definičních bodů pomocí metody hrubé síly.

5.3.2 Prostorový index R-tree

Oracle Spatial umožňuje použití dvou prostorových indexů, a to R-tree a Quadtree nebo kombinaci obou indexů nad jedním sloupcem s geometrií. Který z indexů bude použit, se pak specifikuje při konkrétním prostorovém dotazu. Pomocí R-tree je možné indexovat až čtyřdimenzionální data, Quadtree umožňuje indexaci pouze dvoudimenzionálních dat. Při vyhledávání nejbližšího souseda v Oracle pomocí

funkce pro vyhledání blízkých bodů *SDO_NN* je dotaz při použití R-tree vyřešen rychleji než při použití Quadtree [32].

Data a metadata indexů jsou v Oracle uložena v pohledech. Existují dvě skupiny pohledů pro uložení informací o prostorových indexech: *xxx_SDO_INDEX_INFO* a *xxx_SDO_INDEX_METADATA*, kde *xxx* může být *USER, DBA* nebo *ALL*. Pro uživatele jsou tyto pohledy pouze ke čtení. Pohledy *SDO_INDEX_INFO* obsahují základní informace o prostorových indexech a *SDO_INDEX_METADATA* obsahují detailní informace o datech. Pohledy jsou určeny pro uživatele s různým oprávněním podle následujícího rozdělení:

- *USER_SDO_INDEX_INFO, USER_SDO_INDEX_METADATA* obsahuje informace o indexech pro všechny prostorové tabulky vlastněné uživatelem,
- *ALL_SDO_INDEX_INFO, ALL_SDO_INDEX_METADATA* obsahuje informace o indexech pro všechny prostorové tabulky, na kterých může uživatel provádět dotaz *SELECT,*
- *DBA_SDO_INDEX_INFO, DBA_SDO_INDEX_METADATA* obsahuje informace o indexech pro všechny prostorové tabulky, na kterých může administrátor provádět dotaz *SELECT.*

Všechny pohledy typu *SDO_INDEX_INFO* obsahují stejné sloupce, jejich názvy, datový typ a obsah jsou znázorněny v tabulce 4. Stejně tak pohledy typu *SDO_INDEX_METADATA* obsahují stejné sloupce s mnoha dalšími informacemi jako například jméno indexu, dimenze sloupce s geometrií, nad kterým je index vytvořen, výška stromu pro R-tree index, počet uzlů v R-tree a mnoho dalších. Celý přehled je uveden v [32].

Tabulka 4: Sloupce pohledu *SDO_INDEX_INFO*.

NÁZEV SLOUPCE	DATOVÝ TYP	OBSAH SLOUPCE
INDEX_NAME	VARCHAR2	název indexu
TABLE_NAME	VARCHAR2	název tabulky, jež obsahuje indexovaný sloupec
COLUMN_NAME	VARCHAR2	název indexovaného sloupce
SDO_INDEX_TYPE	VARCHAR2	informace, zda se jedná o R-tree nebo Quadtree
SDO_INDEX_NAME	VARCHAR2	jméno tabulky prostorového indexu

Při aktualizaci sloupce, nad kterým je vytvořen R-tree index, je potřeba průběžně aktualizovat i index. K tomu slouží příkaz *ALTER INDEX REBUILD*. O tom, zda je nutné index aktualizovat, se rozhodne na základě výsledků následujících funkcí:

- *SDO_TUNE.ANALYZE_RTREE* poskytuje informace o kvalitě indexu a porovnává je s kvalitou indexu při jeho vytvoření nebo při poslední aktualizaci,
- *SDO_TUNE.RTREE_QUALITY* vrací aktuální kvalitu indexu,
- *SDO_TUNE.QUALITY_DEGRADATION* vrací degradaci aktuálního indexu [32].

Pro řešení CPP problému obsahuje Oracle Spatial defaultní funkci *SDO_NN.* Tato funkce používá k určení nejbližších bodů prostorový index. Funkce se volá ve tvaru *SDO_NN(GEOM1, GEOM2, PARAM [, DIST]);*

Jednotlivé parametry funkce znamenají:

- *GEOM1* sloupec geometrie, nad kterým musí být vytvořen prostorový index,
- *GEOM2* geometrie, ke které jsou z *GEOM1* vyhledáváni nejbližší sousedé,
- *PARAM* parametr určuje způsob chování funkce; datový typ je *VARCHAR2*, možnosti tohoto parametru jsou:
    - *SDO_BATCH_SIZE* specifikuje počet řádků, které mají být funkcí vráceny najednou a jsou dále omezovány podmínkou *WHERE*, popřípadě *ROWNUM*; vhodné k tomu, aby funkce vrátila potřebný

počet záznamů; toto klíčové slovo je možné použít pouze za použití R-tree indexu,

- *SDO_NUM_RES* specifikuje počet výsledků, které mají být funkcí vráceny; pokud je použito *SDO_BATCH_SIZE*, je toto klíčové slovo ignorováno,
- *UNIT* specifikuje použité jednotky,

- *DIST* se používá při volání funkce *SDO_NN_DISTANCE*, sloužící k určení vzdálenosti nejbližších bodů; datový typ je *NUMBER* [32].

Vytvoření R-tree indexu a použití funkce SDO_NN

Před vytvořením indexu je potřeba aktualizovat metadata. Metadata geometrie jsou uložena v tabulce *USER_SDO_GEOM_METADATA*, ke které uživatel nemá přímý přístup. Přístup k tabulce je pro uživatele zajištěn pomocí pohledu *USER_SDO_GEOM_METADATA*. Aktualizace metadat tedy spočívá ve vložení záznamu do tohoto pohledu. Vložení se provede příkazem, který je uveden v příkladu 2. Po aktualizaci metadat je možné vytvořit samotný index příkazem uvedeným v příkladu 3.

```
INSERT INTO user_sdo_geom_metadata
  (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES(
  'StavebniObjektDFB_INDEX', 'geom', SDO_DIM_ARRAY(
      SDO_DIM_ELEMENT('X', 0, 0.005, 0.005),
      SDO_DIM_ELEMENT('Y', 0, 0.005, 0.005)),
  NULL);
```

Příklad 2: Aktualizace metadat; vložení záznamu do tabulky *SDO_GEOM_ USER_METADATA*.

```
CREATE INDEX SO_spatial_idx
   ON StavebniObjektDFB_INDEX(geom)
   INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

Příklad 3: Příkaz pro vytvoření R-tree indexu

Spuštěním tohoto příkazu je automaticky vytvořena tabulka indexu, kterou Oracle pojmenuje MDRT% (kde % je systémem generovaný kód), která obsahuje jednoznačný identifikátor jednotlivých uzlů R-tree, jejich úroveň a další informace

v datovém typu *BLOB*. Dále je vytvořen záznam v pohledu *SDO_INDEX_METADATA*. Tabulka má téměř 40 sloupců. Záznam několika sloupců vytvořeného R-tree je uveden na obrázku 11. Výška vytvořeného stromu je 4, strom má 1616 uzlů, maximální počet potomků každého uzlu je 34.

SDO_INDEX_OWNER	SDO_INDEX_TYPE	SDO_RTREE_HEIGHT	SDO_RTREE_NUM_NODES	SDO_RTREE_FANOUT	SDO_MAXLEVEL
JINDRAM	RTREE	4	1616	34	64

Obr. 11: Záznam vytvořeného indexu v pohledu *SDO_INDEX_METADATA* (některé atributy).

Po vytvoření indexu je možné použít funkci Oracle *SDO_NN* pro vyhledání nejbližších bodů. Příkaz, kterým jsou vyhledány dvojice bodů, jež jsou blíže než 3 metry, je uveden v příkladu 4.

```
SELECT  distinct
   ISO.kod AS SO1, SO.kod AS SO2, mdsys.SDO_NN_DISTANCE(1) AS VZDAL
   FROM StavebniObjektDFB SO, StavebniObjektDFB_INDEX ISO
   WHERE SDO_NN(ISO.geom,  SO.geom,
      'sdo_num_res=4', 1) = 'TRUE' AND ISO.kod<>SO.kod
      AND mdsys.SDO_NN_DISTANCE(1)<=3;
```

Příklad 4: Použití funkce *SDO_NN*. Příkaz pro vyhledání dvojic bodů, které jsou blíže než 3 metry.

5.3.3 Metoda založená na sweep-line

Stejně jako metoda hrubé síly i sweep-line metoda byla na data aplikována v rámci katastrálních území. Vytvořená procedura má opět vstupní parametr minimální vzdálenost a dvojice bodů, jejichž vzdálenost je pod hranicí minimální vzdálenosti, jsou ukládány do tabulky blízkých definičních bodů. Jádro procedury je uvedeno v příkladu 5.

Procedura funguje tak, že do kurzoru načte katastrální území a postupně pro každé katastrální území vybere kód, geometrii a souřadnice X, Y jednotlivých definičních bodů stavebních objektů v daném katastrálním území, seřadí je vzestupně podle souřadnice X a uloží je do pole. Pole je pak postupně procházeno a pro každý záznam v poli na pozici *n* je porovnávána souřadnice X se souřadnicí X bodu na pozici *n-1*, pokud je rozdíl menší nebo roven minimální vzdálenosti, jsou stejným způsobem porovnány souřadnice Y. Pokud i jejich rozdíl je menší nebo roven minimální vzdálenosti, je spočtena vzdálenost těchto dvou definičních bodů

z geometrie. Pokud i tato spočtená vzdálenost splňuje kritérium minimální vzdálenosti, je dvojice uložena do tabulky blízkých definičních bodů a stejný postup se opakuje pro bod na pozici *n-2*. Postup se opakuje pro všechny body *n-i*, dokud rozdíl souřadnice X bodu na pozici *n* a bodu na pozici *n-i* nepřekročí minimální vzdálenost.

```
--naplneni pole
FOR i IN (SELECT s.kod, s.geom, s.geom.SDO_POINT.X, s.geom.SDO_POINT.Y
      FROM StavebniObjektDFB s, ParcelaDFB
      WHERE s.IdentifikacniParcela=ParcelaDFB.ID
      AND ParcelaDFB.KatastralniUzemi=AktKatUz
      ORDER BY s.geom.SDO_POINT.X) LOOP
  n := n + 1;
  SO(n) := i;
END LOOP;

FOR i IN SO.FIRST..SO.LAST LOOP
  j:=0;
  WHILE (j<i-2) LOOP
      j:=j+1;
      IF (SO(i).X-SO(i-j).X)<=minvzdal THEN
        IF (SO(i).Y-SO(i-j).Y)<=minvzdal THEN
          vzdal := SDO_GEOM.SDO_DISTANCE (SO(i).geom, SO(i-j).geom, tol, NULL);
          IF (vzdal<=minvzdal) THEN
            --vlozeni nalezenych prilis blizkych bodu do tabulky chyb
            INSERT INTO K3_DFB (StavebniObjekt1, StavebniObjekt2, Vzdalenost)
              VALUES (SO(i).kod, SO(i-j).kod, vzdal);
          END IF;
        END IF;
      ELSE
        EXIT;
      END IF;
  END LOOP;
END LOOP;
COMMIT;
END LOOP;
CLOSE c_KatUz;
```

Příklad 5: Vyhledání blízkých definičních bodů pomocí metody sweep-line.

5.3.4 Porovnání testovaných metod

Vytvořené metody byly testovány pro vstupní parametr minimální vzdálenosti 3 metry a 5 metrů. Z časového je potřeba porovnat u metody hrubé síly a metody sweep-line dobu běhu procedur. U funkce *SDO_NN* pak dobu běhu příkazu *SELECT*, který je uveden na konci kapitoly 4.3.2 v příkladu 4, a dobu vytvoření R-tree indexu. Doba běhu jednotlivých procedur je uvedena v tabulce 5.

Tabulka 5: Časová náročnost testovaných metod.

METODA	AKCE	3 metry		5 metrů	
		ČAS [s]	[min]	ČAS [s]	[min]
HRUBÁ SÍLA	běh procedury	59860,37	997,67	59875,42	997,92
R-TREE + SDO_NN	vytvoření R-tree	5,55	0,09	5,55	0,09
	běh *SELECT* příkazu	121,46	2,02	952,51	15,87
SWEEP-	běh procedury	1098,38	18,31	1585,01	26,42

Časově tedy vychází nejlépe vytvoření R-tree indexu nad daty a následné použití funkce *SDO_NN*. Časová náročnost tohoto postupu se zvyšující se hodnotou minimální vzdálenosti roste především v závislosti na velikosti parametru *SDO_NUM_RES*, který specifikuje počet navrácených výsledků. Pro minimální vzdálenost 2 metry byla hodnota tohoto parametru nastavena na 4, pro minimální vzdálenost 5 metrů byla nastavena na 8. Tento parametr je tedy potřeba volit rozumně tak, aby se příliš vysokou hodnotou parametru zbytečně neprodlužovala doba běhu. Na druhou stranu nesmí být parametr příliš nízký, protože pak by mohlo dojít k tomu, že nebudou vyhledány všechny dvojice blízkých bodů.

Doba běhu metody sweep-line se zvyšuje v závislosti na růstu vstupní hodnoty rychle. Doba běhu procedury založené na hrubé síle je stejná pro jakoukoli vstupní hodnotu.

Po vymazání duplicitních záznamů z tabulek podezřelých definičních bodů, nalezených jednotlivými procedurami, můžeme porovnat i výsledky jednotlivých metod, tedy konkrétní dvojice bodů, které byly metodami označeny za příliš blízké.

Předpoklad drobných rozdílů ve výsledcích se potvrdil. Stejné výsledky dávají metoda sweep-line a metoda hrubé síly. Konkrétní počet těchto nalezených dvojic závisí vždy na parametru minimální vzdálenost. Například pro minimální vzdálenost 2 metry je metodou R-tree indexu nalezeno 355 dvojic blízkých bodů. O 2 dvojice blízkých bodů méně pak naleznou metoda hrubé síly a metoda sweep-line. To je způsobeno tím, že první dvě metody kontrolují data v rámci katastrálních území. Neodhalí tedy takové případy, kde z dvojice blízkých bodů leží každý v jiném katastrálním území. Metody je samozřejmě možné použít i na celá data bez dělení na katastrální území, to by ale znamenalo výrazně vyšší časovou náročnost. Nalezené dvojice blízkých bodů ale nelze definitivně označit za chyby, tyto dvojice jsou pouze podezřelé z chyby. S těmito nalezenými dvojicemi je potřeba dále pracovat a analyzovat jejich další atributy, což je provedeno v kapitole 5.4.

Jak již bylo řečeno výše, pro metodu hrubé síly by v takovém případě časová náročnost byla neúnosná. Z hlediska počtu nalezených chyb se jeví tedy také jako nejvhodnější použití prostorového indexu R-tree a funkce *SDO_NN*.

Problém tohoto postupu je pouze v tom, že při časté aktualizaci dat je potřeba hlídat kvalitu indexu a index v případě potřeby přestavět nebo odstranit a vytvořit nový index.

5.4 Doporučené řešení kontroly

Pro řešení kontroly v produkčním prostředí RÚIAN byl na základě provedeného testování (viz kapitola 5.3.4) vybrán postup vytvoření R-tree indexu a následného použití standardní funkce *SDO_NN*. Důvodem jsou především nízké časové nároky tohoto postupu v porovnání s ostatními testovanými postupy a kvalita (úplnost) výsledků.

Pro uložení nalezených dvojic blízkých definičních bodů stavebních objektů a jejich důležitých atributů byla vytvořena tabulka *K3_BlizkeDFB*. Struktura tabulky, její atributy a význam těchto atributů jsou znázorněny na obrázku 12.

Před spuštěním kontroly je nejdříve je potřeba vytvořit kopii tabulky, nad kterou bude kontrola prováděna. Jedná se o tabulku, která obsahuje geometrii definičních bodů stavebních objektů. Nad sloupcem s geometrií této nově vytvořené tabulky je potřeba vytvořit prostorový R-tree index. Vytvoření prostorového R-tree indexu je popsáno v kapitole 5.3.2.

COLUMN_NAME	DATA_TYPE	COMMENTS
SO1	NUMBER	-- kod prvniho stavebniho objektu
SO2	NUMBER	-- kod druheho stavebniho objektu
VZDALENOST	NUMBER	-- vzdalenost SO1 a SO2
SO1_CD	VARCHAR2(254 BYTE)	--cislo domovni SO1
SO2_CD	VARCHAR2(254 BYTE)	--cislo domovni SO2
SO1_CO	NUMBER(9,0)	--cast obce SO1
SO2_CO	NUMBER(9,0)	--cast obce SO2
SO1_ZV	NUMBER(4,0)	--zpusob vyuziti SO1
SO2_ZV	NUMBER(4,0)	--zpusob vyuziti SO2
SO1_KU	NUMBER(9,0)	--katastralni uzemi, na ktere ma vazbu parcela SO1
SO2_KU	NUMBER(9,0)	--katastralni uzemi, na ktere ma vazbu parcela SO2
SO1_PAM	NUMBER	--pocet adresnich mist, ktera maji vazbu na SO1
SO2_PAM	NUMBER	--pocet adresnich mist, ktera maji vazbu na SO2

Obr. 12: Struktura tabulky *K3_BlizkeDFB* pro uložení blízkých definičních bodů.

Pro řešení kontroly je vytvořena funkce *pK3_BlizkeDFB*. Procedura má vstupní parametr *minvzdal*, tedy vzdálenost dvou definičních bodů, při jejímž překročení již nejsou definiční body považovány za příliš blízké. Procedura nejprve smaže záznamy z tabulky *K3_BlizkeDFB*. Dále pomocí příkazu *SELECT*, v němž je volána funkce *SDO_NN*, nalezne dvojice blízkých bodů a jejich vzdálenost a uloží je do tabulky *K3_BlizkeDFB*. Takto jsou ale všechny dvojice blízkých bodů nalezeny a uloženy duplicitně pouze s tím rozdílem, že ve druhém záznamu jsou objekty SO1 a SO2 prohozeny. Následuje tedy vymazání duplicitních záznamů a spuštění funkce *pK3_DoplneniAtributu*. Tato funkce vyhledá a doplní příslušné atributy pro jednotlivé záznamy v tabulce *K3_BlizkeDFB*.

Procedura *K3_DoplneniAtributu* pracuje tak, že načte do kurzoru dvojice kódů stavebních objektů *SO1* a *SO2* z tabulky *pK3_BlizkeDFB*. Pro oba stavební objekty pomocí příkazů *SELECT* vybere z tabulek stavebních objektů, parcel a adresních míst požadované hodnoty, kterými jsou číslo domovní, část obce,

katastrální území, způsob využití stavebního objektu a počet adresních míst. Po výběru těchto hodnot provede *UPDATE* tabulky *K3_BlizkeDFB*, v rámci kterého nastaví hodnotu příslušných atributů podle vyhledaných hodnot. Část procedury, která vyhledává atributy pro stavební objekt *SO1*, je uvedena v příkladu 6. Vyhledání atributů pro stavební objekt *SO2* funguje stejným způsobem.

```
LOOP
  FETCH c_Kody INTO tmp_Kody;
  EXIT WHEN c_Kody%NOTFOUND;
-- vyber cisla domovniho, casti obce a zpusobu vyuziti objektu SO1
  SELECT CislaDomovni, CastObce, ZpusobVyuz INTO tmp_Atrib1
    FROM StavebniObjektDFB
    WHERE StavebniObjektDFB.kod=tmp_Kody.SO1;
-- vyber katastralniho uzemi objektu SO1
  SELECT KatastralniUzemi INTO KU1
    FROM StavebniObjektDFB, ParcelaDFB
    WHERE StavebniObjektDFB.kod=tmp_Kody.SO1
      AND StavebniObjektDFB.IdentifikacniParcela=ParcelaDFB.Id;
-- vyber poctu abresnich mist, ktere maji vazbu na stavebni objekt SO1
  SELECT COUNT(kod) INTO PAM1
    FROM AdresniMisto
    WHERE stavebniob=tmp_Kody.SO1;
-- aktualizace tabulky K3_BlizkeDFB
-- vlozeni atributu k prislusnym stavebnim objektum
  UPDATE K3_BlizkeDFB SET
        SO1_CD=tmp_Atrib1.SO_CD, SO1_CO=tmp_Atrib1.SO_CO,
        SO1_ZV=tmp_Atrib1.SO_ZV, SO1_KU=KU1,
        SO1_PAM=PAM1
      WHERE SO1=tmp_Kody.SO1;
END LOOP;
```

Příklad 6: Část procedury *pK3_DoplneniAtributu*. Doplnění atributů stavebnímu objektu SO1.

Procedura *pK3_BlizkeDFB* je uvedena v příkladu 7.

```
CREATE OR REPLACE PROCEDURE pK3_BlizkeDFB(minvzdal INTEGER) AS
BEGIN
--pouziti funkce SDO_NN
-- vymazani zaznamu z pomocne tabulky pro ukladani dvojic blizkych stavebnich objektu
DELETE K3_BlizkeDFB;
INSERT INTO K3_BlizkeDFB(SO1, SO2, VZDALENOST)
--vyhledani dvojic blizkych definicnich bodu a jejich vzdalenosti
SELECT  distinct
   ISO.kod AS SO1, SO.kod AS SO2, MDSYS.SDO_NN_DISTANCE(1) AS VZDAL
   FROM StavebniObjektDFB SO, StavebniObjektDFB_INDEX ISO
   WHERE SDO_NN(ISO.geom,  SO.geom,
      'sdo_num_res=5',1)='TRUE'
      AND ISO.kod<>SO.kod
      AND MDSYS.SDO_NN_DISTANCE(1)<=minvzdal;
-- vamazani stejnych zaznamu s prohozenym SO1 a SO2
DELETE FROM K3_BlizkeDFB
    WHERE SO1 IN (Select SO2 FROM K3_BlizkeDFB)
    AND SO2 IN (Select SO1 FROM K3_BlizkeDFB)
    AND SO1>SO2;
-- spusteni procedury pro doplneni atributu k nalezenym blizkym definicnim bodum
  pK3_DoplneniAtributu;
END;
```

Příklad 7: Procedura *pK3_BlizkeDFB.* Procedura pro vyhledání definičních bodů stavebních objektů, které jsou blíž než definovaná minimální vzdálenost *minvzdal*.

Parametr *sdo_num_res* je v proceduře *pK3_DoplneniAtributu* nastaven na hodnotu 5. Podle potřeby je možné ho snížit, případně zvýšit v závislosti na velikosti vstupního parametru *minvzdal*.

Při kompilaci procedury ale nastává problém, procedura nelze pomocí Oracle SQL Developeru zkompilovat a kompilace končí chybou:

ERROR: PLS-00707: Nepodporovaná konstrukce nebo vnitřní chyba [2603].

Pro řešení tohoto problému byl vytvořen dotaz v Oracle fóru. Dotaz byl zodpovězen tak, že se pravděpodobně jedná o problém Oracle SQL Developeru a je doporučováno zkompilovat proceduru pomocí SQL*PLUS, což je konzolový SQL klient pro Oracle [32]. Další možností, jak vyřešit problém s kompilací, je změna nastavení Oracle SQL Developeru. V nabídce *Tools/Preferences/Database* je potřeba vybrat záložku *PL/SQL Compiler* a změnit nastavení položky *PLScope identifiers* z hodnoty *All* na hodnotu *None*. Po této změně nastavení se Oracle SQL

Developer při kompilaci chová stejně jako SQL*PLUS a proceduru je možné bez dalších problémů zkompilovat.

Další možností je místo procedury použít anonymní PL/SQL blok, zobrazený v příkladu 8. Nevýhodou bloku je to, že nelze použít vstupní parametr, ale minimální vzdálenost je možné volitelně nastavit uvnitř bloku omezením funkce *MDSYS.SDO_NN_DISTANCE* v podmínce *WHERE* požadovanou vzdáleností, například pro minimální vzdálenost 2 metry se nastaví *MDSYS.SDO_NN_DISTANCE(1)<=2*. Dále je potřeba pro různé minimální vzdálenosti upravit hodnotu parametru SDO_NUM_RES. Doporučuji hodnotu volit ve velikosti cca 1,5 násobku minimální vzdálenosti, tedy pro hodnotu 2 metry nastavit *SDO_NUM_RES=3*, popřípadě *4*. Při volbě příliš vysoké hodnoty zbytečně prodloužíme dobu vyhledávání. Při volbě příliš nízké hodnoty zase hrozí, že nebudou odhaleny všechny dvojice blízkých bodů. Hodnotu funkce *MDSYS.SDO_NN_DISTANCE* je potřeba omezit v PL/SQL bloku zobrazeném v příkladu 8 na řádku 15, parametr *SDO_NUM_RES* se nastaví na řádku 13.

```
1  BEGIN
2  --anonymni blok pro nalezeni blizkych bodu pomoci R-tree indexu
3  --pouziti funkce SDO_NN
4
5  -- vymazani zaznamu z pomocne tabulky pro ukladani dvojic blizkych bodu
6  DELETE K3_BlizkeDFB;
7  INSERT INTO K3_BlizkeDFB(SO1, SO2, VZDALENOST)
8  --vyhledani dvojic blizkych definicnich bodu a jejich vzdalenosti
9  SELECT  distinct
10    ISO.kod AS SO1, SO.kod AS SO2, MDSYS.SDO_NN_DISTANCE(1) AS VZDAL
11    FROM StavebniObjektDFB SO, StavebniObjektDFB_INDEX ISO
12    WHERE SDO_NN(ISO.geom,  SO.geom,
13       'sdo_num_res=4', 1) = 'TRUE'
14       AND ISO.kod<>SO.kod
15       AND MDSYS.SDO_NN_DISTANCE(1)<=2;
16 -- vamazani stejnych zaznamu s prohozenym SO1 a SO2
17 DELETE FROM K3_BlizkeDFB
18     WHERE SO1 IN (Select SO2 FROM K3_BlizkeDFB)
19     AND SO2 IN (Select SO1 FROM K3_BlizkeDFB)
20     AND SO1>SO2;
21 -- spusteni procedury pro doplneni atributu k nalezenym blizkym bodum
22   pK3_DoplneniAtributu;
23 END;
24 /
```

Příklad 8: PL/SQL blok pro vyhledání blízkých definičních bodů. Vyhledání definičních bodů, které jsou blíž než definovaná minimální vzdálenost, pomocí funkce *SDO_NN*.

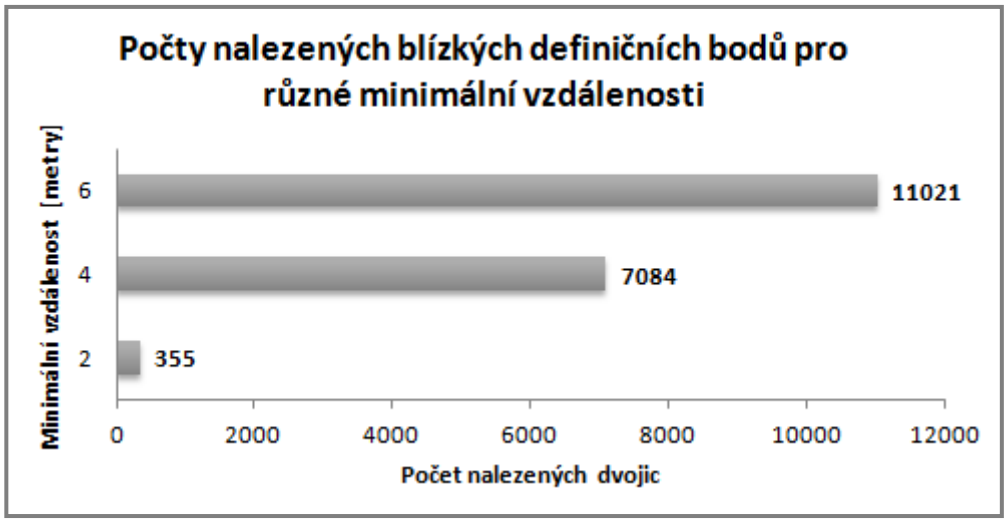
5.5 Vyhodnocení kontroly

Kontrola byla provedena nad testovacími daty ORP Plzeň pro minimální vzdálenosti 2, 4 a 6 metrů. Výsledky jsou zobrazeny v grafu 1.

Zajímavé jsou případy dvojic definičních bodů, jejichž vzdálenost je nulová. Těchto dvojic bylo v testovacích datech nalezeno 144. Ve 139 případech je jeden ze stavebních objektů bez čísla domovního a tedy bez vazby na část obce.

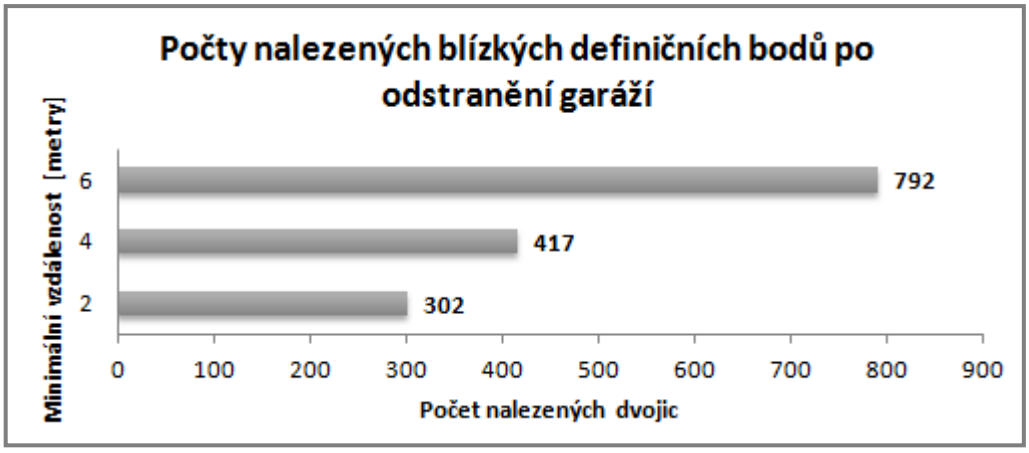
Je zřejmé, že v některých případech se nebude jednat o duplicity. Jedná se například o problém garáží, jež byl popsán výše. Pokud oba stavební objekty z dvojice blízkých bodů mají způsob využití garáž, je vysoce pravděpodobné, že se o duplicitu nejedná. Tento předpoklad potvrdilo i vyhledání mnoha těchto dvojic v katastrální mapě a vizuální porovnání s vrstvou definičních bodů stavebních objektů RÚIAN. Tyto záznamy lze z tabulky blízkých bodů odstranit jednoduchým příkazem: *DELETE FROM K3_BlizkeDFB WHERE SO1_ZV=18 AND SO2_ZV=18;* (kód způsobu využití garáž je 18 [33]).
Počty dvojic po odstranění garáží jsou znázorněny v grafu 2. Z porovnání grafů 1 a 2 je zřejmé, že tyto záznamy tvořily velkou část nalezených blízkých dvojic.

Graf 1: Porovnání počtu blízkých bodů pro různé minimální vzdálenosti.



Zbylé vyhledané dvojice definičních bodů stavebních objektů jsou podezřelé z duplicity a je potřeba pečlivé porovnání dalších atributů těchto stavebních objektů, kontrola katastrální mapy, popřípadě kontrola skutečného stavu v terénu, aby bylo možné s jistotou tvrdit, zda se jedná o duplicity. Slabým místem jsou například podzemní stavební objekty, které jsou v RÚIAN evidovány a jejichž geometrie je vedena formou definičních bodů. U nich se může stát, že jejich definiční bod je vyhledán jako blízký jinému definičnímu bodu, ačkoli se o duplicitu nejedná. V některých případech může být blízkost bodů způsobena pouze jejich špatným umístěním, například umístěním na kraji stavebního objektu místo v těžišti. V těchto případech by tedy bylo vhodné opravit umístění definičních bodů.

Graf 2: Počty blízkých bodů po odstranění garáží.



6 Vyhledání stavebních objektů, které mají podezřelá čísla (popisná, evidenční)

Úkolem této kontroly je vyhledat stavební objekty, jejichž čísla domovní (popisná nebo evidenční) se vymykají číslování v řadě.

Podle Vyhlášky o základním registru územní identifikace, adres a nemovitostí [6] je identifikačním údajem pro zápis stavebního objektu číslo popisné nebo evidenční spolu s údajem o části obce, nebo údaj o tom, že se toto číslo nepřiděluje. K očíslování budovy (tedy stavebního objektu ve smyslu RÚIAN) dochází zápisem do RÚIAN [34].

Číslo domovní je identifikátor stavebního objektu v rámci části obce. To znamená, že čísla popisná i čísla evidenční jsou v části obce jedinečná a neopakují se. V RÚIAN jsou evidovány i stavební objekty bez čísla domovního.

Přidělení čísla domovního je nutnou podmínkou pro zápis budovy do katastru nemovitostí. Většina budov je označena číslem popisným. Čísla evidenční se přidělují stavbám pro rodinnou rekreaci, stavbám dočasným a budovám, které nevyžadují stavební povolení. O tom, jaké číslo domovní se budově přidělí, rozhoduje obec. Čísla domovní, která již byla přidělena, nelze použít opakovaně. Opakovaně lze použít pouze číslo orientační v případě, že nová budova stojí na místě zaniklé budovy [34].

Čísla popisná i evidenční by měla v rámci části obce tvořit souvislou číselnou řadu. Ta může být přerušena tehdy, došlo-li ke zrušení stavebního objektu, který měl přidělené číslo domovní. Velká přerušení číselné řady jsou tedy podezřelá.

K řešení této kontroly je možné přistupovat dvěma způsoby z hlediska přerušení číselné řady:

1. Podezřelá jsou konkrétní čísla domovní, v jejichž okolí dochází z obou stran (případně pouze z jedné strany, pokud se jedná o číslo na začátku nebo na konci číselné řady) k nepřiměřeně velkému přerušení souvislé číselné řady. To

znamená, že při zkoumání číselné řady {1, 2, 4, 5, 36, 37, 38, 39, 40, 78, 112, 113, 115, 156} a použití mezní hodnoty 30, budou podezřelá čísla {78, 156}. Tento přístup je odůvodněn přímo zadáním kontroly, která má vyhledat konkrétní stavební objekty s podezřelými čísly domovními. Další důvod je možné uvést na příkladu výše uvedené číselné řady. Jedná se o to, že až do čísla 5 je číselná řada relativně spojitá, od čísla 36 do čísla 40 je také spojitá. Podezřelé je pak až číslo 78, které je od sousedních čísel z obou stran odděleno velkou mezerou. Následuje opět relativně spojitý úsek mezi čísly 112 a 115. Číslo 156 je opět podezřelé, protože je před ním opět velká mezera a nenavazuje na předchozí číselnou řadu.

2. Podezřelá jsou přerušení číselné řady, která jsou větší než daná hodnota. To znamená, že při zkoumání číselné řady {1, 2, 4, 5, 36, 37, 38, 39, 40, 78, 112, 113, 115, 156} a použití mezní hodnoty 30, budou podezřelá přerušení mezi dvojicemi {5, 36}, {40, 78}, {78, 112} a {115, 156}. Tento přístup je odůvodněn tím, že jakákoli nepřiměřeně velká přerušení souvislé číselné řady jsou podezřelá.

Pro vyřešení obou možností přístupu jsou vytvořeny dvě procedury. Jedna z nich vyhledá konkrétní podezřelá čísla domovní a druhá vyhledá podezřelá přerušení. Obě procedury mají vstupní parametr, který v prvním případě určuje, jak velké přerušení musí být kolem čísla domovního z obou stran, aby bylo označeno za podezřelé. Ve druhém případě vstupní parametr určuje, jak velké musí být přerušení, aby bylo označeno za podezřelé. Po kontrole dat a po spuštění procedur s různými hodnotami vstupního parametru a porovnání výsledků se jeví jako vhodné testování se vstupním parametrem o hodnotách 15, 30 a 50.

6.1 Řešení kontroly

Číslo domovní je v tabulce stavebních objektů uloženo ve sloupci *cisladomovni*, který je datového typu *VARCHAR2.* Zda se jedná o stavební objekt s číslem popisným nebo evidenčním, je pak rozlišeno pomocí atributu *typSO*, který pro stavební objekt s číslem popisným nabývá hodnotu 1 a pro stavební objekt s číslem evidenčním nabývá hodnotu 2. Jeden stavební objekt může mít přiděleno více čísel domovních. Jedná se například o bytové domy s několika samostatnými vchody,

kde každý vchod má vlastní číslo domovní. V databázi jsou uložena všechna čísla stavebního objektu a jsou od sebe oddělena čárkou. Ukázka takového záznamu v databázi je na obrázku 13.

KOD	CISLADOMOVNI	TYPSO	IDENTIFIKACNIPARCELA	CASTOBCE	ZPUSOBVYUZ
23995408	346,347,348	1	616020405	71056	6
23995483	350,351,352,353,354	1	615938405	71056	6

Obr. 13: Uložení stavebního objektu, který má více čísel popisných.

Tento způsob uložení čísel domovních je ale nevhodný pro další zpracování. Před samotnou kontrolou je tedy nutné tyto záznamy rozdělit. To je vyřešeno vytvořením tabulky *K1_CislaDomovni*, ve které je pro každé číslo domovní samostatný záznam. Tato tabulka neobsahuje všechny atributy stavebních objektů, ale jen několik potřebných atributů. Struktura tabulky, její atributy a datové typy atributů jsou na obrázku 14. V této tabulce jsou uloženy pouze stavební objekty s číslem popisným nebo evidenčním a pro každý stavební objekt je v tabulce tolik záznamů, kolik má čísel domovních.

COLUMN_NAME	DATA_TYPE	COMMENTS
KOD	NUMBER(9,0)	-- kod stavebniho objektu
CD	NUMBER(10,0)	-- cislo domovni stavebniho objektu
TYPSO	NUMBER(4,0)	-- typ stavebniho objektu
CASTOBCE	NUMBER(9,0)	-- cast obce, ve ktere lezi stavebni objekt

Obr. 14: Struktura tabulky *K1_CislaDomovni*.

Pro naplnění této tabulky je vytvořena procedura *pK1_RozlozeniCD*. Ta nejprve smaže všechny záznamy z tabulky *K1_CislaDomovni*. Následně načte do kurzoru kódy částí obce a postupně je prochází. Pro každou část obce načte do kurzoru všechny stavební objekty, které mají na příslušnou část obce vazbu a postupně je zpracovává. Pokud se pro stavební objekt v atributu *cisladomovni* nenachází znak čárka (,), je tento stavební objekt ihned uložen do tabulky *K1_CislaDomovni* a jeho číslo domovní je pomocí Oracle funkce *TO_NUMBER* převedeno na datový typ *NUMBER*. Pokud se v něm znak čárka nachází, je potřeba tento záznam rozložit. Celý záznam, obsahující více čísel popisných s čárkami, je uložen do proměnné *retezec*. Každé číslo domovní je z proměnné *retezec* vyseparováno jako podřetězec od začátku řetězce až do posledního znaku před

výskytem znaku čárka pomocí Oracle funkce *SUBSTR*. Vyseparované číslo je převedeno na datový typ NUMBER a je spolu s kódem, typem stavebního objektu a částí obce uloženo do tabulky *K1_CislaDomovni*. Po uložení je proměnná *retezec* oříznuta o již nalezené číslo domovní a čárku za číslem a postup se opakuje, dokud nejsou vyseparována všechna čísla domovní. Jádro procedury *pK1_RozlozeniCD* je uvedeno v příkladu 9. Rozložená čísla domovní stavebních objektů z obrázku 13, uložená v tabulce *K1_CislaDomovni*, jsou na obrázku 15. Další postup se u přístupu 1 a 2 liší.

```
BEGIN
  carka:=0;
  OPEN cSt_Obj;
  LOOP
    FETCH cSt_Obj INTO tmp_st;
    EXIT WHEN cSt_Obj%NOTFOUND;
    --pokud se v CislaDomovni nenachazi znak ',' je radek kurzoru ihned ulozen do pomocne tabulky
    IF (INSTR(tmp_st.CislaDomovni,',')=0) THEN
      --vlozeni radku kurzoru do pomocne tabulky K1_CislaPopisna
      INSERT INTO K1_CislaDomovni (kod, CD, typSO, CastObce)
        VALUES (tmp_st.kod, TO_NUMBER(tmp_st.CislaDomovni,'99999999'), tmp_st.typSO, tmp_st.CastObce);
    ELSE  --v CislaDomovni nachazi znak ',' CislaDomovni je potreba rozlozit na jednotliva cisla popisna
      retezec:=tmp_st.CislaDomovni;
      LOOP
        carka:=INSTR(retezec,',');
        IF carka=0 THEN --nastava kdyz v retezci zbyde posledni cislo popisne
          --vlozeni do pomocne tabulky K1_CislaPopisna
          INSERT INTO K1_CislaDomovni (kod, CD, typSO, CastObce)
            VALUES (tmp_st.kod, TO_NUMBER(retezec,'99999999'), tmp_st.typSO, tmp_st.CastObce);
          EXIT;
        ELSE
          --AktCislo(aktualni cislo popisne) vytvoreno jako substring od zacatku retezce
          AktCislo:=SUBSTR(retezec,1,carka-1);--do posledni pozice pred vyskytem carky
          --vlozeni do pomocne tabulky K1_CislaPopisna
          INSERT INTO K1_CislaDomovni (kod, CD, typSO, CastObce)
            VALUES (tmp_st.kod, TO_NUMBER(AktCislo,'99999999'), tmp_st.typSO, tmp_st.CastObce);
          retezec:=SUBSTR(retezec,carka+1); --oriznuti retezce o jiz nalezene cislo popisne a carku
        END IF;
      END LOOP;
    END IF;
  END LOOP;
  CLOSE cSt_Obj;
END;
```

Příklad 9: Procedura *pK1_RozlozeniCD*; procedura pro rozložení řetězce čísle domovních na jednotlivé záznamy.

KOD	CD	TYPSO	CASTOBCE
23995408	346	1	71056
23995408	347	1	71056
23995408	348	1	71056
23995483	350	1	71056
23995483	351	1	71056
23995483	352	1	71056
23995483	353	1	71056
23995483	354	1	71056

Obr. 15: Uložení stavebních objektů z obrázku 13 v tabulce *K1_CislaDomovni.*

1. Vyhledání podezřelých čísel domovních

Pro řešení tohoto přístupu ke kontrole je vytvořena procedura *pK1_PodezrelaCPCE*. Pro ukládání podezřelých výsledků kontroly jsou vytvořeny dvě totožné tabulky, tabulka *K1_PodezrelaCP* pro ukládání podezřelých čísel popisných a *K1_PodezrelaCE* pro ukládání podezřelých čísel evidenčních. Struktura tabulek je na obrázku 16. Do tabulky se kromě podezřelého čísla domovního ukládají ještě číslo, které mu v seřazené číselné řadě předchází a které ho následuje. To proto, aby bylo z tabulky podezřelých čísel domovních zřejmé, jak velké nespojitosti jsou v okolí podezřelých čísel.

COLUMN_NAME	DATA_TYPE	COMMENTS
STAVEBNIOBJEKT	NUMBER(9,0)	-- kod stavebniho objektu
PODEZRELE	NUMBER(10,0)	-- podezrele cislo domovni
CASTOBCE	NUMBER(10,0)	-- cast obce, na kterou ma stavebni objekt vazbu
PREDCHOZI	NUMBER(10,0)	-- cislo domovni, ktere v ciselne rade predchazi
NASLEDUJICI	NUMBER(10,0)	-- cislo domovni, ktere v ciselne rade nasleduje

Obr. 16: Struktura tabulek *K1_PodezrelaCP* a *K1_PodezrelaCE.*

Procedura *pK1_PodezrelaCPCE* má vstupní parametr *MaxRozdil*, který definuje maximální rozdíl, který může být mezi aktuálně kontrolovaným číslem domovním a číslem domovním, které v číselné řadě předchází, a aktuálně kontrolovaným číslem domovním a číslem domovním, které v číselné řadě následuje. Pokud je tento rozdíl překročen, je aktuálně kontrolované číslo

považováno za podezřelé. Část procedury *pK1_PodezrelaCPCE*, kde se řeší vyhledávání podezřelých čísel popisných, je uvedena v příkladu 10.

Procedura nejprve vymaže všechny záznamy z tabulek *K1_PodezrelaCP* a *K1_PodezrelaCE*. Poté načte do kurzoru kódy všech částí obce a následně prochází jednotlivé části obce. Pro každou část obce jsou do kurzoru vybrány všechny stavební objekty s číslem popisným, jež mají na danou část obce vazbu a jsou seřazeny vzestupně podle čísla popisného. První číslo popisné je porovnáváno s následujícím číslem popisným, a pokud jejich rozdíl překročí hodnotu *MaxRozdil*, je toto číslo popisné, stavební objekt, kterému náleží, příslušná část obce a následující číslo popisné uloženo do tabulky *K1_PodezrelaCP*. Atribut *predchozi* zůstane v tomto případě nevyplněn, protože nejnižšímu číslu žádné číslo popisné nepředchází. Dále se pokračuje tak, že je vždy udržováno v paměti aktuálně zkoumané číslo, číslo, které mu předchází a následující číslo. Z těchto hodnot jsou vypočteny rozdíly aktuálního a předchozího čísla (*prozdil*) a aktuálního a následujícího čísla (*nrozdil*). Pokud hodnota *nrozdil* překročí *MaxRozdil*, kontroluje se dále, zda i *prozdil* překročí *MaxRozdil*. Pokud ano, je aktuálně zkoumané číslo uloženo do tabulky *K1_PodezrelaCP*. Pokud ne, pak dojde k posunu na další číslo, hodnotě *prozdil* se přiřadí hodnota *nrozdil* a hodnota *nrozdil* je nově spočtena. Poslední číslo popisné je porovnáno pouze s předchozím číslem popisným a atribut *nasledujici* zůstane v případě ukládání do *K1_PodezrelaCP* nevyplněn. Stejným způsobem jsou zkontrolována i čísla evidenční s tím rozdílem, že výsledky jsou ukládány do tabulky *K1_PodezrelaCE*. Část procedury, ve které dochází k porovnávání čísel popisných, je uvedena v příkladu 10. Pokud by se v datech nevyskytovala žádná přerušení větší než hodnota *MaxRozdil*, provedla by procedura n porovnání, kde n je počet testovaných domovních čísel. Nejhorší případ by nastal, pokud by přerušení větší než *MaxRozdil* bylo za každým číslem. Pak by bylo provedeno $2n$ porovnání. Nejhorší výpočetní složitost procedury je tedy $O(2n)$, řádově se jedná o složitost $O(n)$.

```
--zkoumani nejnizsiho cisla domovniho
IF (NaslCislo-AktCislo)>MaxRozdil THEN
  INSERT INTO K1_PodezrelaCP (StavebniObjekt, CastObce, Podezrele, Predchozi, Nasledujici)
    VALUES (AktKod, tmp_CO, AktCislo, NULL, NaslCislo);
END IF;
LOOP
  FETCH c_CD INTO tmp_CD;
  EXIT WHEN c_CD%NOTFOUND;
  PredchCislo:=AktCislo;
  AktCislo:=NaslCislo;
  prozdil:=nrozdil;
  AktKod:=NaslKod;
  NaslCislo:=tmp_CD.CD;
  NaslKod:=tmp_CD.kod;
  nrozdil:=NaslCislo-AktCislo;
  IF (nrozdil)>MaxRozdil THEN
    IF (prozdil)>MaxRozdil THEN
      INSERT INTO K1_PodezrelaCP (StavebniObjekt, CastObce, Podezrele, Predchozi, Nasledujici)
        VALUES (AktKod, tmp_CO, AktCislo, PredchCislo, NaslCislo);
    END IF;
  END IF;
END LOOP;
CLOSE c_CD;
PredchCislo:=AktCislo;
AktCislo:=NaslCislo;
AktKod:=NaslKod;
IF (nrozdil)>MaxRozdil THEN
  INSERT INTO K1_PodezrelaCP (StavebniObjekt, CastObce, Podezrele, Predchozi, Nasledujici)
    VALUES (AktKod, tmp_CO, AktCislo, PredchCislo, NULL);
END IF;
```

Příklad 10: Část procedury *pK1_PodezrelaCPCE*; vyhledání podezřelých čísel popisných.

2. Vyhledání podezřelých přerušení číselné řady

Pro řešení druhého přístupu ke kontrole je vytvořena procedura *pK1_PodezrelaPreruseni.* Pro ukládání podezřelých výsledků kontroly jsou vytvořeny dvě totožné tabulky, tabulka *K1_PodezrelaPrerCP* pro ukládání podezřelých přerušení souvislé číselné řady čísel popisných a *K1_PodezrelaPrerCE* pro ukládání podezřelých přerušení souvislé číselné řady čísel evidenčních. Struktura tabulek je na obrázku 17.

COLUMN_NAME	DATA_TYPE	COMMENTS
POCATECNISO	NUMBER(9,0)	-- stavebni objekt s cislem domovnim PocatecniCD
POCATECNICD	NUMBER(10,0)	-- cislo domovni, na kterem zacina preruseni souvisle ciselne rady
KONECNYSO	NUMBER(9,0)	-- stavebni objekt s cislem domovnim KonecneCD
KONECNECD	NUMBER(10,0)	-- cislo domovni, na kterem konci preruseni souvisle ciselne rady
CASTOBCE	NUMBER(10,0)	-- cast obce, na kterou maji vazbu oba stavebni objekty

Obr. 17: Struktura tabulek *K1_PodezrelaPrerCP* a *K1_PodezrelaPrerCE.*

Procedura *pK1_PodezrelaPreruseni* má vstupní parametr *MaxPreruseni*, který definuje maximální velikost přerušení souvislé číselné řady, která se může v datech vyskytnout. Pokud je tato hodnota překročena, je přerušení podezřelé. Procedura *pK1_PodezrelaPreruseni* nejprve vymaže všechny záznamy z tabulek *K1_PodezrelaPrerCP* a *K1_PodezrelaPrerCE*. Poté načte do kurzoru kódy všech částí obce a následně prochází jednotlivé části obce. Pro každou část obce jsou do kurzoru vybrány všechny stavební objekty s číslem popisným, jež mají na danou část obce vazbu a jsou seřazeny vzestupně podle čísla popisného. Z kurzoru jsou načtena první dvě čísla a je zkontrolováno, zda jejich rozdíl nepřekročí hodnotu *MaxPreruseni*. Pokud ji překročí, je vložen záznam do tabulky *K1_PodezrelaPrerCP*. Pokud rozdíl hodnotu *MaxPreruseni* nepřekročí, je z kurzoru načteno třetí číslo a je porovnáno s předchozím druhým číslem. Tento postup se opakuje až do konce číselné řady. Stejným způsobem jsou vyhledána přerušení číselné řady evidenčních čísel s tím rozdílem, že výsledky jsou ukládány do tabulky *K1_PodezrelaPrerCE*. Část procedury *pK1_PodezrelaPreruseni*, ve které jsou vyhledávána přerušení řady popisných čísel, je uvedena v příkladu 11. Při vyhledávání je pro každou část obce provedeno $k - 1$ porovnání, kde k je počet čísel v části obce. Celkově je provedeno $n - p$ porovnání, kde n je celkový počet čísel domovních a p je počet částí obce. Algoritmus tedy pracuje s výpočetní složitostí $O(n)$.

```
--kontrola cisel popisnych
 OPEN c_CO;  --postupne prochazeni jednotlivych casti obce
 LOOP
   FETCH c_CO INTO tmp_CO;
   EXIT WHEN c_CO%NOTFOUND;
   OPEN c_CD FOR  --otevreni kurzoru pro prochazeni cisel popisnych v tabulce K1_CislaPopisna
     SELECT kod, CD FROM K1_CislaDomovni
     WHERE CastObce=tmp_CO AND typSO=1
     ORDER BY CD;
   FETCH c_CD INTO tmp_CD;
   PredchCislo:=tmp_CD.CD; --aktualni cislo
   PredchKod:=tmp_CD.kod;
   LOOP
     FETCH c_CD INTO tmp_CD;
     EXIT WHEN c_CD%NOTFOUND;
     AktCislo:=tmp_CD.CD; --nasledujici cislo
     AktKod:=tmp_CD.kod;
     IF (AktCislo-PredchCislo)>MaxPreruseni THEN  --porovnani
       INSERT INTO K1_PodezrelaPrerCP (PocatecniSO, PocatecniCD, KonecnySO, KonecneCD, CastObce)
         VALUES (PredchKod, PredchCislo, AktKod, AktCislo, tmp_CO);
     END IF;
   PredchCislo:=AktCislo;
   PredchKod:=AktKod;
   END LOOP;
   CLOSE c_CD;
 END LOOP;
 CLOSE c_CO;
 dbms_output.put_line('Podezrela preruseni byla ulozena do tabulky K1_PodezrelaPrerCP');
```

Příklad 11: Část procedury *pK1_PodezrelaPreruseni*; vyhledání podezřelých přerušení v řadě popisných čísel.

6.2 Vyhodnocení kontroly

Testovací data ORP Plzeň obsahují 29 793 stavebních objektů, které mají přidělené číslo domovní. Z toho je 24 606 stavebních objektů s číslem popisným a 5 187 s číslem evidenčním. 989 stavebních objektů má přiděleno více čísel domovních. Do tabulky *K1_CislaDomovni* je vloženo 31 240 záznamů. Tato hodnota je rovna počtu čísel domovních v testovacích datech. Vyhodnocení výsledků je opět potřeba rozdělit na dvě části podle toho, jaký přístup byl použit.

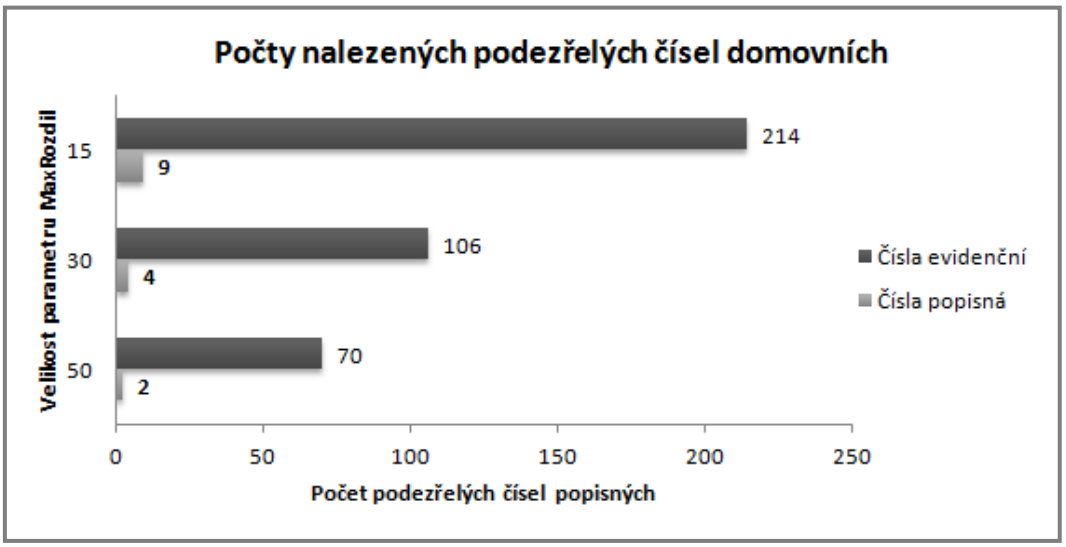
1. **Vyhledání podezřelých čísel domovních**

Pro hodnotu parametru *MaxRozdil* 15 bylo nalezeno 9 podezřelých čísel popisných. Pět z nich je na konci číselné řady. Příkladem je číslování v části obce Valcha. V této části obce je až na několik chybějících čísel popisných číselná řada spojitá až do čísla 247. Následuje mezera o velikosti 72 a posledním číslem

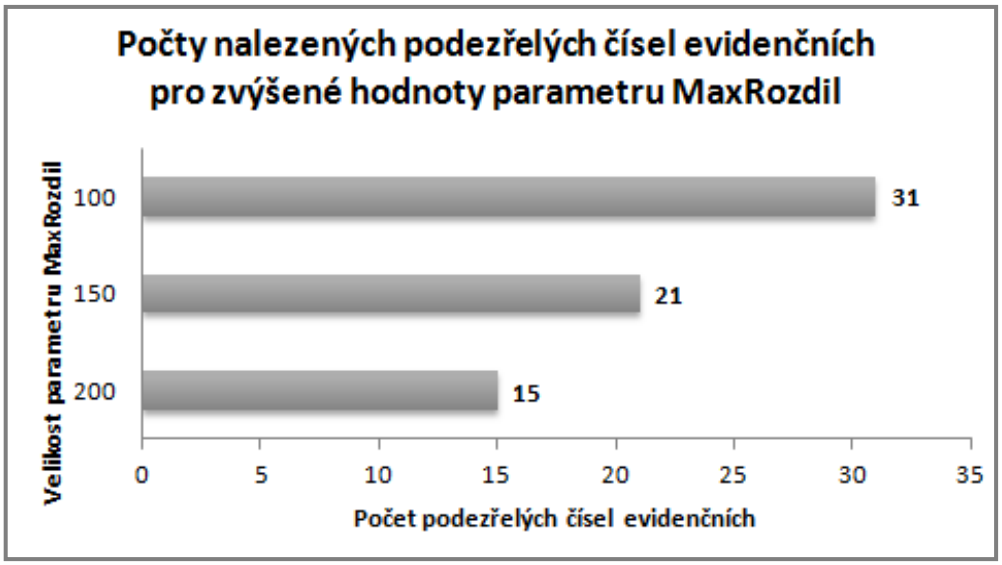
popisným je 319. Počty nalezených podezřelých čísel popisných a evidenčních pro hodnoty parametru *MaxRozdil* 15, 30 a 50 jsou uvedeny v grafu 3.

Z grafu je zřejmé, že nalezených podezřelých čísel evidenčních je výrazně víc. To je dáno tím, že čísly evidenčními jsou číslovány mimo jiné i dočasné stavební objekty, u kterých dochází ke zrušení častěji než u jiných stavebních objektů a řady čísel evidenčních tedy obsahují větší množství přerušení. Pro čísla evidenční je tedy vhodné zvolit vyšší hodnoty parametru *MaxRozdil*. Pro hodnoty 100, 150 a 200 jsou výsledky kontroly pro čísla evidenční shrnuty v grafu 4.

Graf 3: Počty nalezených čísel domovních pro různé hodnoty parametru *MaxRozdil*.



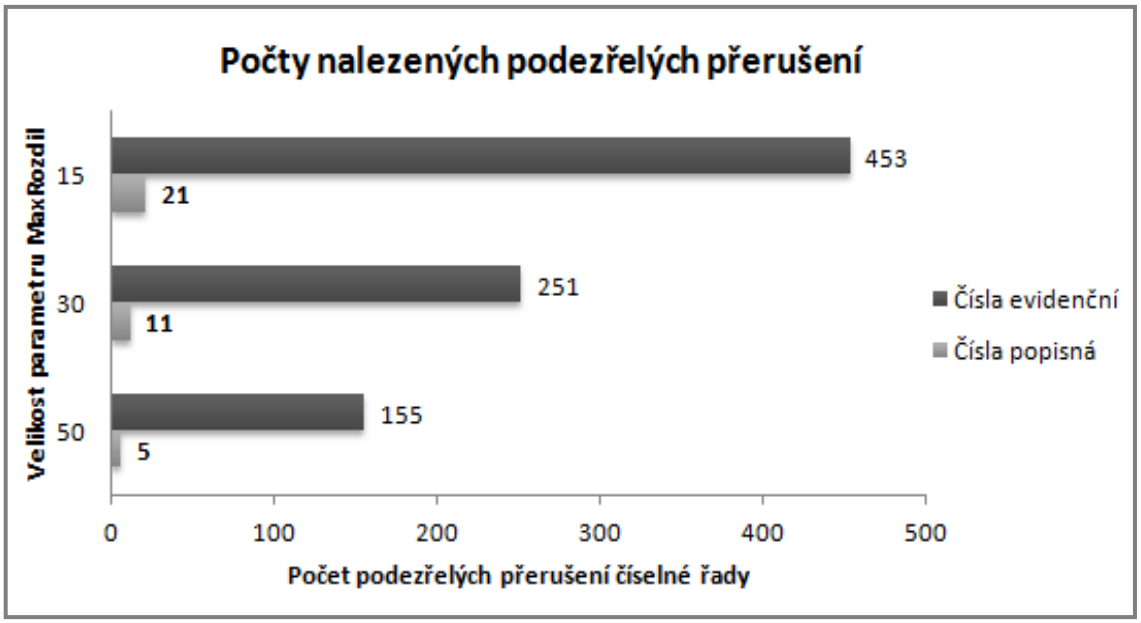
Graf 4: Počty podezřelých čísel evidenčních pro zvýšené hodnoty parametru *MaxRozdil*.



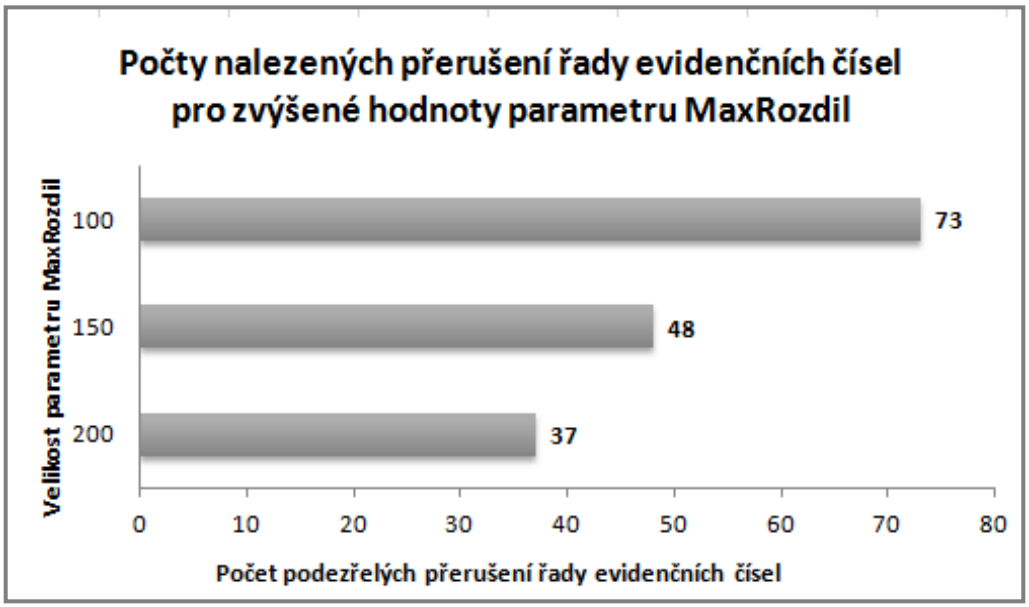
2. Vyhledání podezřelých přerušení číselné řady

Počty nalezených přerušení, které překročí hodnotu parametru *MaxPreruseni*, jsou pro čísla popisná i evidenční uvedena v grafu 5.

Graf 3: Počty nalezených přerušení číselné řady pro různé hodnoty parametru *MaxPreruseni*.



Graf 4: Počty podezřelých přerušení řady evidenčních čísel pro zvýšené hodnoty parametru *MaxPreruseni*.



Z grafu je zřejmé, že přerušení řady evidenčních čísel je výrazně víc. To je stejně jako v předchozím případě dáno tím, že čísly evidenčními jsou číslovány mimo jiné i dočasné stavební objekty, u kterých dochází ke zrušení častěji než u jiných stavebních objektů a řady čísel evidenčních tedy obsahují větší množství přerušení. Počty přerušení řady evidenčních čísel pro hodnoty 100, 150 a 200 jsou shrnuty v grafu 6.

7 Vyhledání stavebních objektů, které mají vazbu na stejnou parcelu KN

Úkolem této kontroly je vyhledat ty stavební objekty, které mají vazbu na stejnou parcelu katastru nemovitostí. Na jednom pozemku může stát více staveb. Například na pozemku, jehož druh je zastavěná plocha a nádvoří, může stát rodinný dům spolu s drobnou stavbou, například kůlnou [9], [10]. Stejně tak v katastru nemovitostí může být v souboru geodetických informací (SGI) vyznačeno více staveb na jedné parcele, ale pouze jedna z nich, takzvaná stavba hlavní, může mít vazbu do souboru popisných informací (SPI). Ostatní stavby, stavby vedlejší, které jsou příslušenstvím hlavní stavby a vazbu do SPI nemají, jsou evidovány pouze v SGI. Vedlejší stavbou je podle katastrální vyhlášky [10] budova, která je určena k tomu, aby se jí trvale užívalo s hlavní stavbou v rámci jejich hospodářského účelu, a která není drobnou stavbou. Pokud na pozemku vzniknou dvě hlavní stavby, musí být pozemek rozdělen tak, aby každá stavba byla evidována na samostatné parcele.

V databázi testovacích dat je parcela, na kterou má stavební objekt vazbu, uložena v atributu *IdentifikacniParcela*. Identifikační parcelou stavebního objektu je obvykle parcela, na které je stavební objekt evidován. Pokud stavební objekt stojí na více pozemcích, je identifikační parcelou ta, na které se nachází nejvýznamnější část stavebního objektu. Nejvýznamnější část bude pravděpodobně ve většině případů znamenat největší část, případně tu část stavebního objektu, ve které je vchod do stavebního objektu, atp. Pokud očíslovaný stavební objekt leží na hranici obcí, pak musí být identifikační parcela z toho katastrálního území, které patří do obce, respektive části obce, v rámci které mu bylo přiděleno číslo popisné či evidenční, a to i kdyby větší část stavebního objektu ležela na území sousední obce [35].

7.1 Možnosti řešení kontroly

Pro řešení tohoto problému byly vytvořeny a otestovány dvě metody, které budou podrobněji popsány a porovnány v následujících podkapitolách. Jedná se o použití kurzoru a o použití operátoru *JOIN*.

7.1.1 Použití kurzoru

První metoda používá kurzor, do kterého je načten kód a identifikační parcela stavebních objektů a pro každý záznam jsou příkazem *SELECT* vyhledávány stavební objekty, u kterých je identifikační parcela stejná, jako u aktuálně kontrolovaného stavebního objektu. Procedura, která byla vytvořena k otestování tohoto postupu, je uvedena v příkladu 12.

```
CREATE OR REPLACE PROCEDURE pK4_KURZORY AS
--procedura pro nalezeni stavebnich objektu, jez maji vazbu na stejnou parcelu KN
  CURSOR C_SO IS SELECT KOD, IDENTIFIKACNIPARCELA FROM STAVEBNIOBJEKTDFB;
  --definice typu pro ulozeni kodu, cisla domovniho a casti obce
  TYPE t_SO IS RECORD(
    kod NUMBER(9,0), --kod stavebniho objektu
    IdentifikacniParcela FLOAT); --id parcely, na kterou ma stavebni objekt vazbu
  tmp_SO t_SO; --promenna pro ukladani hodnot kurzoru c_SO1

BEGIN
    DELETE K4_SONA1PARCELE;
    OPEN c_SO;  --otevreni kurzoru pro prochazeni stavebnich objektu
    LOOP
      FETCH c_SO INTO tmp_SO;
      EXIT WHEN c_SO%NOTFOUND;
          INSERT INTO K4_KURZORY (kod_SO, ID_parcely)
            SELECT kod, IdentifikacniParcela FROM StavebniObjektDFB
            WHERE IdentifikacniParcela=tmp_SO.IdentifikacniParcela AND kod<>tmp_SO.kod;
    END LOOP;
    CLOSE c_SO;  --uzavreni kurzoru pro prochazeni stavebnich objektu
  DBMS_OUTPUT.PUT_LINE('Stavebni objekty,
    jez maji vazbu na stejnou parcelu KN, byly ulozeny do tabulky K4_KURZORY');
END;
/
```

Příklad 12: Použití kurzoru k vyhledání stavebních objektů, které mají vazbu na stejnou parcelu KN.

7.1.2 Použití operátoru JOIN

Druhá metoda využívá operátor *JOIN* a připojuje tabulku stavebních objektů samu na sebe. Operátor *JOIN* je možné použít uvnitř příkazu *SELECT*. Struktura příkazu je uvedena v příkladu 13.

```
SELECT DISTINCT SO1.kod, SO1.IdentifikacniParcela
  FROM StavebniObjektDFB SO1
  RIGHT JOIN StavebniObjektDFB SO2
  ON SO1.IDENTIFIKACNIPARCELA=SO2.IDENTIFIKACNIPARCELA
  WHERE SO1.kod<>SO2.kod;
```

Příklad 13: Použití operátoru *JOIN* k vyhledání stavebních objektů, které mají vazbu na stejnou parcelu KN.

7.1.3 Porovnání testovaných metod

Metody je opět potřebné porovnat z hlediska kvality jejich výstupu a z hlediska časového. Po spuštění na testovacích datech ORP Plzeň vrací obě metody stejné výsledky. Obě metody vyhledají 549 stavebních objektů, které mají celkem vazbu na 272 parcel. Výsledky obou metod i jejich časová náročnost jsou porovnány v tabulce 6. Z časového hlediska se jednoznačně jeví jako výhodnější použití operátoru *JOIN*.

Tabulka 6: Porovnání časové náročnosti a výsledků použití kurzoru a operátoru *JOIN*.

METODA	ČAS BĚHU	Počet stavebních objektů	Počet parcel
POUŽITÍ KURZORU	**243,28**	**549**	**272**
POUŽITÍ OPERÁTORU	**0,06**	**549**	**272**

7.2 Doporučené řešení kontroly

Pro řešení kontroly v produkčním prostředí RÚIAN byl na základě vyhodnocení provedeného testování (viz kapitola 7.1.3) vybrán postup použití operátoru *JOIN*.

Pro uložení nalezených záznamů je vytvořena tabulka *K4_SoNa1Parcele*. Atributy tabulky a její struktura jsou uvedeny na obrázku 18.

COLUMN_NAME	DATA_TYPE	COMMENTS
KOD_SO	NUMBER(9,0)	--kod stavebniho objektu
ID_PARCELY	FLOAT	--identifikator parcely, na ktreou ma vazbu vice stavebnic objektu

Obr. 18: Struktura tabulky *K4_SoNa1Parcele*.

Pro vyhledání stavebních objektů, které mají vazbu na stejnou parcelu, je vytvořena procedura *pK4_SoNa1Parcele*. Procedura nejprve smaže záznamy z tabulky *K4_SoNa1Parcele*. Dále pomocí příkazu *SELECT* za použití operátoru *JOIN* najde hledané záznamy, seřadí je vzestupně podle hodnoty ve sloupci *ID_PARCELY* a uloží je. Po proběhnutí procedury jsou v tabulce *K4_SoNa1Parcele* uloženy stavební objekty, které mají vazbu na stejnou parcelu v katastru nemovitostí tak, že jednotlivé stavební objekty s vazbou na stejnou parcelu jsou seřazeny pod sebou. Tento způsob uložení není úplně ideální, ale byl zvolen z toho důvodu, že na jednu parcelu mohou mít vazbu dva, tři i více stavebních objektů. Celá procedura *pK4_SoNa1Parcele* je uvedena v příkladu 14.

```
CREATE OR REPLACE PROCEDURE pK4_SoNa1Parcele AS
-- procedura pro vyhledani stavebnich objektu,
-- ktere maji vazbu na stejnou parcelu KN
BEGIN
-- vymazani zaznamu z pomocne tabulky
DELETE K4_SoNa1Parcele;
-- join tabulky stavebnich objektu na sebe samu
INSERT INTO K4_SoNa1Parcele -- vlozeni nalezenych do pomocne tabulky
  SELECT DISTINCT SO1.kod, SO1.IdentifikacniParcela
    FROM StavebniObjektDFB SO1
    RIGHT JOIN StavebniObjektDFB SO2
    ON SO1.IdentifikacniParcela=SO2.IdentifikacniParcela
    WHERE SO1.kod<>SO2.kod
    ORDER BY SO1.IdentifikacniParcela; -- serazeni podle parcely
END;
/
```

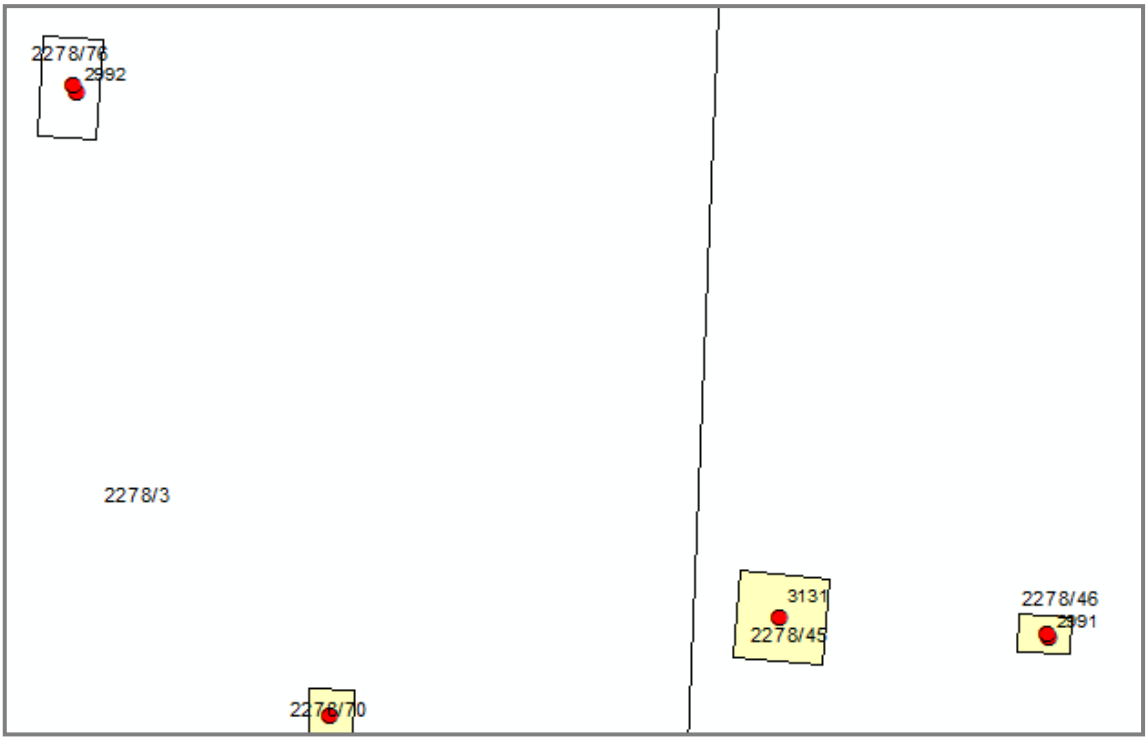
Příklad 14: Procedura *pK4_SoNa1Parcele*; procedura vyhledá stavební objekty, které mají vazbu na stejnou parcelu KN.

7.3 Vyhodnocení kontroly

Procedura odhalila v testovacích datech ORP Plzeň 272 parcel, na které má vazbu více stavebních objektů. Celkem se jedná o 549 stavebních objektů, které mají na těchto 272 parcel vazbu. V 270 případech mají na jednu parcelu vazbu dva stavební objekty, v jednom případě mají na jednu parcelu vazbu 3 stavební objekty a v jednom případě má na jednu parcelu vazbu 6 stavebních objektů. Zde se jedná

o skupinu stavebních objektů, jejichž atribut *IdentifikacniParcela* má hodnotu 0. Jedná se o stavební objekty bez vazby na parcelu.

Ukázka tří stavebních objektů, jež mají vazbu na jednu parcelu KN a příslušné záznamy v tabulce *K4_SoNa1Parcele*, je na obrázku 19. Jedná se o parcelu, jejíž *ID* je 580154405. Číslo parcely je 2278/46. Uvnitř polygonu parcely jsou dva definiční body stavebních objektů, z nichž jeden má číslo popisné 2991. Třetí definiční bod, jež má na parcelu vazbu, leží na území parcely s číslem 2278/76.

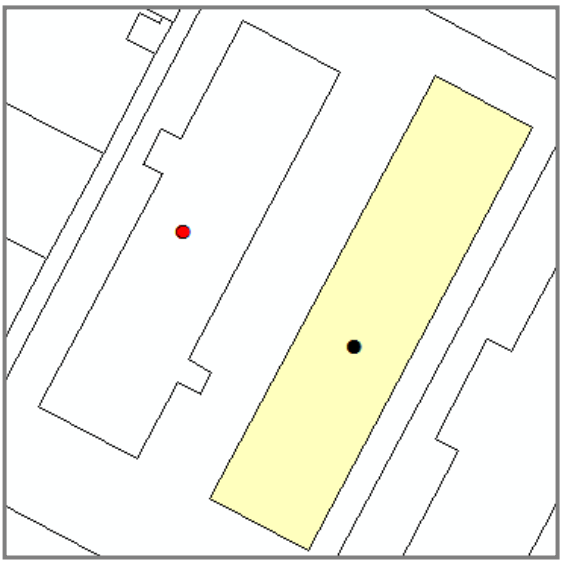


KOD_SO	ID_PARCELY	CISLADOMOVNI	CISLOPARCE
50600311	580154405	2991	2278/46
50603876	580154405	2992	2278/46
49973894	580154405	(null)	2278/46

Obr. 19: Tři stavební objekty s vazbou na jednu parcelu. Na parcelu číslo 22278/46 mají vazbu 3 stavební objekty. Jedná se o stavební objekt s číslem popisným 2991 a stavební objekt bez čísla popisného (jejich definiční body leží v polygonu parcely) a stavební objekt s číslem popisným 2992 (jeho definiční bod leží v polygonu parcely 2278/76).

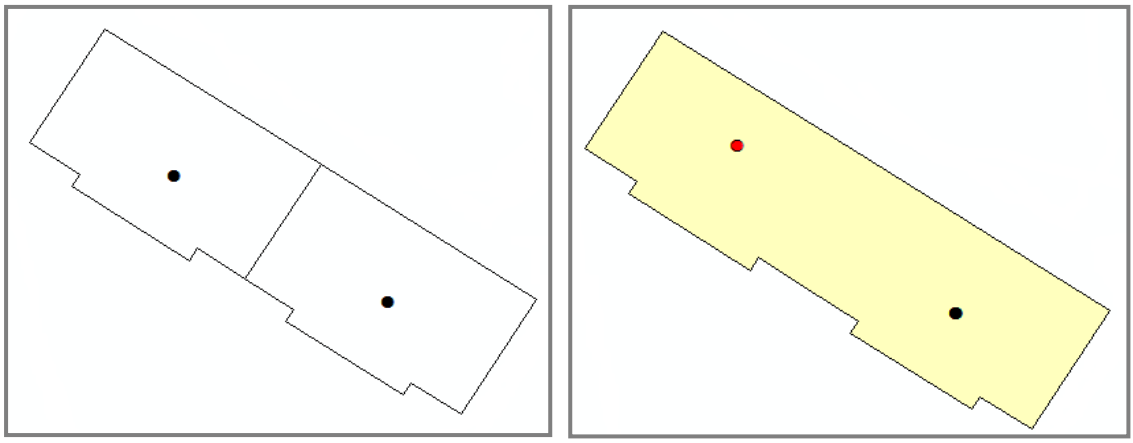
8 Vyhledání stavebních objektů, které mají vzdálený definiční bod stavebního objektu a definiční bod parcely

Kontrola má vyhledat stavební objekty, které mají definiční bod příliš vzdálený od definičního bodu parcely, na kterou mají vazbu. Jde tedy o vyhledání stavebních objektů, jejichž definiční bod neleží na parcele, na které stavební objekt stojí, tedy o vyhledání chybně umístěných definičních bodů stavebních objektů nebo parcel. Příklad hledaných situací je na obrázku 20. Jedná se o chybně umístěný definiční bod stavebního objektu (stavební objekt zvýrazněn žlutě, jeho definiční bod je červený).



Obr. 20: Definiční bod stavebního objektu vzdálený od definičního bodu parcely. Definiční bod parcely je černý, definiční bod stavebního objektu je červený.

Problém nastává u stavebních objektů, které stojí na více parcelách. Jedná se většinou o bytové domy s více adresními místy. Tyto stavební objety mají v RÚIAN vazbu pouze na jednu parcelu, na identifikační parcelu. V těchto případech se často stává, že definiční bod stavebního objektu (správně umístěný v těžišti stavebního objektu) a definiční bod jeho identifikační parcely (správně umístěný v těžišti parcely) jsou od sebe vzdáleny a definiční bod stavebního objektu leží vně identifikační parcely. Příklad situace je na obrázku 21.



Obr.21: Stavební objekt stojící na dvou parcelách. Vlevo: parcely a jejich definiční body. Vpravo: stavební objekt, jeho definiční bod (červený) a definiční bod jeho identifikační parcely (černý).

Zásadní je zde opět otázka, při jaké vzdálenosti definičního bodu stavebního objektu a parcely můžeme tyto body považovat za příliš vzdálené. Tato vzdálenost bude závislá na velikosti parcel a vzhledem k širokému rozpětí velikosti parcel je zřejmé, že nelze stanovit jednotnou mez pro všechny parcely.

8.1 Možnosti řešení kontroly

Pro řešení kontroly byly navrženy a otestovány čtyři přístupy, které se různým způsobem vypořádávají s určením maximální vzdálenosti definičních bodů stavebních objektů a parcel a s problémem stavebních objektů, které stojí na více parcelách.

8.1.1 Metoda 1

Metoda vychází z toho, že parcely, na kterých se nachází stavební objekty, rozdělí do intervalů podle velikosti výměry parcely a pro každý interval určí maximální přípustnou vzdálenost definičního bodu stavebního objektu a definičního bodu parcely. Před tvorbou samotné procedury pro testování je potřeba udělat následující kroky:

a) zjistit počet parcel, na které má vazbu stavební objekt,
b) zjistit počet intervalů, do kterých je potřeba parcely rozdělit,
c) rozdělit parcely, na které má vazbu stavební objekt, do intervalů,

d) určit maximální vzdálenost definičního bodu parcely a definičního bodu stavebního objektu, který má na parcelu vazbu.

Řešení jednotlivých kroků je následující:

a) Počet parcel, na které má vazbu stavební objekt, lze zjistit příkazem:

SELECT COUNT(DISTINCT IdentifikacniParcela) FROM StavebniObjektDFB;

b) Počet intervalů, do kterých je vhodné parcely rozdělit, je určen podle Sturgesova pravidla, které zní $k = 1 + 3{,}3 \cdot \log n$, kde n je počet prvků množiny, která je dělena do intervalů (v tomto konkrétním případě se jedná o počet parcel, na které má vazbu stavební objekt) a k je výsledný počet intervalů určený Sturgesovým pravidlem [36].

c) Pro rozdělení dat do definovaného počtu intervalů lze použít systém ArcGIS [37], který disponuje funkcí *Jenks natural breaks*, někdy nazývané také *goodness of variance fit* [38]. Jedná se o metodu shlukové analýzy, která funguje tak, že rozdělí data do definovaného počtu intervalů k tak, aby byl minimalizován rozptyl uvnitř intervalů a maximalizován rozptyl mezi intervaly. *Jenks natural breaks* funguje tak, že náhodně rozdělí data do k tříd. Dále jsou iterativně opakovány následující kroky:
   - vypočte se součet čtverců odchylek mezi třídami (OT),
   - pro každou třídu vypočte součet čtverců odchylek jednotlivých prvků třídy od průměru třídy (OP_k),
   - pro každou třídu se vypočte se rozdíl $R_k = OP_k - OT$,
   - ze třídy s nejvyšší hodnotou R_k se jeden prvek přesune směrem ke třídě s nejnižší hodnotou R_k.

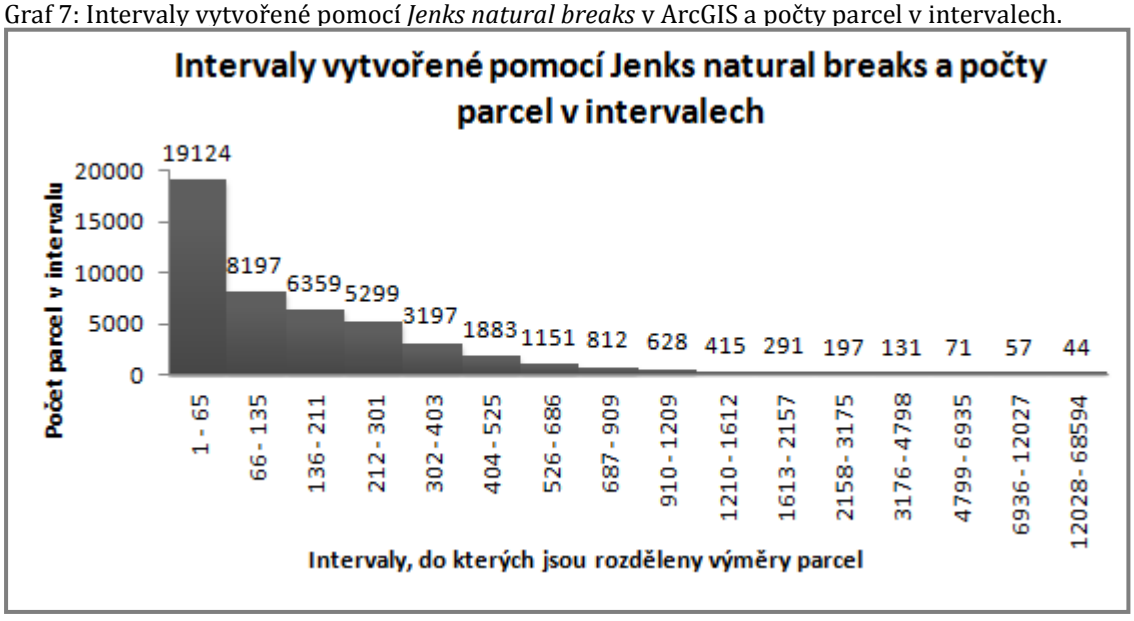
   Tento postup se opakuje, dokud rozdíl R_k nedosáhne minima [38]. Metodu navrhl americký kartograf a profesor geografie George Frederick Jenks [39]. Zjednodušeně lze říci, že jde o minimalizaci odchylek od průměru třídy uvnitř každé třídy a zároveň o maximalizaci odchylek mezi jednotlivými třídami.

d) Mezní hodnota vzdálenosti m definičního bodu stavebního objektu a jeho identifikační parcely je pak určena pro každý interval podle vzorce:

$m = \frac{\sqrt{2 \cdot s^2}}{2}$, kde $s = \frac{\sqrt{I_p} + \sqrt{I_k}}{2}$, I_p je počátek intervalu (výměra nejmenší parcely v intervalu) a I_k je konec intervalu (výměra největší parcely v intervalu). Kdyby parcely s výměrou I_p a I_k měly čtvercový tvar, bylo by s průměrem z velikosti strany těchto dvou čtverců a mezní vzdálenost m by se rovnala polovině úhlopříčky čtverce se stranou velikosti s. Tento způsob určení mezní vzdálenosti vychází z umístění definičního bodu parcely zhruba v jejím těžišti. Je vhodný pro parcely přibližně čtvercového tvaru, ale není ideální pro úzké dlouhé parcely.

Počet parcel, na které má vazbu stavební objekt, je 48136. Sturgesovým pravidlem je určeno 16 intervalů, do kterých je potřeba rozdělit parcely podle velikosti jejich výměry. Hranice intervalů, určené pomocí *Jenks natural breaks*, a počty parcel, které padnou do jednotlivých intervalů, jsou uvedeny v grafu 7. Interpretace grafu je zřejmá, například do prvního intervalu spadají parcely s výměrou 1 - 65 metrů a počet parcel v prvním intervalu je 19124, do druhého intervalu spadají parcely o výměře 66 – 135 metrů, počet parcel ve druhém intervalu je 8197, atd.

Graf 7: Intervaly vytvořené pomocí *Jenks natural breaks* v ArcGIS a počty parcel v intervalech.



Výsledné mezní hodnoty vzdálenosti definičního bodu parcely a definičního bodu stavebního objektu, který má na parcelu vazbu, jsou pro jednotlivé intervaly uvedeny v tabulce 7.

Tabulka 7: Mezní vzdálenosti definičního bodu stavebního objektu a parcely.

INTERVAL	Mezní vzdálenost [m]	INTERVAL	Mezní vzdálenost [m]
1 - 65	3,2	**910 - 1209**	23,0
66 - 135	7,0	**1210 - 1612**	26,5
136 - 211	9,3	**1613 - 2157**	30,6
212 - 301	11,3	**2158 - 3175**	36,3
302 - 403	13,2	**3176 - 4798**	44,4
404 - 525	15,2	**4799 - 6935**	53,9
526 - 686	17,4	**6936 - 12027**	68,2
687 - 909	19,9	**12028 - 68594**	131,4

Problém stavebních objektů stojících na více parcelách je zde řešen prostřednictvím adresních míst. Pokud na stavební objekt má vazbu více adresních míst a mezní vzdálenost definičního bodu stavebního objektu a definičního bodu parcely je překročena, předpokládá se, že v tomto případě se o chybu nejedná.

Samotná procedura v PL/SQL je řešena pomocí kurzoru, do kterého se vybere kód stavebního objektu, *ID* parcely, výměra parcely, geometrie definičního bodu stavebního objektu a geometrie definičního bodu parcely. V deklarační části procedury jsou dále definovány mezní vzdálenosti pro jednotlivé intervaly. Pro každý záznam kurzoru je pak na základě velikosti výměry parcely rozhodnuto, do jakého spadá parcela intervalu, tedy jaká mezní vzdálenost je pro něj určující. Funkcí *SDO_GEOM.SDO_DISTANCE* je vypočtena skutečná vzdálenost definičního bodu stavebního objektu a definičního bodu parcely a je porovnána s mezní vzdáleností. Pokud skutečná vzdálenost mezní vzdálenost překročí, je příslušný objekt považován za chybný. Část procedury je uvedena v příkladu 15.

Tento postup odhalí 187 nevyhovujících stavebních objektů. Ve 24 případech je mezní vzdálenost překročena o méně než 10%, v 55 případech je překročena o méně než 20%. Dá se předpokládat, že v některých z těchto případů se o chybu jednat nemusí, ale objekty byly označeny za chybné z důvodu přísné

mezní vzdálenosti. Naopak můžou existovat chyby, které tímto postupem nebudou odhaleny.

```
OPEN c_SO;  --otevreni kurzoru pro prochazeni stavebnich objektu
LOOP
  FETCH c_SO INTO tmp_SO;
  EXIT WHEN c_SO%NOTFOUND;
  IF (tmp_SO.vymera>=1) AND (tmp_SO.vymera<=65) THEN -- vymera parcely v intervalu 1-65m
    vzdal := SDO_GEOM.SDO_DISTANCE (tmp_SO.SOgeom, tmp_SO.Pgeom, tol, NULL);
    IF (vzdal>mez1) THEN
      SELECT count(kod) INTO PocAdrMist
        FROM AdresniMisto
        WHERE stavebniob=tmp_SO.kod;
      INSERT INTO K2_Metoda1 (StavebniObjekt, Parcela, Vzdalenost, Mez, PocetAdresnichMist)
        VALUES (tmp_SO.kod, tmp_SO.IdentifikacniParcela, vzdal, mez1, PocAdrMist);
    END IF;
   ELSIF (tmp_SO.vymera>=66) AND (tmp_SO.vymera<=135) THEN -- vymera parcely v intervalu 66-135m
    vzdal := SDO_GEOM.SDO_DISTANCE (tmp_SO.SOgeom, tmp_SO.Pgeom, tol, NULL);
    IF (vzdal>mez2) THEN
      SELECT count(kod) INTO PocAdrMist
        FROM AdresniMisto
        WHERE stavebniob=tmp_SO.kod;
     INSERT INTO K2_Metoda1 (StavebniObjekt, Parcela, Vzdalenost, Mez, PocetAdresnichMist)
       VALUES (tmp_SO.kod, tmp_SO.IdentifikacniParcela, vzdal, mez2, PocAdrMist);
    END IF;
```

Příklad 15: Vyhledání stavebních objektů, které mají definiční bod stavebního objektu a definiční bod parcely vzdálený více než definovaná maximální vzdálenost pomocí metody 1.

8.1.2 Metoda 2

Druhá metoda vychází z předpokladu, že lepších výsledků bude dosaženo, pokud se stanoví mezní vzdálenost definičních bodů stavebního objektu a parcely pro každou parcelu individuálně, opět na základě velikosti její výměry. Pro určení mezní vzdálenosti je použit obdobný postup jako při určení mezní vzdálenosti u metody 1. Mezní vzdálenost m je vypočtena ze vzorce: $m = \sqrt{v}$, kde v je výměra parcely. Pokud by parcely měly tvar čtverce, pak mezní vzdálenost m bude rovna velikosti strany tohoto čtverce.

Problém stavebních objektů stojících na více parcelách je zde řešen stejným způsobem jako u metody 1. To znamená, že pokud má na stavební objekt vazbu více adresních míst, a mezní vzdálenost definičního bodu stavebního objektu a definičního bodu parcely je překročena, předpokládá se, že v tomto případě se o chybu nejedná.

Procedura je řešena následujícím způsobem. Je vytvořen kurzor, do kterého se vybere kód stavebního objektu, *ID* parcely, výměra parcely, geometrie definičního bodu stavebního objetu a geometrie definičního bodu parcely. Pro každý záznam kurzoru je v závislosti na velikosti výměry parcely vypočtena mezní vzdálenost podle výše uvedeného vzorce. Skutečná vzdálenost definičního bodu stavebního objektu a definičního bodu parcely je vypočtena pomocí funkce *SDO_GEOM.SDO_DISTANCE*. Skutečná vzdálenost je porovnána s mezní vzdáleností a v případě, že je mezní vzdálenost překročena, je příslušný objekt považován za chybný. Stěžejní část procedury je uvedena v příkladu 16.

```
OPEN c_SO;  --otevreni kurzoru pro prochazeni stavebnich objektu
LOOP
  FETCH c_SO INTO tmp_SO;
  EXIT WHEN c_SO%NOTFOUND;
  strana:=SQRT(tmp_SO.vymera);
  mez:=SQRT(strana*strana);
  vzdal := SDO_GEOM.SDO_DISTANCE (tmp_SO.SOgeom, tmp_SO.Pgeom, tol, NULL);
    IF (vzdal>mez) THEN
      SELECT count(kod) INTO PocAdrMist
        FROM AdresniMisto
        WHERE stavebniob=tmp_SO.kod;
      INSERT INTO K2_Metoda2 (StavebniObjekt, Parcela, Vzdalenost, Mez, PocetAdresnichMist)
        VALUES (tmp_SO.kod, tmp_SO.IdentifikacniParcela, vzdal, mez, PocAdrMist);
    END IF;
  END LOOP;
  CLOSE c_SO;
```

Příklad 16: Vyhledání stavebních objektů, které mají definiční bod stavebního objektu a definiční bod parcely vzdálený více než definovaná maximální vzdálenost pomocí metody 2.

8.1.3 Metoda 3

Metoda 3 se liší od metody 2 způsobem řešení stavebních objektů, na které má vazbu více adresních míst.

Problém stavebních objektů, které stojí na více parcelách, je možné řešit tak, že se při stanovení mezní vzdálenosti nebude vycházet z výměry parcely, ale z velikosti stavebního objektu. Velikost stavebního objektu je v databázi uložena v tabulce stavebních objektů ve sloupci *zastavenap*. Tento atribut je ale vyplněn pouze u 3788 stavebních objektů, to znamená zhruba u 8% z celkového počtu stavebních objektů v testovacích datech. Vzhledem k tomu, že 46268 stavebních objektů v testovacích datech je určeno polygonem, lze hodnotu tohoto atributu

vypočíst pomocí geometrické funkce *SDO_GEOM.SDO_AREA* (více o funkci v [32]). Tímto výpočtem by se počet stavebních objektů, u kterých by byl atribut vyplněn, zvýšil téměř na 96%. Tato možnost ale nebyla použita, protože problém je možné řešit efektivněji, jak je popsáno dále.

Další možností je vycházet z toho, že v těchto případech je správné, když definiční bod identifikační parcely leží uvnitř polygonu stavebního objektu v případě, že na území existuje digitální katastrální mapa (DKM), případně katastrální mapa digitalizovaná (KMD). Tam, kde DKM a KMD chybí, jsou tyto stavební objekty označeny za chybné. Metoda 3 tedy spočítá pro každou parcelu mezní vzdálenost stejným způsobem jako metoda 2. V případě, že je mezní vzdálenost překročena, zjistí se počet adresních míst, které mají vazbu na příslušný stavební objekt. Pokud je jejich počet nula nebo jedna, je stavební objekt považován za chybný. Pokud je počet adresních míst větší než jedna a stavební objekt je určen polygonem, je kontrolován průnik polygonu stavebního objektu a definičního bodu parcely. Pokud definiční bod parcely neleží uvnitř polygonu stavebního objektu, je stavební objekt považován za chybný. V případě, že budova není určena polygonem, je stavební objekt automaticky považován za chybný.

Procedura funguje následujícím způsobem. Nejprve je vytvořen kurzor, do kterého je vybrán kód stavebního objektu, jeho identifikační parcelu, geometrii definičního bodu stavebního objektu a geometrii polygonu stavebního objektu, *ID* parcely, výměru parcely a geometrii definičního bodu parcely. Vytvoření kurzoru je řešeno pomocí operátoru *JOIN*, kterým jsou spojeny tři tabulky, obsahující potřebné údaje. Vytvoření kurzoru je uvedeno ve skriptu 16.

```
CURSOR C_SO IS SELECT SOD.KOD, SOD.IDENTIFIKACNIPARCELA,
  SOD.geom, SO.geom, PD.Id, PD.vymeraparc, PD.geom
    FROM StavebniObjektDFB SOD
      LEFT JOIN StavebniObjekt SO
        ON SOD.kod=SO.kod
      FULL JOIN ParcelaDFB PD
        ON SOD.IdentifikacniParcela=PD.Id; -- definice kurzoru
```

Příklad 17: Definice kurzoru, který je vytvořen pomocí operátoru *JOIN* přes tři tabulky.

Dále je pro každý záznam kurzoru stejným způsobem jako v metodě 2 určena mezní vzdálenost a vypočtena skutečná vzdálenost definičního bodu stavebního objektu a definičního bodu parcely. Pokud je mezní vzdálenost překročena, je zjištěn počet adresních míst, které mají na stavební objekt vazbu. Můžou nastat tři situace, které jsou řešeny následujícími způsoby:

Způsob 1: Na stavební objekt nemá vazbu žádné adresní místo nebo na něj má vazbu pouze jedno adresní místo. V tomto případě je stavební objekt označen za chybný.

Způsob 2: Na stavební objekt má vazbu více adresních míst, ale stavební objekt není určen polygonem. V tomto případě je stavební objekt také označen za chybný.

Způsob 3: Na stavební objekt má vazbu více než jedno adresní místo a stavební objekt je určen polygonem. V tomto případě musí definiční bod parcely ležet uvnitř polygonu stavebního objektu. Ke zjištění, zda je tomu tak, je použita geometrická funkce *SDO_GEOM.RELATE*. Funkce má následující povinné parametry:

- *GEOM1* obsahuje geometrii vstupního objektu,
- *MASK* definuje prostorový vztah vstupních geometrií,
- *GEOM2* obsahuje geometrii druhého vstupního objektu,
- *TOL* je hodnota tolerance (podobně jako u funkce *SDO_DISTANCE*), která určuje, při jaké vzdálenosti jsou dva body ještě považovány za totožné [32].

Zde konkrétně je za parametr *GEOM1* dosazován polygon stavebního objektu, jako maska je použito *ANYINTERACT* a za parametr *GEOM2* je dosazována geometrie definičního bodu parcely. Hodnota tolerance je volena 0,001. Při použití masky *ANYINTERACT* vrací funkce hodnotu *TRUE*, pokud spolu dva zkoumané objekty nemají žádný prostorový vztah. Část procedury je uvedena v příkladu 18.

```
LOOP
  FETCH c_SO INTO tmp_SO;
  EXIT WHEN c_SO%NOTFOUND;
  strana:=SQRT(tmp_SO.vymera);
  mez:=SQRT(strana*strana);
  vzdal := SDO_GEOM.SDO_DISTANCE (tmp_SO.Pgeom, tmp_SO.SOgeom, tol, NULL);
    IF (vzdal>mez) THEN
      SELECT count(kod) INTO PocAdrMist
        FROM AdresniMisto
        WHERE stavebniob=tmp_SO.kod;
        IF (POCADRMIST<=1) THEN
          INSERT INTO K2_METODA3 (StavebniObjekt, Parcela, Vzdalenost, Mez, PocetAdresnichMist, zpusob)
            VALUES (tmp_SO.kod, tmp_SO.IdentifikacniParcela, vzdal, mez, PocAdrMist, 1);
        ELSIF (TMP_SO.SOPOLYGON IS NULL AND POCADRMIST>1) THEN
            INSERT INTO K2_METODA3 (StavebniObjekt, Parcela, Vzdalenost, Mez, PocetAdresnichMist, zpusob)
              VALUES (tmp_SO.kod, tmp_SO.IdentifikacniParcela, vzdal, mez, PocAdrMist, 2);
        ELSE
          IF (SDO_GEOM.RELATE(TMP_SO.SOPOLYGON, 'mask=anyinteract', TMP_SO.PGEOM, TOL)='TRUE') THEN
              INSERT INTO K2_METODA3 (StavebniObjekt, Parcela, Vzdalenost, Mez, PocetAdresnichMist, zpusob)
                VALUES (tmp_SO.kod, tmp_SO.IdentifikacniParcela, vzdal, 0, PocAdrMist, 3);
          END IF;
        END IF;
    END IF;
END LOOP;
```

Příklad 18: Vyhledání stavebních objektů, které mají definiční bod stavebního objektu a definiční bod parcely vzdálený více než definovaná maximální vzdálenost pomocí metody 3.

Tímto způsobem je v datech nalezeno podezřelých 165 objektů.

8.1.4 Metoda 4

Poslední metoda vychází z toho, že jakkoli určená mezní vzdálenost definičního bodu stavebního objektu a definičního bodu parcely je vždy zavádějící a kvůli různým tvarům parcel ji není možné určit tak, aby nedocházelo k tomu, že budou kontrolou vyhledány i objekty, u kterých se nejedná o chyby. Je tedy nejvhodnější problém řešit pomocí polygonů parcel a ve všech případech, kde existuje DKM, případně KMD, budou za chybné označeny ty situace, kde definiční bod stavebního objektu neleží uvnitř polygonu parcely. Problém stavebních objektů, na které má vazbu více adresních míst, bude řešen stejným způsobem jako v metodě 3, to znamená, že v těchto případech musí ležet definiční bod identifikační parcely uvnitř polygonu stavby. Tam, kde digitální mapa neexistuje, se použije postup z metody 2, tedy pro každou parcelu se spočte mezní vzdálenost z výměry parcely. Stavební objekty, na které má vazbu více adresních míst a je u nich překročena mezní vzdálenost, budou v oblastech bez digitální mapy označeny za chybné.

Kontrola je pak řešena procedurou *pK2_SOMimoParcelu*. O tom, zda bude konkrétní stavební objekt označen za chybný, se rozhoduje čtyřmi způsoby. Pro každý stavební objekt je způsob vybrán podle toho, jestli je parcela, na kterou má stavební objekt vazbu, definovaná polygonem, jestli je stavební objekt definován polygonem a podle počtu adresních míst. Způsoby jsou následující:

Způsob 1: Parcela je určena polygonem a na stavební objekt má vazbu pouze jedno adresní místo. V tomto případě musí definiční bod stavebního objektu ležet uvnitř polygonu parcely. Ke zjištění, zda je tomu tak, je použita geometrická funkce *SDO_GEOM.RELATE*. Zde konkrétně je za parametr *GEOM1* dosazován polygon parcely, jako maska je použito *COVERS + CONTAINS* a za parametr *GEOM2* je dosazována geometrie definičního bodu stavebního objektu. Hodnota tolerance je volena 0,001. Při použití masky *CONTAINS* vrací funkce hodnotu *CONTAINS*, pokud druhý objekt leží uvnitř prvního objektu a jejich hranice se nedotýkají. Jinak vrací funkce hodnotu *FALSE*. Při použití masky *COVERS* vrací funkce hodnotu *COVERS*, pokud druhý objekt leží uvnitř prvního objektu a jejich hranice se dotýkají v jednom nebo více bodech. Jinak vrací funkce hodnotu *FALSE*. Použití obou masek tedy znamená, že jsou vyhledány všechny stavební objekty, jejichž definiční bod neleží uvnitř polygonu parcely nebo na hranici parcely. Tyto objekty jsou označeny za chybné.

Způsob 2: Je použit v případě, že parcela není definovaná polygonem a na stavební objekt má vazbu jen jedno adresní místo. V tomto případě je vypočtena mezní vzdálenost definičního bodu parcely a definičního bodu stavebního objektu podle vzorce používaného metodou 2 (kapitola 8.1.2). Pomocí funkce *SDO_DISTANCE* je vypočtena skutečná vzdálenost definičního bodu stavebního objektu a parcely. Pokud vypočtená vzdálenost překračuje mezní vzdálenost, je objekt označen za chybný

Způsob 3: Zde se jedná o případ, kdy na stavební objekt má vazbu více adresních míst a předpokládá se tedy, že stojí na více parcelách. Stavební objekt je navíc definován polygonem. V tomto případě musí definiční bod parcely ležet uvnitř polygonu stavebního objektu. To je kontrolováno opět funkcí

SDO_GEOM.RELATE. Za parametr *GEOM1* je dosazován polygon stavebního objektu, jako maska je použito opět *COVERS + CONTAINS* a parametr *GEOM2* je definiční bod parcely. Pokud definiční bod parcely neleží uvnitř polygonu stavebního objektu nebo na jeho hranici, je objekt označen za chybný.

Způsob 4: Na stavební objekt má vazbu více adresních míst a stavební objekt není definován polygonem. V tomto případě je objekt rovnou označen za chybný.

Způsoby 1, 2 a 3 jsou uvnitř procedury řešeny pomocí příkazu *SELECT*, kterým jsou vždy pro daný způsob vybrány podezřelé stavební objekty a jejich identifikační parcely, případně vzdálenost a dopustná vzdálenost u způsobu 2. Příkazem *UPDATE* je pak nalezeným objektům doplněn způsob řešení. Část procedury *pK2_SOMimoParcelu*, která řeší způsob 1, 2 a 3, je uvedena v příkladu 19.

```
--zpusob 1, definicni bod stavebniho objektu musi lezet v polygonu parcely
INSERT INTO Pokus (StavebniObjekt, Parcela)
  SELECT SOD.kod, SOD.IdentifikacniParcela
  FROM StavebniObjektDFBKM SOD, ParcelaKM P
  WHERE P.geom IS NOT NULL AND SOD.IdentifikacniParcela=P.Id
  AND (SELECT COUNT(AM.kod) FROM  AdresniMistoKM AM WHERE AM.stavebniob=SOD.kod)<=1
  AND SDO_GEOM.RELATE(P.geom, 'mask=COVERS+CONTAINS', SOD.geom, 0.005)='FALSE';
UPDATE Pokus SET zpusob=1
  where zpusob IS NULL;
-- zpusob 2, urceni dopustne vzdalenosti z vymery parcely
INSERT INTO Pokus (StavebniObjekt, Parcela, Vzdalenost, Mez)
  SELECT SOD.kod, SOD.IdentifikacniParcela,
    SDO_GEOM.SDO_DISTANCE (P.geom, SOD.geom, 0.005, NULL),SQRT(P.vymeraparc)
  FROM StavebniObjektDFBKM SOD, ParcelaDFBKM P
  WHERE SOD.IdentifikacniParcela=P.Id AND P.Id NOT IN (SELECT Id FROM Parcela)
  AND (SELECT COUNT(AM.kod) FROM  AdresniMistoKM AM WHERE AM.stavebniob=SOD.kod)<=1
  AND SDO_GEOM.SDO_DISTANCE (P.geom, SOD.geom, 0.005, NULL)>SQRT(P.vymeraparc);
UPDATE Pokus SET zpusob=2
  where zpusob IS NULL;
--zpusob 3, definicni bod parcely musi lezet v polygonu stavebniho objektu
INSERT INTO Pokus (StavebniObjekt, Parcela)
  SELECT SO.kod, SO.IdentifikacniParcela
  FROM StavebniObjektKM SO, ParcelaDFBKM P
  WHERE SO.geom IS NOT NULL AND SO.IdentifikacniParcela=P.Id
  AND (SELECT COUNT(AM.kod) FROM  AdresniMistoKM AM WHERE AM.stavebniob=SO.kod)>1
  AND SDO_GEOM.RELATE(SO.geom, 'mask=COVERS+CONTAINS', P.geom, 0.005)='FALSE';
UPDATE Pokus SET zpusob=3
  where zpusob IS NULL;
```

Příklad 19: Procedura *pK2_SoMimoParcelu*, řešení způsobů 1, 2, 3.

Způsob 4 je pak řešen pomocí totožného kurzoru jako v metodě 3. Část procedury, která řeší způsob 4, je uvedena v příkladu 20. Před spuštěním procedury je pro snížení časové náročnosti potřeba nad tabulkami stavebních objektů a parcel, které obsahují definiční body a polygony, vytvořit prostorový index (postup je popsán v kapitole 5.3.2).

```
--reseni zpusobu 4
OPEN c_SO;  --otevreni kurzoru pro prochazeni stavebnich objektu
LOOP
  FETCH c_SO INTO tmp_SO; --nacitani hodnot z kurzoru
  EXIT WHEN c_SO%NOTFOUND;
  SELECT count(kod) INTO PocAdrMist --vyber pootu adresnich mist
    FROM AdresniMisto
    WHERE stavebniob=tmp_SO.kod;
  --pocet adresnich mist>nez 1 a parcela nema polygon
  IF (PocAdrMist>1) AND (tmp_SO.SOPolygon IS NULL) THEN
      vzdal := SDO_GEOM.SDO_DISTANCE (tmp_SO.SOgeom, tmp_SO.Pgeom, 0.005, NULL);
        INSERT INTO K2_SoMimoParcelu (StavebniObjekt, Parcela, Vzdalenost, Mez, zpusob)
          VALUES (tmp_SO.kod, tmp_SO.IdentifikacniParcela, vzdal, mez, 4);
  END IF;
END LOOP;
CLOSE c_SO;
```

Příklad 20: Příklad 19: Procedura *pK2_SoMimoParcelu*, řešení způsobů 4.

Pro uložení chybných objektů je vytvořena tabulka *K2_SOMimoParcelu*. Struktura tabulky a její atributy je na obrázku 22. Parametr *mez* je vyplněn pouze při použití způsobů 2 a 4.

COLUMN_NAME	DATA_TYPE	COMMENTS
STAVEBNIOBJEKT	NUMBER	-- stavebni objekt
PARCELA	NUMBER	-- parcela, na kterou ma stavebni objekt vazbu
VZDALENOST	NUMBER	-- vzdalenost definicniho bodu stavebniho objektu a parcely
MEZ	NUMBER(3,1)	-- mezni hodnota vzdalenosti
ZPUSOB	NUMBER(2,0)	-- kod pouziteho zpusobu, kterym se rozhoduje

Obr. 22: Struktura tabulky *K2_SOMimoParcelu.*

Tento postup odhalí 322 nevyhovujících stavebních objektů, z nichž 290 je v oblasti s digitální mapou. O těchto 290 objektech lze s jistotou tvrdit, že v těchto oblastech leží definiční bod stavebního objektu mimo polygon parcely, na kterou má stavební objekt vazbu. Ve 2 případech dochází k tomu, že na stavební objekt má vazbu více adresních míst a defininčí bod parcely neleží uvnitř polygonu stavebního objektu. Případů, kdy v oblasti neexistuje digitální mapa, na stavební objekt má vazbu nula nebo jedno adresní místo a je překročena mezní vzdálenost,

je 7. Ve 23 případech neexistuje digitální mapa a na stavební objekt má vazbu více než jedno adresní místo. V těchto 23 případech tedy nelze jednoznačně rozhodnout, zda se jedná o chybu.

8.2 Porovnání metod a doporučení kontroly

Čtyři výše popsané metody byly porovnány ze dvou různých hledisek, a to z hlediska časové náročnosti metody a podle kvality jejích výsledků. Časová náročnost je shrnuta v tabulce 5. Je zřejmé, že metody, které neprovádí výpočty z geometrie, popřípadě je provádí pouze v případě stavebních objektů, které leží na více parcelách (metoda 3), fungují velmi rychle. Naopak metoda 4, která provádí geometrické výpočty pro většinu objektů, běží asi tři minuty, jak je uvedeno v tabulce 8.

Tabulka 8: Časová náročnost metod 1, 2, 3, 4.

METODA	ČAS BĚHU [s]
METODA 1	15,4
METODA 2	15,3
METODA 3	28,3
METODA 4	42,4

Výsledky jednotlivých metod jsou shrnuty v tabulce 6. Tabulka na diagonále uvádí počty objektů, které jsou vyhledány pomocí jednotlivých metod. Dále jsou v ní uvedeny počty objektů, které jsou shodně vyhledány dvojicemi různých metod. To znamená, že například metoda 1 označila 187 objektů za chybné, metoda 2 označila 164 objektů za chybné, ale pouze 124 z těchto objektů bylo za chybné označeno současně metodou 1 i 2.

Je překvapivé, že metoda 1 a metoda 2 vyhledají podobný počet objektů, ale pouze 124 objektů je shodně vyhledáno oběma metodami. Předpoklad, že metoda 2 bude dávat lepší výsledky než metoda 1, se nepotvrdil. Nepřekvapí téměř totožný počet objektů, vyhledaný metodou 2 a metodou 3, protože tyto metody pracují shodně, pouze metoda 3 je rozšířena o řešení stavebních objektů, na které má vazbu více adresních míst. Překvapivé jsou výsledky metody 4, která vyhledá 322

objektů, což je téměř dvakrát více, než kolik vyhledají ostatní metody. Výsledky metody 4 jsou ale nejspolehlivější, protože všude, kde je to možné, pracuje s polygony parcel, případně stavebních objektů. Použití metod 1, 2, 3 se tedy z hlediska výsledků jeví jako nevhodné, protože velké množství chyb nenaleznou a jak lze vidět z tabulky 9, označí za chybné některé objekty, které chybné nejsou. U těchto metod nepomůže ani změna mezních vzdáleností, protože při zvětšení mezních vzdáleností pro intervaly u metody 1, případně pro jednotlivé parcely u metod 2 a 3 sice vzroste počet nalezených objektů, ale zároveň vzroste počet objektů, které jsou onačeny za chybné, ale chybné nejsou.

Tabulka 9: Počty vyhledaných objektů, které jsou totožné pro metody 1, 2, 3, 4.

METODA	METODA 1	METODA 2	METODA 3	METODA 4
METODA 1	187	124	124	129
METODA 2	124	164	164	126
METODA 3	124	164	165	127
METODA 4	129	126	127	322

Metoda 4 je tedy časově náročnější než zbývající metody, ale na rozdíl od ostatních metod dává spolehlivé výsledky. Vzhledem k tomu, že pro kontroly je stěžejní, aby výsledky byly co nejkvalitnější, to znamená, aby kontroly odhalily co nejvíce chyb a zároveň aby co nejméně správných objektů označily za chybné, je pro použití v prostředí RÚIAN doporučen postup metody 4 i přes její vyšší časovou náročnost.

8.3 Vyhodnocení kontroly

Procedura *pK2_SOMimoParcelu* vyhledá 322 nevyhovujících stavebních objektů. U 290 z nich je rozhodováno způsobem 1. U 7 objektů je rozhodnuto pomocí způsobu 2, ve 2 případech bylo rozhodováno způsobem 3 a u 23 objektů je rozhodnuto způsobem 4.

Je zřejmé, že u objektů, u kterých bylo rozhodnuto způsobem 1 a 3, je možné tvrdit s jistotou, že je definiční bod stavebního objektu chybně umístěn. U způsobu 2 to s jistotou tvrdit nelze a v případě, že je zde mezní hodnota překročena

minimálně, je vhodná kontrola reálného stavu. Stejně tak u způsobu 4 je nutný další rozbor situace, případně kontrola reálného stavu. Způsobem 4 jsou v podstatě vyhledány případy, u kterých nelze s jistotou rozhodnout o chybovosti dat.

9 Testování a diskuze doporučených řešení kontrol nad daty všech krajských měst

Testovací sada dat všech krajských měst je popsána v kapitole 4. Nad touto datovou sadou byla testována všechna doporučená řešeníkontrol

9.1 Diskuze výsledků vyhledání stavebních objektů, které mají blízké definiční body

Při kontrole blízkých definičních bodů je pro různé hodnoty parametru *MinVzdal* nalezeno množství blízkých dvojic. Mezi vyhledanými objekty jsou opět dvojice, kdy oba stavební objekty mají evidovaný způsob využití garáž. Dále je nalezeno velké množství dvojic, kdy jeden ze stavebních objektů má *ID* identifikační parcely rovno 0, to znamená, že nemá vazbu na parcelu. Počty vyhledaných dvojic pro některé hodnoty parametru *MinVzdal* jsou shrnuty v tabulce 10.

Mezi dvojicemi blízkých bodů se opět nachází množství dvojic, kde oba ze stavebních objektů mají způsob využití garáž. Překvapivě velké množství je i dvojic, kdy alespoň jeden ze stavebních objektů nemá vazbu na parcelu (atribut *IdentifikacniParcela* je 0). Vzhledem k tomu, že v datové sadě ORP Plzně bylo takových stavebních objektů 6 a datová sada krajských měst je cca desetkrát větší, dalo by se předpokládat, že zde takových objektů budou řádově desítky. V datové sadě krajských měst je ale takových stavebních objektů 4814.

Časová náročnost vytvoření prostorového indexu nad daty definičních bodů stavebních objektů všech krajských měst je asi 60 s. Časová náročnost vyhledávání blízkých bodů pomocí procedury *pK3_BlizkeDFB* pro různé vstupní hodnty je uvedena v tabulce 10. Je zřejmé, že oproti testování procedury na datech ORP Plzeň čas běhu procedury prudce vzrostl. Vzhledem k růstu časové náročnosti procedury spolu s velikostí dat by bylo pravděpodobně vhodné provádět vyhledávání blízkých bodů v rámci menších územních celků, nejlépe v rámci krajů. Hranice krajů procházejí extravilánem, dá se tedy předpokládat, že pokud by došlo

k tomu, že ze dvou blízkých bodů je každý v jiném kraji, je takových situací minimum.

Tabulka 10: Výsledky kontroly blízkých definičních bodů na testovací sadě krajských měst

Velikost parametru *Minvzdal*	**2**	**4**	**6**
Velikost parametru *sdo_num_res*	4	6	10
Doba běhu procedury *pK3_BlizkeDFB* [s]	1 144	1 538	1 844
Počet nalezených blízkých dvojic	6 355	77 946	118 848
Počet dvojic, kde oba stavební objekty jsou způsobu využití garáž	823	64 215	92 521
Počet dvojic, kdy alespoň jeden z objektů má identifikační parcelu s *ID*=0	1 995	4 762	6 176
Počty blízkých dvojic bez garáží a objetků s identifikační parcelou 0	3 537	8 969	20 151

9.2 Diskuze výsledků vyhledání stavebních objektů, které mají podezřelá čísla popisná nebo evidenční

Výsledky vyhledání stavebních objektů, které mají podezřelá čísla popisná nebo evidenční, jsou shrnuty v tabulce 11. Jedná se o prvky vyhledané procedurou *pK1_PodezrelaCPCE* pro vybrané hodnoty parametru *MaxRozdil*. Podezřelá přerušení souvislé číselné řady, nalezená procedurou *pK1_PodezrelaPreruseni* pro vybrané hodnoty parametru *MaxPreruseni*, jsou uvedena v tabulce 12. V tabulce 13 jsou uvedeny časy běhu jednotlivých procedur. Procedury *pK1_PodezrelaCPCE* a *pK1_PodezrelaPreruseni* běží na této testovací sadě stejně dlouho pro libovolnou hodnotu vstupního parametru.

Tabulka 11: Počty nalezených podezřelých čísel domovních v testovací sadě krajských měst.

Podezřelý prvek	**Podezřelá čísla popisná**	**Podezřelá čísla evidenční**
***MaxRozdil*=15**	137	929
***MaxRozdil*=30**	102	583
***MaxRozdil*=50**	87	417
***MaxRozdil*=100**	63	257
***MaxRozdil*=150**	51	194
***MaxRozdil*=200**	43	152

Tabulka 12: Počty podezřelých přerušení v řadě čísel v testovací sadě krajských měst.

Přerušení řady	čísel popisných	čísel evidenčních
***MaxPreruseni* =15**	406	1955
***MaxPreruseni* =30**	231	1288
***MaxPreruseni* =50**	170	940
***MaxPreruseni* =100**	120	610
***MaxPreruseni* =150**	99	473
***MaxPreruseni* =200**	90	393

Tabulka 13: Doba běhu procedur, použitých pro vyhledání podezřelých čísel domovních

Procedura	Doba běhu procedury [s]
pK1_RozlozeniCD	42
pK1_PodezrelaCPCE	27
pK1_PodezrelaPreruseni	26

9.3 Diskuze výsledků vyhledání stavebních objektů, které mají vazbu na stejnou parcelu KN

Touto kontrolou bylo v testovací sadě krajských měst vyhledáno 14 743 stavebních objektů, které mají společně vazbu na 4 856 parcel. 4 814 z těchto stavebních objektů má vazbu na parcelu, jejíž *ID* je 0. Pokud tyto hodnoty nebudeme uvažovat, jedná se o tedy o 9 929 nalezených stavebních objektů. Doba trvání procedury *pK3_SoNa1Parcele* je 82 vteřin.

9.4 Diskuze výsledků vyhledání stavebních objektů, které mají definiční bod stavebního objektu a parcely příliš vzdálený

Doba běhu procedury *pK2_SoMimoParcelu* nad daty všech krajských měst je 25583 s. Procedura vybere 4509 stavebních objektů, jejichž definiční body jsou příliš vzdáleny od definičního bodu parcely. Počty objektů, které byly zpracovány jednotlivými způsoby popsanými v kapitole 8.1.4, jsou shrnuty v tabulce 14.

Tabulka 14: Počty podezřelých objektů vyhledaných procedurou *pK2_SoMimoParcelu*.

Způsob	Popis způsobu	Počet vyhledaných objektů
Způsob 1	Průnik polygonu parcely a definičního bodu stavebního objektu	2959
Způsob 2	Výpočet mezní vzdálenosti z výměry parcely	59
Způsob 3	Průnik polygonu stavebního objektu a definičního bodu parcely	67
Způsob 4	Objekt rovnou označen za podezřelý (objekt nemá polygon a má na něj vazbu více adresních míst)	1424

Doba běhu procedury je 25583 s. To znamená asi 426 minut, tedy něco přes 7 hodin. Vysoká časová náročnost je způsobena především řešením způsobu 4, kde je potřeba pro každý stavební objekt zjišťovat, zda má evidovaný polygon. Vzhledem k tomu, že způsob 4 vyhledá objekty, o kterých lze v podstatě říct jen to, že u nich nelze na základě kontroly rozhodnout, zda jsou chybné nebo ne, je pro urychlení běhu procedury vhodné způsob 4 nepoužít. To lze vyřešit jednoduše pomocí vložení znaku komentáře (--) před příslušné části skriptu. Bez řešení způsobu 4 je pak časová náročnost procedury 966,6 s. Časovou náročnost je opět možné snížit tím, že kontrola bude na data aplikována v rámci menších územních celků.

10 Závěr

Zadání práce vzniklo ve spolupráci s ČÚZK, který se aktivně snaží zlepšovat kvalitu dat základního registru RÚIAN. Cílem práce bylo navrhnout a otestovat řešení čtyř zadaných kontrol stavebních objektů, evidovaných v registru.

Pro řešení kontrol byl nad testovacími daty vytvořen datový model, založený na datovém modelu RÚIAN. Další metodika tvorby kontrol je následující: v případě, že existují algoritmy pro řešení kontroly, byla provedena jejich rešerše a vybrané algoritmy byly implementovány a otestovány. V případě, kdy nebyly použity existující algoritmy, byly pro řešení navrženy nové postupy, které byly opět implementovány a otestovány na testovacích datech. Na základě výsledků porovnání byl vybrán nejvhodnější a nejefektivnější postup pro realizaci kontroly v produkčním prostředí RÚIAN. Jednotlivé kontroly jsou implementovány formou procedur v jazyce PL/SQL. Části kódů těchto kontrol jsou uvedeny v textu práce, celé kontroly jsou pak v příloze nebo na přiloženém CD.

Kontroly vyhledají objekty, které jsou podezřelé z chyb. V mnoha případech nelze totiž pouze na základě výsledků kontroly jednoznačně říci, zda se jedná o chybu či nikoli a je nutné další zkoumání stavebního objektu, například analýza jeho atributů, kontrola katastrální mapy, přezkoumání podkladů na základě kterých došlo k zápisu, případně časově nejnáročnější kontrola v terénu.

Kontroly lze aplikovat na data RÚIAN a vytvořit tak seznam objektů, které jsou podezřelé z chyb. Dále je možné vytvořit rozdílové sestavy, na jejichž základě lze po editorech RÚIAN, v případě stavebních objektů se jedná o obce a stavební úřady, požadovat opravu v RÚIAN. Ne vždy však editoři k opravám dat v RÚIAN přistupují stejně aktivně jako ČÚZK a stává se, že k odstranění některých vyhledaných chyb v dohledné době nedojde.

Použitá literatura

[1] Správa základních registrů. [online]. [cit. 2015-03-23]. Dostupné z: http://www.szrcr.cz/

[2] Předpis č. 111/2009 Sb., Zákon o základních registrech, ve znění pozdějších předpisů.

[3] HEJDOVÁ, Jana. *Kontrola vybraných územních prvků a evidenčních jednotek v RÚIAN*. Plzeň, 2014. Diplomová práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd.

[4] Český statistický úřad: Registr osob. [online]. [cit. 2015-02-15]. Dostupné z: https://www.czso.cz/csu/czso/registr_osob

[5] ČÚZK. Aplikace VDP. [online]. [cit. 2015-02-15]. Dostupné z: http://vdp.cuzk.cz/

[6] Vyhláška č. 359/2011 Sb., o základním registru územní identifikace, adres a nemovitostí.

[7] Časopis Veřejná správa: *Mimořádná příloha čísla 20/2011, ročník XXII* [online]. Ministerstvo vnitra České republiky, 2011. [cit. 2015-02-25]. Dostupné z: https://www.uoou.cz/VismoOnline_ActionScripts/File.ashx?id_org=200144&id_dokumenty=3053

[8] Hospodářské noviny. Základní registr územní identifikace, adres a nemovitostí (RÚIAN) v cílové rovince. [online]. 19. 6. 2013. [cit. 2015-04-25]. Dostupné z: http://komercniprezentace.ihned.cz/c1-60099770-zakladni-registr-uzemni-identifikace-adres-a-nemovitosti-ruian-v-.cilove-rovince

[9] Předpis č. 256/2013 Sb. Zákon o katastru nemovitostí (katastrální zákon).

[10] Předpis č. 357/2013 Sb. Vyhláška o katastru nemovitostí (katastrální vyhláška).

[11] Předpis č. 183/2006 Sb. Zákon o územním plánování a stavebním řádu (stavební zákon).

[12] BUREŠOVÁ, Kateřina. *Formuláře pro reklamaci údajů vedených v RÚIAN*. In: GIS Ostrava 2015: Současné výzvy geoinformatiky.
Dostupné z:
http://gis.vsb.cz/gisostrava/cz/abstractview.php?id=gis2015541c0186940e2

[13] JANEČKA, Karel. *Modelování konzistentní báze geodat na úrovni datového modelu katastru nemovitost*. Plzeň, 2009. Disertační práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd.

[14] BARTOŠ, Jiří. *Geodata and metadata of ISCRE in INSPIRE environment*. Praha, 2011. Disertační práce. ČVUT, Fakulta stavební, 2011.

[15] KAROCHOVÁ, Simona. *Analýza uliční sítě v obcích České republiky*. Praha, 2014. Disertační práce. ČVUT, Fakulta stavební.

[16] RUIAN / GDAL. Portál Free GIS. [online]. [cit. 2015-03-25]. Dostupné z: http://freegis.fsv.cvut.cz/gwiki/RUIAN_/_GDAL

[17] CHÁVEZ, Edgar; FIGUEROA, Karina; NAVARRO, Gonzalo. Proximity searching in high dimensional spaces with a proximity preserving order. In:MICAI 2005: Advances in Artificial Intelligence. Springer Berlin Heidelberg, 2005. p. 405-414.

[18] LV, Qin, et al. Multi-probe LSH: efficient indexing for high-dimensional similarity search. In: Proceedings of the 33rd international conference on Very large data bases. VLDB Endowment, 2007. p. 950-961.

[19] KAMOUSI, Pegah; CHAN, Timothy M.; SURI, Subhash. Closest pair and the post office problem for stochastic points. Computational Geometry, 2014, 47.2: 214-223.

[20] BRIN, Sergey. Near Neighbor Search in Large Metric Spaces. In: 21th International Conference on Very Large Data Bases (VLDB 1995). Zurich, 1995.

[21] ARYA, Sunil, et al. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. Journal of the ACM (JACM), 1998, 45.6: 891-923.

[22] BŘEZINA, Jan. Aplikovaná matematika: Poznámky ke cvičením včetně příkladů. [online]. [cit. 2015-02-12].
Dostupné z: http://atrey.karlin.mff.cuni.cz/~morf/vyuka/ama/jb-ama.pdf

[23] ZHANG, Jun, et al. All-nearest-neighbors queries in spatial databases. In:Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on. IEEE, 2004. p. 297-306.

[24] DU, Shaoyi, et al. Affine iterative closest point algorithm for point set registration. *Pattern Recognition Letters*, 2010, 31.9: 791-799.

[25] CHAUDHRY, Surbhi; XU, Xiabing; AMATO, Nancy. Nearest neighbor search methods. Technical Report, 2008.

[26] SAMETH, Hanan. *Foundations of multidimensional and metric data structures*. San Francisco: Morgan Kaufmann, 2006. ISBN 0123694469.

[27] MOORE, Andrew W. *An intoductory tutorial on kd-trees*. Carnegie Mellon University, 1991.

[28] ROUSSOPOULOS, Nick; KELLEY, Stephen; VINCENT, Frédéric. Nearest neighbor queries. In: ACM sigmod record. ACM, 1995. p. 71-79.

[29] BEYGELZIMER, Alina; KAKADE, Sham; LANGFORD, John. Cover trees for nearest neighbor. In: Proceedings of the 23rd international conference on Machine learning. ACM, 2006. p. 97-104.

[30] ŽALIK, Borut. An efficient sweep-line Delaunay triangulation algorithm.Computer-Aided Design, 2005, 37.10: 1027-1038..

[31] BENTLEY, John Louis, Bruce W. WEIDE a Andrew C. YAO. *Optimal Expected-Time Algorithms for Closest Point Problems*. ACM Transactions on Mathematical Software. New York: Association for Computing Machinery, 1980.

[32] ORACLE CORPORATION. Oracle Spatial User's Guide and Reference [online]. [cit. 2015-02-23]. Dostupné z: https://docs.oracle.com/en/

[33] ČÚZK. Číselníky ISÚI. [online]. [cit. 2015-03-23]. Dostupné z: http://www.cuzk.cz/Uvod/Produkty-a-sluzby/RUIAN/2-Poskytovani-udaju-RUIAN-ISUI-VDP/Ciselniky-ISUI/Ciselniky-ISUI.aspx

[34] Předpis č. 128/2000 Sb., Zákon o obcích (obecní zřízení).

[35] ČÚZK. Řešení nestandardních situací při zadání parcely a definičního bodu. [online]. [cit. 2015-03-14]. Dostupné z: http://www.cuzk.cz/Uvod/Produkty-a-sluzby/RUIAN/1-Editacni-agendovy-system-ISUI/Uzivatelske-postupy-v-ISUI/Zadani_parcely_a_definicniho_bodu.aspx

[36] Statistika: E-learningový systém Moodle. [online]. Fakulta životního prostředí J. E. Purkyně v Ústí nad Labem [cit. 2015-03-20]. Dostupné z: http://moodle.fzp.ujep.cz/course/view.php?id=11

[37] ArcGIS for desktop: ESRI [onffline]. [cit. 2015-04-29]. Dostupné z: http://www.esri.com/software/arcgis/arcgis-for-desktop

[38] What is the Jenks optimization method?. ESRI Support. [online]. [cit. 2015-03-23]. Dostupné z: http://support.esri.com/en/knowledgebase/techarticles/detail/26442

[39] Words From the Chair. The Association of American Geographers, Cartography Specialty Group. [online]. California State University, Northridge [cit. 2015-03-20]. Dostupné z: http://www.csun.edu/~hfgeg003/csg/winter97.html

[40] ODM as a spatial index. Orientation and Processing od Airborne Laser Scanning Data. [online]. Vienna University of Technology [cit. 2015-02-12]. Dostupné z: http://geo.tuwien.ac.at/opals/html/ref_odm.html

[41] FORMÁNEK, Jiří. RÚIAN. [online]. [cit. 2015-02-15]. Dostupné z: http://download.arcdata.cz/konf/2012/prezentace/Formanek_RUIAN.pdf

[42] ČÚZK. Struktura a popis výměnného formátu RÚIAN [online]. 22. 04. 2015 [cit. 2015-05-03]. Dostupné z: http://www.cuzk.cz/Uvod/Produkty-a-sluzby/RUIAN/2-Poskytovani-udaju-RUIAN-ISUI-VDP/Vymenny-format-RUIAN/Vymenny-format-RUIAN-(VFR)/Struktura-a-popis-VFR-1_6_0.aspx

[43] VOGIATZIS, Michael. How to spot first stories on Twitter using Storm. [online]. [cit. 2015-05-03]. Dostupné z: http://micvog.com/2013/09/08/storm-first-story-detection/

Přílohy

I. Struktura přiloženého CD

II. Kódy výsledných kontrol

Vyhledání stavebních objektů, které mají blízké definiční body

Tabulka pro uložení vyhledaných hodnot:

```
CREATE TABLE K3_BlizkeDFB
-- Tabulka pro ulozeni stavebnich objektu a vzdalenosti jejich definicnich bodu
-- Vyhledání stavebních objektů, u kterých jsou totožné definiční body
  (SO1 NUMBER, --kod prvniho stavebniho objektu
   SO2 NUMBER, -- kod druheho stavebniho objektu
   vzdalenost NUMBER,   -- vzdalenost dvou blizkych stavebnich objektu
   SO1_CD VARCHAR(254), --cislo domovni SO1
   SO2_CD VARCHAR(254), --cislo domovni SO2
   SO1_CO NUMBER(9,0),  --cast obce SO1
   SO2_CO NUMBER(9,0),  --cast obce SO2
   SO1_ZV NUMBER(4,0),  --zpusob vyuziti SO1
   SO2_ZV NUMBER(4,0),  --zpusob vyuziti SO2
   SO1_KU NUMBER(9,0),  --katastralni uzemi, na ktere ma vazbu identifikacni parcela SO1
   SO2_KU NUMBER(9,0),  --katastralni uzemi, na ktere ma vazbu identifikacni parcela SO2
   SO1_PAM NUMBER (2),  --pocet adresnich mist, ktera maji vazbu na SO1
   SO2_PAM NUMBER (2)   --pocet adresnich mist, ktera maji vazbu na SO2
   );
```

Procedura *pK3_BlizkeDFB* pro vyhledání blízkých definičních bodů:

```
CREATE OR REPLACE PROCEDURE pK3_BlizkeDFB(minvzdal INTEGER) AS
--procedura pro vyhledani blizkych definicnich bodu stavebnich objektu
--pouziti funkce SDO_NN

BEGIN
-- vymazani zaznamu z pomocne tabulky pro ukladani dvojic blizkych stavebnich objektu
DELETE K3_BlizkeDFB;
INSERT INTO K3_BlizkeDFB(SO1, SO2, VZDALENOST)
--vyhledani dvojic blizkych definicnich bodu a jejich vzdalenosti
SELECT  distinct
   ISO.kod AS SO1, SO.kod AS SO2, MDSYS.SDO_NN_DISTANCE(1) AS VZDAL
   FROM StavebniObjektDFB SO, StavebniObjektDFB_INDEX ISO
   WHERE SDO_NN(ISO.geom,  SO.geom,
      'sdo_num_res=5',1)='TRUE'
      AND ISO.kod<>SO.kod
      AND MDSYS.SDO_NN_DISTANCE(1)<=minvzdal;
-- vamazani stejnych zaznamu s prohozenym SO1 a SO2
DELETE FROM K3_BlizkeDFB
    WHERE SO1 IN (Select SO2 FROM K3_BlizkeDFB)
    AND SO2 IN (Select SO1 FROM K3_BlizkeDFB)
    AND SO1>SO2;
-- spusteni procedury pro doplneni atributu k nalezenym blizkym definicnim bodum
  pK3_DoplneniAtributu;
END;
/
```

Procedura *pK3_DoplneniAtributu*:

```
CREATE OR REPLACE PROCEDURE pK3_DoplneniAtributu AS
-- Procedura pro nalezeni bodu do vzdalenosti minvzdal pomoci R-tree indexu
-- Pouziti funkce SDO_NN
CURSOR c_Kody IS SELECT K3.SO1, K3.SO2
  FROM K3_BlizkeDFB K3;
TYPE t_Kody IS RECORD (
-- typ pro definici promenne pro ukladani hodnot kurzoru c_Kody
```

```
  SO1   NUMBER,
  SO2   NUMBER
);
tmp_Kody t_Kody;
TYPE t_Atrib IS RECORD(
-- typ pro ulozeni vybranych atributu stavebnich objektu
  SO_CD VARCHAR2(254 BYTE),
  SO_CO NUMBER(9,0),
  SO_ZV NUMBER(4,0));
tmp_Atrib1 t_Atrib; -- promenna pro vybrani nekterych atributu SO1
tmp_Atrib2 t_Atrib; -- promenna pro vybrani nekterych atributu SO2
KU1 NUMBER(9,0); -- katastralni uzemi SO1
KU2 NUMBER(9,0); -- katastralni uzemi SO2
PAM1 NUMBER(2,0); -- pocet adresnich mist, ktere maji vazbu na SO1
PAM2 NUMBER(2,0); -- pocet adresnich mist, ktere maji vazbu na SO2

BEGIN
OPEN c_Kody;
LOOP
  FETCH c_Kody INTO tmp_Kody;
  EXIT WHEN c_Kody%NOTFOUND;
-- vyber cisla domovniho, casti obce a zpusobu vyuziti objektu SO1
  SELECT CislaDomovni, CastObce, ZpusobVyuz INTO tmp_Atrib1
    FROM StavebniObjektDFB
    WHERE StavebniObjektDFB.kod=tmp_Kody.SO1;
-- vyber katastralniho uzemi objektu SO1
  SELECT KatastralniUzemi INTO KU1
    FROM StavebniObjektDFB, ParcelaDFB
    WHERE StavebniObjektDFB.kod=tmp_Kody.SO1
      AND StavebniObjektDFB.IdentifikacniParcela=ParcelaDFB.Id;
-- vyber poctu abresnich mist, ktere maji vazbu na stavebni objekt SO1
  SELECT COUNT(kod) INTO PAM1
    FROM AdresniMisto
    WHERE stavebniob=tmp_Kody.SO1;

-- vyber cisla domovniho, casti obce a zpusobu vyuziti objektu SO2
  SELECT CislaDomovni, CastObce, ZpusobVyuz INTO tmp_Atrib2
    FROM StavebniObjektDFB
    WHERE StavebniObjektDFB.kod=tmp_Kody.SO2;
  SELECT KatastralniUzemi INTO KU2
    FROM StavebniObjektDFB, ParcelaDFB
    WHERE StavebniObjektDFB.kod=tmp_Kody.SO2
      AND StavebniObjektDFB.IdentifikacniParcela=ParcelaDFB.Id;

-- vyber poctu abresnich mist, ktere maji vazbu na stavebni objekt SO2
  SELECT COUNT(kod) INTO PAM2
    FROM AdresniMisto
    WHERE stavebniob=tmp_Kody.SO2;
-- aktualizace tabulky K3_BlizkeDFB
-- vlozeni atributu k prislusnym stavebnim objektum
  UPDATE K3_BlizkeDFB SET
        SO1_CD=tmp_Atrib1.SO_CD, SO2_CD=tmp_Atrib2.SO_CD,
        SO1_CO=tmp_Atrib1.SO_CO, SO2_CO=tmp_Atrib2.SO_CO,
        SO1_ZV=tmp_Atrib1.SO_ZV, SO2_ZV=tmp_Atrib2.SO_ZV,
        SO1_KU=KU1, SO2_KU=KU2,
        SO1_PAM=PAM1, SO2_PAM=PAM2
      WHERE SO1=tmp_Kody.SO1 AND SO2=tmp_Kody.SO2;
END LOOP;
CLOSE c_Kody;
END;
/
```

Vyhledání podezřelých čísel popisných/evidenčních

Rozložení řetězce domovních čísel na jednotlivá čísla:

Tabulka pro uložení rozložených záznamů:

```
CREATE TABLE K1_CislaDomovni (
--Pomocna tabulka pro ulozeni kodu, cisla popisneho a casti obce z tabulky
       --stavebniobjekt
  kod NUMBER(9, 0),
  CD NUMBER (10,0),
  typSO NUMBER(4,0),
  CastObce NUMBER(9));
```

Procedura *pK1_RozlozeniCD*:

```
CREATE OR REPLACE PROCEDURE pK1_RozlozeniCD AS

--Procedura pro rozlozeni sloupce CislaDomovni z tabulky StavebniObjet a vyber
       --stavebnich objektu s cislem popisnym
--Rozlozeni stavebnich objektu s vice cisly popisnymi (v DB ve forme 111,112,113,114) na
       --zaznamy, kde kazdy bude obsahovat
    --pouze 1 cislo popisne
--Procedura ulozi separovana cisla popisna, kod SO a cast obce do tabulky

  CURSOR cSt_Obj IS  --kurzor pro prochazeni tabulky StavebniObjekt, vybira pouze
       --stavebni objekty s cislem popisnym
    SELECT kod, CislaDomovni, typSO, CastObce
    FROM StavebniObjektDFB
    WHERE typSO=1 or typSO=2;
  TYPE t_StObj IS RECORD ( --definice typu pro ulozeni kodu, cisla domovniho a casti
       -- obce z tabulky StavebniObjekty
    kod NUMBER(9,0),
    CislaDomovni VARCHAR2(254),
    typSO NUMBER(4,0),
    CastObce NUMBER(9,0));
  tmp_st t_StObj; --promenna pro ukladani hodnot kurzoru cSt_Obj
  carka NUMBER;  --promenna pro ukladani pozice carky v retezci
  retezec VARCHAR(254);       --promenna pro ulozeni hodnot ze sloupce CislaDomovni z
       -- tabulky StavebniObjekty
  AktCislo VARCHAR(10);       --aktualne nalezene cislo popisne v retezci

BEGIN
DELETE K1_CislaDomovni;
  carka:=0;
  OPEN cSt_Obj;
  LOOP
    FETCH cSt_Obj INTO tmp_st;
    EXIT WHEN cSt_Obj%NOTFOUND;
    IF (INSTR(tmp_st.CislaDomovni,',')=0) THEN      --pokud se v CislaDomovni nenachazi
       --znak ',' je radek kurzoru ihned ulozen do
      --pomocne tabulky
      INSERT INTO K1_CislaDomovni (kod, CD, typSO, CastObce)       --vlozeni radku
       -- kurzoru do pomocne tabulky K1_CislaPopisna
        VALUES (tmp_st.kod, TO_NUMBER(tmp_st.CislaDomovni,'99999999'), tmp_st.typSO,
tmp_st.CastObce);
    ELSE       --v CislaDomovni nachazi znak ',' CislaDomovni je potreba rozlozit na
       --jednotliva cisla popisna
      retezec:=tmp_st.CislaDomovni;
      LOOP
        carka:=INSTR(retezec,',');
        IF carka=0 THEN       --nastava kdyz v retezci zbyde posledni cislo popisne
          INSERT INTO K1_CislaDomovni (kod, CD, typSO, CastObce)  --vlozeni do pomocne
        --tabulky K1_CislaPopisna
            VALUES (tmp_st.kod, TO_NUMBER(retezec,'99999999'), tmp_st.typSO,
tmp_st.CastObce);
```

```
          EXIT;
        ELSE
          AktCislo:=SUBSTR(retezec,1,carka-1);      --AktCislo(aktualni cislo popisne)
       --vytvoreno jako substring od zacatku retezce
            --do posledni pozice pred vyskytem carky
          INSERT INTO K1_CislaDomovni (kod, CD, typSO, CastObce)  --vlozeni do pomocne
       -- tabulky K1_CislaPopisna
            VALUES (tmp_st.kod, TO_NUMBER(AktCislo,'99999999'), tmp_st.typSO,
tmp_st.CastObce);
          retezec:=SUBSTR(retezec,carka+1); --oriznuti retezce o jiz nalezene cislo
       --popisne a carku
        END IF;
      END LOOP;
    END IF;
  END LOOP;
  CLOSE cSt_Obj;
END;
/
```

Tabulky pro uložení vyhledaných podezřelých čísel:

```
CREATE TABLE K1_PodezrelaCP (--Tabulka pro ulozeni podezrelych cisel popisnych
  StavebniObjekt NUMBER(9,0),
  Podezrele NUMBER(10),
  CastObce NUMBER(10),
  Predchozi NUMBER(10),
  Nasledujici NUMBER(10));

CREATE TABLE K1_PodezrelaCE (--Tabulka pro ulozeni podezrelych cisel evidencnich
  StavebniObjekt NUMBER(9,0),
  Podezrele NUMBER(10),
  CastObce NUMBER(10),
  Predchozi NUMBER(10),
  Nasledujici NUMBER(10));
```

Procedura *pK1_PodezrelaCPCE* pro vyhledání podezřelých čísel:

```
CREATE OR REPLACE PROCEDURE pK1_PodezrelaCPCE (MaxRozdil IN INTEGER) AS
--Procedura pro nalezeni podezrelych cisel popisnych a cisel evidencnich

  CURSOR c_CO  --kurzor pro prochazeni tabulky CastObce
    IS SELECT kod FROM CastObce;
  tmp_CO NUMBER(9,0); --aktualne kontrolovana cast obce
  TYPE t_c_CD IS REF CURSOR;
  c_CD t_c_CD;  --kurzor pro prochazeni tabulky K1_CislaDomovni
  TYPE t_CD IS RECORD ( --definice typu pro ulozeni kodu, cisla domovniho a casti obce
    kod NUMBER(9,0),
    CD NUMBER(10,0));
  tmp_CD t_CD; --promenna pro ukladani hodnot kurzoru c_CP

  PredchCislo NUMBER(10); --cislo domovni, ktere v ciselne rade predchazi
  AktCislo NUMBER(10); --aktualne zkoumane cislo domovni
  NaslCislo NUMBER(10); --cislo domovni, ktere v ciselne rade nasleduje
  AktKod NUMBER(10,0);
  NaslKod NUMBER(10,0);
  prozdil NUMBER (10); --rozdil aktualniho a predchoziho cisla
  nrozdil NUMBER (10); --rozdil aktualniho a nasledujiciho cisla
BEGIN
-- vymazani dat v tabulkach K1_PodezrelaCP a K1_PodezrelaCE
DELETE K1_PodezrelaCP;
DELETE K1_PodezrelaCE;

--kontrola cisel popisnych
OPEN c_CO;  --postupne prochazeni jednotlivych casti obce
LOOP
  FETCH c_CO INTO tmp_CO;
  EXIT WHEN c_CO%NOTFOUND;
  OPEN c_CD FOR  --otevreni kurzoru pro prochazeni cisel popisnych v tabulce
       --K1_CislaPopisna
    SELECT kod, CD FROM K1_CislaDomovni
```

```
    WHERE CastObce=tmp_CO AND typSO=1
    ORDER BY CD;
  FETCH c_CD INTO tmp_CD;
  AktCislo:=tmp_CD.CD; --aktualni cislo
  AktKod:=tmp_CD.kod; --kod stavebniho objektu s aktualnim cislem
  FETCH c_CD INTO tmp_CD;
  NaslCislo:=tmp_CD.CD; --nasledujici cislo
  NaslKod:=tmp_CD.Kod; --kod stavebniho objektu s nasledujicim cislem
  nrozdil:=NaslCislo-AktCislo;
  --zkoumani nejnizsiho cisla domovniho
  IF (NaslCislo-AktCislo)>MaxRozdil THEN
    INSERT INTO K1_PodezrelaCP (StavebniObjekt, CastObce, Podezrele, Predchozi,
Nasledujici)
      VALUES (AktKod, tmp_CO, AktCislo, NULL, NaslCislo);
  END IF;
  LOOP
    FETCH c_CD INTO tmp_CD;
    EXIT WHEN c_CD%NOTFOUND;
    PredchCislo:=AktCislo;
    AktCislo:=NaslCislo;
    prozdil:=nrozdil;
    AktKod:=NaslKod;
    NaslCislo:=tmp_CD.CD;
    NaslKod:=tmp_CD.kod;
    nrozdil:=NaslCislo-AktCislo;
    IF (nrozdil)>MaxRozdil THEN
      IF (prozdil)>MaxRozdil THEN
        INSERT INTO K1_PodezrelaCP (StavebniObjekt, CastObce, Podezrele, Predchozi,
Nasledujici)
          VALUES (AktKod, tmp_CO, AktCislo, PredchCislo, NaslCislo);
      END IF;
    END IF;
  END LOOP;
  CLOSE c_CD;
  PredchCislo:=AktCislo;
  AktCislo:=NaslCislo;
  AktKod:=NaslKod;
  IF (nrozdil)>MaxRozdil THEN
    INSERT INTO K1_PodezrelaCP (StavebniObjekt, CastObce, Podezrele, Predchozi,
Nasledujici)
      VALUES (AktKod, tmp_CO, AktCislo, PredchCislo, NULL);
  END IF;
END LOOP;
CLOSE c_CO;
dbms_output.put_line('Podezrela cisla popisna byla ulozena do tabulky K1_PodezrelaCP');

  --kontrola cisel evidencnich
  OPEN c_CO;  --postupne prochazeni jednotlivych casti obce
  LOOP
    FETCH c_CO INTO tmp_CO;
    EXIT WHEN c_CO%NOTFOUND;
    OPEN c_CD FOR  --otevreni kurzoru pro prochazeni cisel popisnych v tabulce
K1_CislaPopisna
      SELECT kod, CD FROM K1_CislaDomovni
      WHERE typSO=2 AND CastObce=tmp_CO
      ORDER BY CD;
    FETCH c_CD INTO tmp_CD;
    AktCislo:=tmp_CD.CD; --aktualni cislo
    AktKod:=tmp_CD.kod;
    FETCH c_CD INTO tmp_CD;
    NaslCislo:=tmp_CD.CD; --nasledujici cislo
    NaslKod:=tmp_CD.kod;
    nrozdil:=NaslCislo-AktCislo;
    IF (nrozdil>MaxRozdil) THEN  --porovnani nejnizsiho cisla popisneho, predchozi cislo
        --zde chybi
        INSERT INTO K1_PodezrelaCE (StavebniObjekt, Podezrele, CastObce, Nasledujici)
          VALUES (AktKod, AktCislo, tmp_CO, NaslCislo);
      END IF;
    LOOP
      FETCH c_CD INTO tmp_CD;
      EXIT WHEN c_CD%NOTFOUND;
      PredchCislo:=AktCislo;  --posun cisel
```

```
      AktCislo:=NaslCislo;
      prozdil:=nrozdil;
      AktKod:=NaslKod;
      NaslCislo:=tmp_CD.CD;
      NaslKod:=tmp_CD.kod;
      nrozdil:=NaslCislo-AktCislo;
      IF (nrozdil)>MaxRozdil THEN
        IF (prozdil)>MaxRozdil THEN  --porovnani
          INSERT INTO K1_PodezrelaCE (StavebniObjekt, Podezrele, CastObce, Predchozi,
Nasledujici)
            VALUES (AktKod, AktCislo, tmp_CO, PredchCislo, NaslCislo);
          END IF;
        END IF;
    END LOOP;
    CLOSE c_CD;
    PredchCislo:=AktCislo;
    AktCislo:=NaslCislo;
    IF (nrozdil)>MaxRozdil THEN  --porovnani pro posledni cislo, nasledujici cislo zde
       --chybi
      INSERT INTO K1_PodezrelaCE (StavebniObjekt, Podezrele, CastObce, Predchozi)
        VALUES (NaslKod, AktCislo, tmp_CO, PredchCislo);
    END IF;
  END LOOP;
  CLOSE c_CO;
  dbms_output.put_line('Podezrela cisla evidencni byla ulozena do tabulky
K1_PodezrelaCE');
END;
/
```

Tabulky pro uložení podezřelých přerušení souvislých číselných řad:

```
CREATE TABLE K1_PodezrelaPrerCP (
--tabulka pro ulozeni podezrelych preruseni rady popisnych cisel
  PocatecniSO NUMBER(9, 0),
  PocatecniCD NUMBER(10, 0),
  KonecnySO NUMBER(9, 0),
  KonecneCD NUMBER(10, 0),
  CastObce NUMBER(10, 0)
  );
CREATE TABLE K1_PodezrelaPrerCE (
--tabulka pro ulozeni podezrelych preruseni rady evidencnich cisel
  PocatecniSO NUMBER(9, 0),
  PocatecniCD NUMBER(10, 0),
  KonecnySO NUMBER(9, 0),
  KonecneCD NUMBER(10, 0),
  CastObce NUMBER(10, 0)
  );
```

Procedura *pK1_PodezrelaPreruseni* pro vyhledání podezřelých přerušení souvislých číselných řad:

```
CREATE OR REPLACE PROCEDURE pK1_PodezrelaPreruseni (MaxPreruseni IN INTEGER) AS
--Procedura pro nalezeni podezrelych cisel popisnych a cisel evidencnich

  CURSOR c_CO  --kurzor pro prochazeni tabulky CastObce
    IS SELECT kod FROM CastObce;
  tmp_CO NUMBER(9,0); --aktualne kontrolovana cast obce
  TYPE t_c_CD IS REF CURSOR;
  c_CD t_c_CD;  --kurzor pro prochazeni tabulky K1_CislaDomovni
  TYPE t_CD IS RECORD ( --definice typu pro ulozeni kodu, cisla domovniho a casti obce
    kod NUMBER(9,0),
    CD NUMBER(10,0));
  tmp_CD t_CD; --promenna pro ukladani hodnot kurzoru c_CP
  PredchCislo NUMBER(10); --predchozi cislo
  AktCislo NUMBER(10); --aktualne kontrolovane cislo
  PredchKod NUMBER(10,0); --kod stavebniho objektu s predchozim cislem
  AktKod NUMBER(10,0); --kod stavebniho objektu s aktualnim cislem
```

```
BEGIN
-- vymazani dat v tabulkach K1_PodezrelaCP a K1_PodezrelaCE
DELETE K1_PodezrelaPrerCP;
DELETE K1_PodezrelaPrerCE;

--kontrola cisel popisnych
  OPEN c_CO;  --postupne prochazeni jednotlivych casti obce
  LOOP
    FETCH c_CO INTO tmp_CO;
    EXIT WHEN c_CO%NOTFOUND;
    OPEN c_CD FOR  --otevreni kurzoru pro prochazeni cisel popisnych v tabulce
       --K1_CislaPopisna
      SELECT kod, CD FROM K1_CislaDomovni
      WHERE CastObce=tmp_CO AND typSO=1
      ORDER BY CD;
    FETCH c_CD INTO tmp_CD;
    PredchCislo:=tmp_CD.CD; --aktualni cislo
    PredchKod:=tmp_CD.kod;
    LOOP
      FETCH c_CD INTO tmp_CD;
      EXIT WHEN c_CD%NOTFOUND;
      AktCislo:=tmp_CD.CD; --nasledujici cislo
      AktKod:=tmp_CD.kod;
      IF (AktCislo-PredchCislo)>MaxPreruseni THEN  --porovnani
        INSERT INTO K1_PodezrelaPrerCP (PocatecniSO, PocatecniCD, KonecnySO, KonecneCD,
CastObce)
          VALUES (PredchKod, PredchCislo, AktKod, AktCislo, tmp_CO);
      END IF;
    PredchCislo:=AktCislo;
    PredchKod:=AktKod;
    END LOOP;
    CLOSE c_CD;
  END LOOP;
  CLOSE c_CO;
  dbms_output.put_line('Podezrela preruseni byla ulozena do tabulky
       K1_PodezrelaPrerCP');

  --kontrola cisel evidencnich
  OPEN c_CO;  --postupne prochazeni jednotlivych casti obce
  LOOP
    FETCH c_CO INTO tmp_CO;
    EXIT WHEN c_CO%NOTFOUND;
    OPEN c_CD FOR  --otevreni kurzoru pro prochazeni cisel popisnych v tabulce
       --K1_CislaPopisna
      SELECT kod, CD FROM K1_CislaDomovni
      WHERE CastObce=tmp_CO AND typSO=2
      ORDER BY CD;
    FETCH c_CD INTO tmp_CD;
    PredchCislo:=tmp_CD.CD; --aktualni cislo
    PredchKod:=tmp_CD.kod;
    LOOP
      FETCH c_CD INTO tmp_CD;
      EXIT WHEN c_CD%NOTFOUND;
      AktCislo:=tmp_CD.CD; --nasledujici cislo
      AktKod:=tmp_CD.kod;
      IF (AktCislo-PredchCislo)>MaxPreruseni THEN  --porovnani
        INSERT INTO K1_PodezrelaPrerCE (PocatecniSO, PocatecniCD, KonecnySO, KonecneCD,
CastObce)
          VALUES (PredchKod, PredchCislo, AktKod, AktCislo, tmp_CO);
      END IF;
    PredchCislo:=AktCislo;
    PredchKod:=AktKod;
    END LOOP;
    CLOSE c_CD;
  END LOOP;
  CLOSE c_CO;
  dbms_output.put_line('Podezrela preruseni byla ulozena do tabulky
K1_PodezrelaPrerCE');
END;
/
```

Vyhledání stavebních objektů, které mají vazbu na stejnou parcelu

Tabulka pro uložená vyhledaných objektů:

```
CREATE TABLE K4_SONA1PARCELE (
--Tabulka pro ulozeni kodu staveniho objektu a ID parcely
    kod_SO NUMBER(9,0), --kod stavebniho objektu
    ID_parcely FLOAT); --id parcely, na kterou ma stavebni objekt vazbu
```

Procedura *pK4_SoNa1Parcele* pro vyhledání stavebních objektů, které mají vazbu na stejnou parcelu:

```
CREATE OR REPLACE PROCEDURE pK4_SoNa1Parcele AS
-- procedura pro vyhledani stavebnich objektu,
-- ktere maji vazbu na stejnou parcelu KN
BEGIN
-- vymazani zaznamu z pomocne tabulky
DELETE K4_SoNa1Parcele;
-- join tabulky stavebnich objektu na sebe samu
INSERT INTO K4_SoNa1Parcele -- vlozeni nalezenych do pomocne tabulky
  SELECT DISTINCT SO1.kod, SO1.IdentifikacniParcela
    FROM StavebniObjektDFB SO1
    RIGHT JOIN StavebniObjektDFB SO2
    ON SO1.IdentifikacniParcela=SO2.IdentifikacniParcela
    WHERE SO1.kod<>SO2.kod
    ORDER BY SO1.IdentifikacniParcela; -- serazeni podle parcely
END;
/
```

Vyhledání objektů, které mají definiční bod stavebního objektu a parcely příliš vzdálený:

Tabulka pro uložená vyhledaných objektů:

```
CREATE TABLE K2_SOMimoParcelu (
--tabulka pro ulozeni objektu, vyhledanych procedurou pK2_SOMimoParcelu
  StavebniObjekt NUMBER,
  Parcela NUMBER,
  Vzdalenost NUMBER,
  mez NUMBER (5,1),
  zpusob NUMBER (2,0));
```

Procedura *pK2_SoMimoParcelu* pro vyhledání objektů, které mají definiční bod stavebního objektu a definiční bod parcely příliš vzdálený:

```
CREATE OR REPLACE PROCEDURE pK2_SoMimoParcelu AS
-- procedura pro vyhledavani stavebnich objektu, jejichz definicni bod
--    je prilis vzdaleny od definicniho bodu parcely, na kterou ma SO vazbu
-- vyhledani pripadu, kdy definicni bod SO nelezi uvnitr polygonu parcely
-- kde neni digitalni mapa, je definovana pripustna mez pro maxinalni vzdalenost
-- u stavebnich objektu, na ktere ma vazbu vice adresnich mist,
--    je pouzit prunik obvodu SO a definicniho bodu parcely

-- definice kurzoru, ktery nacte kod a identifikacni parcelu stavebniho objektu
--    a vymeru parcely, na kterou ma stavebni objekt vazbu
CURSOR c_SO IS SELECT SOD.kod, SOD.IdentifikacniParcela, SOD.geom, SO.geom, P.geom
  FROM StavebniObjektDFB SOD
  LEFT JOIN StavebniObjekt SO
```

```
  ON SOD.kod=SO.kod
  LEFT JOIN ParcelaDFB P
  ON SOD.IdentifikacniParcela=P.Id; -- definice kurzoru
TYPE t_SO IS RECORD( -- typ pro definici promenne pro ukladani zaznamu kurzoru c_SO
  kod NUMBER(9,0), -- kod stavebniho objektu
  IdentifikacniParcela FLOAT, -- identifikacni parcela stavebniho objektu
  SOgeom SDO_GEOMETRY, -- geometrie definicnich bodu stavebnich objektu
  SOpolygon SDO_GEOMETRY, --geometrie plygonu stavebniho objektu
  Pgeom SDO_GEOMETRY);
PocAdrMist NUMBER(2,0); -- pocet adresnich mist, ktera maji vazbu na stavebni objekt
tmp_SO t_SO; -- promenna pro ukladani hodnot kurzoru c_SO
mez NUMBER; -- definice meze
vzdal NUMBER; --vypoctena vzdalenost mezi definicnim bodem stavebniho objektu a parcely
tol CONSTANT NUMBER:=0.001;

BEGIN
  DELETE K2_SoMimoParcelu;
    --reseni zpusobu 4
    OPEN c_SO;  --otevreni kurzoru pro prochazeni stavebnich objektu
    LOOP
      FETCH c_SO INTO tmp_SO; --nacitani hodnot z kurzoru
      EXIT WHEN c_SO%NOTFOUND;
      SELECT count(kod) INTO PocAdrMist --vyber poctu adresnich mist
        FROM AdresniMisto
        WHERE stavebniob=tmp_SO.kod;
      --pocet adresnich mist>nez 1 a parcela nema polygon
      IF (PocAdrMist>1) AND (tmp_SO.SOPolygon IS NULL) THEN
          vzdal := SDO_GEOM.SDO_DISTANCE (tmp_SO.SOgeom, tmp_SO.Pgeom, 0.005, NULL);
            INSERT  INTO  K2_SoMimoParcelu  (StavebniObjekt,  Parcela,  Vzdalenost,  Mez,
zpusob)
              VALUES (tmp_SO.kod, tmp_SO.IdentifikacniParcela, vzdal, mez, 4);
      END IF;
    END LOOP;
    CLOSE c_SO;

--zpusob 1, definicni bod stavebniho objektu musi lezet v polygonu parcely
INSERT INTO K2_SoMimoParcelu  (StavebniObjekt, Parcela)
  SELECT SOD.kod, SOD.IdentifikacniParcela
  FROM StavebniObjektDFB SOD, Parcela P
  WHERE P.geom IS NOT NULL AND SOD.IdentifikacniParcela=P.Id
  AND (SELECT COUNT(AM.kod) FROM  AdresniMisto AM WHERE AM.stavebniob=SOD.kod)<=1
  AND SDO_GEOM.RELATE(P.geom, 'mask=COVERS+CONTAINS', SOD.geom, 0.005)='FALSE';
UPDATE K2_SoMimoParcelu  SET zpusob=1
  where zpusob IS NULL;
-- zpusob 2, urceni dopustne vzdalenosti z vymery parcely
INSERT INTO K2_SoMimoParcelu  (StavebniObjekt, Parcela, Vzdalenost, Mez)
  SELECT SOD.kod, SOD.IdentifikacniParcela,
    SDO_GEOM.SDO_DISTANCE (P.geom, SOD.geom, 0.005, NULL),SQRT(P.vymeraparc)
  FROM StavebniObjektDFB SOD, ParcelaDFB P
  WHERE SOD.IdentifikacniParcela=P.Id AND P.Id NOT IN (SELECT Id FROM Parcela)
  AND (SELECT COUNT(AM.kod) FROM  AdresniMisto AM WHERE AM.stavebniob=SOD.kod)<=1
  AND SDO_GEOM.SDO_DISTANCE (P.geom, SOD.geom, 0.005, NULL)>SQRT(P.vymeraparc);
UPDATE K2_SoMimoParcelu SET zpusob=2
  where zpusob IS NULL;
--zpusob 3, definicni bod parcely musi lezet v polygonu stavebniho objektu
INSERT INTO K2_SoMimoParcelu  (StavebniObjekt, Parcela)
  SELECT SO.kod, SO.IdentifikacniParcela
  FROM StavebniObjekt SO, ParcelaDFB P
  WHERE SO.geom IS NOT NULL AND SO.IdentifikacniParcela=P.Id
  AND (SELECT COUNT(AM.kod) FROM  AdresniMisto AM WHERE AM.stavebniob=SO.kod)>1
  AND SDO_GEOM.RELATE(SO.geom, 'mask=COVERS+CONTAINS', P.geom, 0.005)='FALSE';
UPDATE K2_SoMimoParcelu SET zpusob=3
  where zpusob IS NULL;
END;
/
```