Ein Partitionierungsdienst für Geographische Daten in Räumlichen Datenbanken

Hendrik Warneke und Udo W. Lipeck
Institut für Praktische Informatik, FG Datenbanken und Informationssysteme
Leibniz Universität Hannover, Welfengarten 1, 30159 Hannover
{hwa,ul}@dbs.uni-hannover.de

ZUSAMMENFASSUNG

Da in der computergestützten Geographie mit zum Teil sehr großen Mengen von räumlichen Daten gearbeitet werden muss, hat sich mittlerweile die Überzeugung durchgesetzt, solche Datensätze in räumlichen Datenbanken zu speichern. Allerdings wird bei der Entwicklung von geographischen Programmen dem Aspekt der Skalierbarkeit oftmals wenig Beachtung geschenkt. So enthalten verbreitete Programme für Geoinformationssysteme wenig bis keine sogenannten externen Algorithmen, die den Geschwindigkeitsunterschied zwischen internem (RAM) und externem Speicher (Festplatte) berücksichtigen und versuchen, die Anzahl der I/O-Zugriffe auf letzteren zu minimieren. Diese Programme arbeiten daher nur dann effizient, wenn genug interner Speicher für die zu bearbeitenden Geodaten zur Verfügung steht. Wir stellen in diesem Beitrag einen auf Partitionierung basierenden Ansatz vor, der den Aufwand für die Entwicklung von auf großen Geodatenmengen skalierenden Programmen stark reduziert. Dazu werden Zugriffe auf externe Speicher in eine vorgelagerte Partitionierungs- und eine nachgelagerte Rekompositionsphase verschoben und für diese Phasen flexible Operationen, die ein breites Spektrum an geographischen Problemstellungen unterstützen, als Partitionierungsdienst für räumliche Datenbanksysteme angeboten.

1. EINLEITUNG

Lange Zeit verwendete man für die Arbeit mit geographischen Informationen größtenteils auf Papier gedruckte Landkarten. Diese wurden in handliche Kartenblätter unterteilt, um die zur Beschreibung der Erdoberfläche in hoher Auflösung benötigten Datenmengen besser handhaben zu können. Der Übergang zu computergestützten Geoinformationssystemen erlaubte es, Geodaten mit Hilfe unabhängig vom Maßstab in Form von Geometrien zu modellieren. Dabei behielt man häufig die Aufteilung in Kartenblätter auf Dateiebene bei, da die verarbeitbare Datenmenge für viele Systeme und Datenformate beschränkt war. Heute baut man sogenannte Geodateninfrastrukturen auf, in denen Datenproduzenten, Dienstleister und Nutzer über Netzwerke, in der Regel das Internet, miteinander verknüpft sind und geographische Informationen mit standardisierten Verfahren austauschen und verarbeiten [4]. Diese Strukturen sehen vor, dass geographische Daten blattschnittfrei in *räumlichen Datenbanken* gespeichert werden, wozu man häufig objektrelationale Systeme mit Erweiterungen um räumliche Datentypen, Operatoren und Indexe einsetzt.

Aus den grundlegenden, mit den Techniken der Landesvermessung hergestellten Daten, den sogenannten *Geobasisdaten*, können durch vielfältige Prozesse neue Datensätze abgeleitet werden, die man für kartographische Darstellungen von Informationen aus Fachbereichen wie z.B. Verkehr, Umwelt und Ökonomie verwendet. Die folgende Auflistung enthält einige Beispiele für Klassen solcher Prozesse, die heute meistens automatisiert durch Computerprogramme, im folgenden *Geoprogramme* genannt, durchgeführt werden.

Generalisierung: Um eine lesbare Kartendarstellung aus den Geobasisdaten zu erzeugen, müssen diese durch Operationen wie z.B. das Weglassen von Objekten sowie Vereinfachen, Zusammenfassen oder Verdrängen von Geometrien verändert werden. Dabei besitzen die Basisdaten meist eine höhere Auflösung als für die beabsichtigte Darstellung benötigt wird.

Transformation: Erfordert eine Anwendung ein bestimmtes Datenmodell, das sich von dem der Geobasisdaten unterscheidet, müssen diese zunächst in das beabsichtigte Modell transformiert werden.

Integration: Um Informationen aus verschiedenen Datensätzen gemeinsam nutzen zu können, werden diese zu einem Datensatz integriert. Dies erfordert z.B. Verfahren wie die Identifikation korrespondierender geographischer Objekte (Matching) sowie die Anpassung von Geometrien für die gemeinsame Darstellung.

Geoprogramme, die diese Prozesse implementieren, müssen aufgrund des beträchtlichen Umfangs von Geobasisdatensätzen in der Lage sein, den begrenzten Hauptspeicher eines Rechners effizient zu verwalten, um Performanceprobleme aufgrund von Swapping oder Programmabstürze wegen Speichermangels zu vermeiden. Dazu würde es sich anbieten, diese Programme als Datenbankanwendungen zu implementieren, die sämtliche Datenzugriffe über SQL-Befehle abwickeln. Dieses Vorgehen ist jedoch mit einigen Nachteilen verbunden. Zunächst ist man auf die von dem räumlichen Datenbanksystem angebotene Funktionalität für geometrische und geographische Berechnungen beschränkt. Die

24^{th} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 29.05.2012 - 01.06.2012, Lübbenau, Germany.

freie Auswahl aus spezialisierten und optimierten Software-Bibliotheken entfällt. Weiterhin hängt die Performance von SQL-basierten Anwendungen entscheidend von der Qualität der Anfrageoptimierung ab. Aufgrund der inhärenten Schwierigkeit der Kostenschätzung für geometrische Operationen stellt dies insbesondere bei räumlichen Datenbanken ein Problem dar. Schließlich müssten die Algorithmen für Geoprogramme in eine relationale Formulierung umgewandelt werden, was aufgrund von deren Komplexität i.A. aufwändig und fehleranfällig ist. Hinzu kommt noch, dass die Entwickler geographischer Anwendungen oftmals keine Datenbankspezialisten sind.

Wir stellen in diesem Beitrag einen Ansatz vor, der diese Probleme durch *Partitionierung* der Geobasisdaten umgeht. Statt ein Geoprogramm auf den gesamten Datensatz anzuwenden, wird jede Datenpartition für sich allein bearbeitet, wobei diese so klein gewählt wird, dass der verfügbare Hauptspeicher nicht überfüllt wird. Die für die einzelnen Partitionen berechneten Ergebnisdatensätze müssen anschließend wieder zu einem Gesamtdatensatz zusammengesetzt werden, was wir als *Rekomposition* bezeichnen. Da unterschiedliche Prozesse auch unterschiedliche Strategien für die Partitionierung und Rekomposition erfordern, bieten wir eine Sammlung solcher Operationen als Dienst auf einem räumlichen Datenbanksystem an. Die Entwickler von Geoprogrammen können diesen über eine einfache Schnittstelle ansprechen, wodurch die Notwendigkeit entfällt, innerhalb der Programme selbst den Austausch der Daten zwischen Haupt- und sekundärem Speicher zu berücksichtigen.

Für geographische Daten bietet es sich an, zur Partitionierung die räumliche Lage zu verwenden und Datenobjekte aus einem zusammenhängenden Gebiet in eine Partition einzuteilen. Dieser Ansatz ist allgemein für Geoprogramme anwendbar, die sich *lokal* verhalten. Dies bedeutet, dass für die Berechnung von Ergebnissen an einer Position nur Daten aus einer begrenzten Umgebung benötigt werden und diese Umgebung klein gegenüber einer Partition ist. In diesem Fall lassen sich Fehler, die insbesondere am Rand einer Partition entstehen, weil dem Geoprogramm nicht alle Daten zur Verfügung stehen, reduzieren oder ganz vermeiden, indem man sich überlappende Partitionen erzeugt [6].

Der Rest des Beitrags ist wie folgt strukturiert: In Abschnitt 2 nennen wir Arbeiten, die mit unserem Ansatz verwandt sind. Abschnitt 3 beschreibt die Architektur und das Funktionsprinzip des Partitionierungsdienstes. In Abschnitt 4 verwenden wir die Liniensegmentierung als konkretes Anwendungsbeispiel und geben für diesen Prozess alternative Partitionierungs- und Rekompositionsoperationen an. Abschnitt 5 beschreibt die Ergebnisse von Experimenten mit diesen Operationen auf realen geographischen Daten. Schließlich liefert Abschnitt 6 ein Fazit über den in diesem Beitrag vorgestellten Ansatz und einen Ausblick auf Folgearbeiten.

2. VERWANDTE ARBEITEN

Das Prinzip, Datensätze zuerst aufzuteilen, Teilergebnisse zu berechnen und später aus den Teilergebnissen ein Gesamtergebnis zu generieren, ist in der Informatik unter dem Namen Divide-and-Conquer bekannt. Auch für geometrische Probleme gibt es eine Reihe von Vorschlägen, beispielsweise von Güting und Schilling [3], die externe Algorithmen nach diesem Prinzip entwickeln. Um eine optimale asymptotische Laufzeit zu erreichen, werden die drei Teilschritte stark aufeinander abgestimmt, indem z.B. Sortierungen oder spezielle Repräsentationen der Daten weitergegeben werden. Für unseren Partitionierungsdienst sind wir hingegen an flexiblen Partitionierungs- bzw. Rekompositionsoperationen interessiert, die für die Entwicklung der Geoprogramme keine einschränkenden Vorgaben machen.

Aufgrund seiner Wichtigkeit bei der Berechnung räumlicher Verbunde ist das Problem der Bestimmung von sich überschneidenden achsenparallelen Rechtecken besonders intensiv untersucht worden. Während für interne Algorithmen das Plane-Sweep-Paradigma [1] favorisiert wird, verwenden externe oder parallele Algorithmen häufig Partitionierung, wie z.B. Patel und DeWitt [5]. Der Einfluss von Redundanz auf die Laufzeit von partitionierungsbasierten Algorithmen für dieses Problem wird beispielsweise von Zhou et.al. [8] untersucht. In einer ähnlichen Untersuchung [2] kommen Dittrich und Seeger u.a. zu dem auf den ersten Blick überraschenden Ergebnis, dass man durch mehr Redundanz in den Berechnungen sogar die Laufzeit verbessern kann.

3. ENTWURF DES PARTITIONIERUNGSDIENSTES

3.1 Architektur

Wir implementieren Partitionierungs- und Rekompositionsoperationen als Stored Procedures auf einem räumlichen Datenbanksystem. Diese können über eine Datenbankschnittstelle von einem *Steuerprogramm* außerhalb der Datenbank aufgerufen werden, um in der Datenbank gespeicherte Daten für ein Geoprogramm aufzuteilen und wieder zusammenzusetzen (Abbildung 1). Das Steuerprogramm liest jeweils

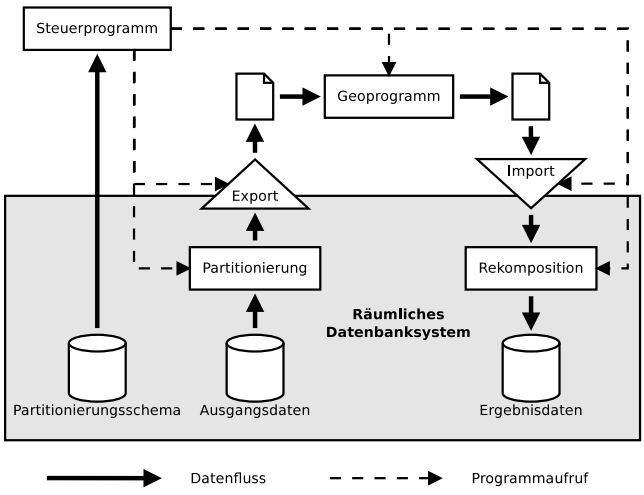

Abb. 1: Geoprogramm (Ablauf) mit Partitionierung

einen Eintrag aus dem Partitionierungsschema (Abschnitt 3.2), das die Geometrien der Partitionen enthält. Damit wird eine geeignete Partitionierungsoperation aufgerufen und eine Partition der Ausgangsdaten berechnet. Diese muss in ein von dem Geoprogramm verwendetes Dateiformat exportiert werden, bevor dieses vom Steuerprogramm aufgerufen wird, um für die Datenpartition ein Ergebnis zu berechnen, das zunächst ebenfalls im Dateisystem abgelegt wird. Nachdem dieses Ergebnis wieder in die Datenbank importiert worden ist, ruft das Steuerprogramm eine geeignete Rekompositionsoperation auf, die die Daten mit den bereits rekompo-

nierten Daten aus anderen Partitionen zusammensetzt und Konflikte auflöst. Anschließend kann das Steuerprogramm mit der nächsten Partition aus dem Partitionierungsschema fortfahren bis der komplette Datensatz bearbeitet ist.

Für Zugriffe auf die möglicherweise großen Datensätze innerhalb der Partitionierungs- und Rekompositionsoperationen verwenden wir konsequent SQL-Anweisungen. Dadurch machen wir implizit von den im räumlichen Datenbanksystem implementierten externen Algorithmen Gebrauch, so dass innerhalb dieser Operationen eine effiziente Abfolge von I/O-Zugriffen erfolgt. Da innerhalb des Geoprogramms nur noch auf die Daten aus einer Partition zugegriffen wird, sind dort keine externen Algorithmen mehr nötig.

3.2 Redundanz

Um vorzugeben, wie geographische Daten bei der Partitionierung aufgeteilt werden, verwenden wir ein sogenanntes *Partitionierungsschema*. Dieses besteht aus einer schnittfreien und lückenlosen Menge von Polygonen (Partitionspolygone), denen jeweils eine eindeutige Partitionsnummer zugeordnet ist. Es ist möglich, ein vordefiniertes Partitionierungsschema (z.B. eine Unterteilung in Verwaltungsbezirke), das in der Datenbank gespeichert ist, zu verwenden, oder den Partitionierungsdienst ein geeignetes Schema (z.B. ein Gitter aus gleich großen Rechtecken) erzeugen zu lassen.

Berechnungen in Geoprogrammen weisen oft eine mehr oder weniger starke Abhängigkeit vom räumlichen Kontext der Datenobjekte auf. Dies bedeutet, dass nur dann korrekte Ergebnisse in einem bestimmten Gebiet erzeugt werden können, wenn auch die Objekte aus einer lokalen Umgebung dieses Gebiets in der Partition enthalten sind. Bei einer streng disjunkten Aufteilung der geographischen Daten, wie durch das Partitionierungsschema vorgegeben, steht insbesondere für Objekte am Rand der Partition möglicherweise nicht genug Kontext zur Verfügung, da Objekte von außerhalb des Partitionspolygons beim Aufruf des Geoprogramms nicht in den Daten enthalten sind. Um diesen Nachteil zu kompensieren, erlauben es die von uns entwickelten Operationen, dass dieselben Datenobjekte in mehreren Partitionen enthalten sind, was wir als *Redundanz* bezeichnen. Das Ziel dabei ist es, dass beim Aufruf des Geoprogramms für eine Partition genug Daten in dieser enthalten sind, um korrekte Ergebnisse zumindest für alle Positionen innerhalb des Partitionspolygons zu berechnen.

Die Erzeugung von Redundanz in den partitionierten Daten lässt sich auf verschiedene Arten erreichen. Eine Möglichkeit ist, die Partitionspolygone um einen festen Abstand zu vergrößern, so dass sich benachbarte Polygone am Rand überlappen. Diese Operation, die auch in Abbildung 2 dargestellt ist, bezeichnet man als Pufferbildung. In Abschnitt 4 geben wir weitere Möglichkeiten an, bei denen redundant zu repräsentierende Daten über die Beziehungen zwischen den Objekten bestimmt werden. Zu beachten ist dabei, dass sich in Abhängigkeit davon, wieviel Redundanz man für eine bestimmte Anwendung benötigt, das Datenvolumen für einzelne Partitionen vergrößert. Im schlimmsten Fall kann eine Partition dann mit dem zur Verfügung stehenden Hauptspeicher nicht mehr bearbeitet werden, wodurch das eigentliche Ziel der Partitionierung verfehlt wird. Häufig kann man für Geoprogramme nachweisen oder experimentell belegen, dass die Abhängigkeit vom Kontext auf Umgebungen mit kleiner räumlicher Ausdehnung beschränkt bleibt (siehe z.B. [6]). In diesen Fällen sprechen wir davon, dass diese Programme

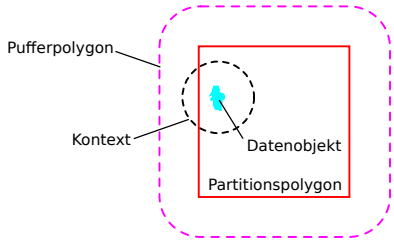

Abb. 2: Kontext, Partitions- und Pufferpolygon

lokale Berechnungen ausführen, und das in diesem Beitrag vorgestellte Konzept ist für solche Anwendungen einsetzbar.

Redundanz führt dazu, dass für einige Datenobjekte mehrere Ergebnisse in unterschiedlichen Partitionen berechnet werden. Wir bezeichnen solche Situationen als *Rekompositionskonflikte*. Bei der Rekomposition müssen diese aufgelöst und die Ergebnisse wieder zu einem Gesamtdatensatz zusammengesetzt werden, der keine Spuren der Partitionierung mehr enthält. Dabei sollen aus den Ergebnisdaten der Berechnung für eine Partition möglichst immer die Objekte übernommen werden, die sich innerhalb des Partitionspolygons befinden, da wir diese Ergebnisse als korrekt ansehen. Die aufgrund von fehlendem Kontext möglicherweise fehlerbehafteten Ergebnisdaten außerhalb des Polygons sind zu verwerfen und aus dem Ergebnis der Partition zu übernehmen, die diese Daten im Inneren enthält. Unterschiedliche Rekompositionsoperationen werden insbesondere benötigt, um verschiedene Konflikte zwischen ausgedehnten Objekten aufzulösen, die die Grenze zwischen benachbarten Partitionen überlappen. Beispiele passender Partitionierungs- und Rekompositionsoperationen für eine geographische Anwendung werden wir in Abschnitt 4 vorstellen.

3.3 Repräsentationsmodelle

Ein wesentliches Ziel beim Entwurf des Partitionierungsdienstes ist Flexibilität. Die von diesem Dienst angebotenen Operationen zur Partitionierung und Rekomposition sollen für eine möglichst große Vielfalt von Geoprogrammen anwendbar sein, um Berechnungen auf großen Datensätzen zu ermöglichen. Geographische Daten können jedoch auf viele verschiedene Arten strukturiert sein. Das Datenbankschema für einen geographischen Datensatz bezeichnen wir im Folgenden als *Repräsentationsmodell* dieser Daten. Um trotz der Heterogenität unterschiedlicher Repräsentationsmodelle nicht für jede Anwendung eigene Operationen anbieten zu müssen, identifizieren wir typische Strukturen bzw. Teilschemata, die in vielen Modellen auftreten. Sind die für eine Anwendung relevanten Informationen in einer solchen Struktur modelliert oder lassen sich in diese transformieren, können wir Operationen zur Partitionierung und Rekomposition einsetzen, die für ein solches Teilschema implementiert sind.

Der zentrale Teil des Schemas für einen geographischen Datensatz bildet häufig die in Abbildung 3 dargestellte Struktur. Die zu beschreibenden Merkmale der Erdoberfläche sind in den Daten durch Objekte dargestellt, die neben der räumlichen Beschreibung durch eine Geometrie noch einen innerhalb des Datensatzes eindeutigen Identifikator und eine Objektart besitzen. Letztere legt fest, um was für einen Typ (z.B. Straße oder Ackerfläche) von Objekt es sich handelt. Zur feineren Charakterisierung der Objekte können weitere Attribute verwendet werden, wobei die erlaubten Typen von

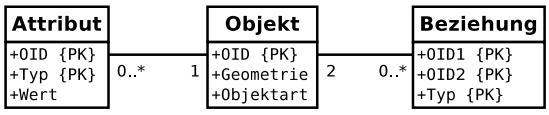

Abb. 3: Objekte, Attribute und Beziehungen

Attributen häufig von der Objektart abhängen (z.B. Breite der Fahrbahn für Straßen). Weiterhin benötigt man Beziehungen zwischen den Objekten, die ebenfalls von unterschiedlichem Typ sein können, um z.B. Über- und Unterführungen an Straßenkreuzungen zu modellieren.

Durch ein Repräsentationsmodell können für die Geometrien der Objekte verschiedene Einschränkungen festgelegt sein, die es beim Partitionieren und Rekomponieren zu berücksichtigen gilt. Eine häufige Einschränkung betrifft z.B. den Geometrietyp, wenn nur Punkte, Linien oder Flächen in einem Datensatz enthalten sind. Weitere gebräuchliche Einschränkungen sind die Forderung der Schnittfreiheit, d.h. Geometrien verschiedener Objekte dürfen sich nur an den Rändern überschneiden, oder das Verbot von Lücken, so dass der komplette Datenraum durch Flächenobjekte überdeckt sein muss. Ein Beispiel für Repräsentationsmodelle mit solchen Einschränkungen sind Landbedeckungsdaten [6]. Diese bestehen aus einer schnittfreien und lückenlosen Menge von Flächenobjekten, denen als Objektart jeweils eine Landbedeckungsklasse (z.B. Nadelwald) zugeordnet ist.

Repräsentationsmodelle mit linien- oder flächenhaften Geometrien, die nicht schnittfrei sind, werden ein wenig abwertend auch als *Spaghetti-Datenmodelle* bezeichnet [4]. Während sie für die Herstellung von Karten gut geeignet sind, bevorzugt man für raumbezogene Analysen sogenannte *topologische Datenmodelle (TDM)*. Die grundlegende Struktur eines topologischen Datenmodells ist in Abbildung 4 dargestellt. Knoten bilden punktförmige Objekte in einem TDM

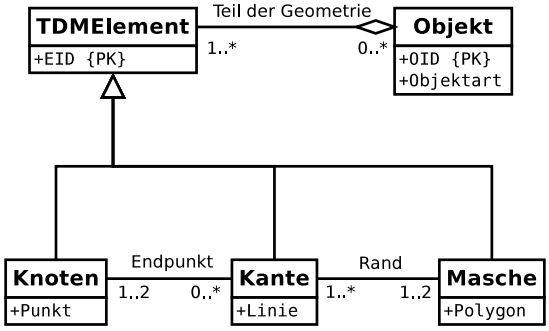

Abb. 4: Datenstruktur eines TDMs

ab und können außerdem die Endpunkte von Kanten darstellen. Kanten wiederum repräsentieren linienhafte Objekte und bilden die Ränder von Maschen. Und solche Maschen entsprechen Teilen von flächenhaften Objekten. Die drei Mengen verschiedenartiger TDM-Elemente sind jeweils schnittfrei, allerdings können ein Knoten oder eine Kante in einer Masche enthalten sein. Ein geographisches Objekt ist durch die TDM-Elemente repräsentiert, durch deren Aggregation man die Geometrie des Objekts rekonstruieren kann. Dabei kann dasselbe Element (z.B. eine Kante) zu mehreren Objekten gehören und repräsentiert in diesem Fall einen gemeinsamen Bestandteil der Geometrien der Objekte, der folglich im Gegensatz zum Spaghetti-Modell nur einmal abgespeichert werden muss. Ein weiterer Unterschied besteht darin, dass im TDM die topologischen Beziehungen der Elemente zueinander explizit gespeichert werden. Dadurch können z.B. benachbarte Flächen bestimmt werden, ohne dass man geometrische Berechnungen durchführen muss.

Weitere typische Schemata werden bei der Integration von Daten verwendet, um Zuordnungen von Objekten aus unterschiedlichen Datensätzen zu modellieren [7]. Wir verzichten aus Platzgründen auf eine detaillierte Beschreibung.

4. ANWENDUNGSBEISPIEL

Als anschauliches Beispiel für die Verwendung unterschiedlicher Partitionierungs- und Rekompositionsoperationen betrachten wir in diesem Abschnitt das geometrische Problem der Liniensegmentierung. Dieses besteht darin, eine Menge von Linienobjekten, die sich beliebig überschneiden dürfen, in eine schnittfreie Menge von Linien zu transformieren. Es gibt eine Reihe von nützlichen Anwendungen, wie z.B. zur Erzeugung der Menge von Kanten bei der Transformation von Spaghetti-Daten in ein topologisches Datenmodell. Die hier verwendete Implementierung besteht aus zwei Schritten. Zuerst werden mit Hilfe eines Plane-Sweep-Verfahrens [1] alle Paare sich schneidender Linien bestimmt. Anschließend wird jede Linie an allen Schnittpunkten unterteilt, so dass diese im Ergebnis durch mehrere Linien dargestellt wird, die sich mit anderen Linien nur noch an den Endpunkten überschneiden. Liegen dabei zwei Linien auf einem längeren Abschnitt übereinander, wird für diesen Abschnitt nur eine einzelne Linie ins Ergebnis übernommen.

Im Folgenden stellen wir für die Liniensegmentierung drei Strategien zur Partitionierung und Rekomposition aus dem in Abschnitt 3 beschriebenen Partitionierungsdienst vor. Es sei angemerkt, dass die für diese Strategien angebotenen Datenbankprozeduren in der Anwendbarkeit nicht auf dieses eine Problem eingeschränkt sind, sondern sich auch für viele weitere Geoprogramme einsetzen lassen. Beispielsweise wurde Variante 1 bereits in [6] erfolgreich für die Generalisierung von Flächendaten angewendet.

4.1 Clipping & Vereinigung

Wir wählen in dieser Variante für eine Partition alle Objekte aus, deren Geometrien sich mit dem Partitionspolygon überschneiden. Zusätzlich schneiden wir bei Objekten am Rand der Partition den Teil der Geometrie ab, der über das Partitionspolygon herausragt. Folglich verbleibt nur der innerhalb des Partitionspolygons liegende Anteil als geometrische Repräsentation des Objekts in der Partition. Dies wird allgemein auch als *Clipping* bezeichnet. Durch diese Art von Aufteilung kann ein Linienobjekt ggf. in mehreren Partitionen auftreten, allerdings jeweils mit einem unterschiedlichen Teil seiner Geometrie, weshalb wir die Partitionierung als disjunkt bezeichnen können. Bei der Liniensegmentierung für eine Partition werden alle Schnitte von Objekten gefunden, die innerhalb des Partitionspolygons liegen, und die Geometrien entsprechend aufgeteilt.

Betrachtet man die Ergebnisse der Liniensegmentierung für die einzelnen Partitionen gemeinsam, fällt auf, dass neben den korrekten Unterteilungen an den Linienschnittpunkten zusätzlich Unterteilungen an den Schnittpunkten mit den Rändern der Partitionen auftreten, was auf das Clipping zurückzuführen ist. Man betrachte z.B. die Situation in Abbildung 5, in der aus drei Linienobjekten a, b und c

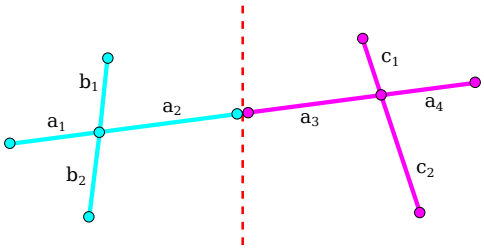

Abb. 5: Überflüssige Segmentierung

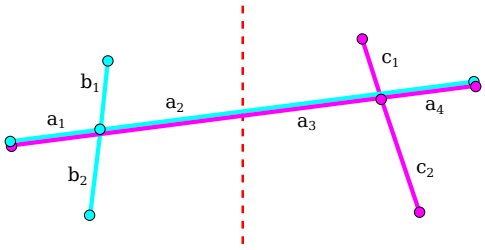

Abb. 6: Überlappende Linien

insgesamt acht segmentierte Linien erzeugt wurden. Bei der Unterteilung zwischen a_2 und a_3 liegt allerdings kein echter Schnitt vor. Da derartige Situationen ohne Partitionierung nicht auftreten würden, liegen hier Rekompositionskonflikte vor und müssen durch eine geeignete Operation bereinigt werden. Dazu bestimmen wir, welche Linien aus dem bereits zusammengesetzten Ergebnis welche anderen Linien aus der aktuellen Partition an der Partitionsgrenze berühren und fügen diese durch eine *Vereinigung* zu einer Linie zusammen.

Wir müssen allerdings auch berücksichtigen, dass ein Linienobjekt aus dem Ausgangsdatensatz mehrmals den Rand der Partition schneiden kann, weshalb wir ggf. auch Gruppen von mehr als zwei Linien zu einem Objekt vereinigen müssen. Ein weiterer Sonderfall liegt vor, wenn sich zwei Linien aus dem Ausgangsdatensatz genau auf dem Rand eines Partitionspolygons überschneiden, weil es sich dann bei dem nach obiger Vorschrift ermittelten Berührungspunkt um einen echten Schnittpunkt handelt und somit keine Vereinigung stattfinden darf. Wir können diese Situationen dadurch erkennen, dass sich zwei segmentierte Linien aus derselben Partition an einem solchen Punkt berühren.

4.2 Partitionsüberschneidung & Durchschnitt

Wie in der ersten Variante wählen wir alle Objekte aus, die das Partitionspolygon schneiden, verzichten aber auf Clipping. Die Intention dabei ist es, aufwändige Berechnungen zum Abschneiden und Vereinigen der Geometrien einzusparen. Dafür nehmen wir etwas Redundanz in Kauf, denn Linien aus dem Ausgangsdatensatz, die über den Rand der Partition hinausragen, werden in mehreren Partitionen mit ihrer vollständigen Geometrie repräsentiert, so dass eine nicht disjunkte Partitionierung vorliegt. Schnitte zwischen solchen Objekten werden demnach bei der Segmentierung mehrfach in unterschiedlichen Partitionen berechnet.

Beim Rekomponieren einer Partition müssen *Duplikate* aus den Ergebnissen entfernt werden, da die aus den mehrfach repräsentierten Objekten gebildeten Linien auch mehrfach in den Ergebnissen auftreten. Allerdings stimmen diese Duplikate in Bezug auf die Segmentierung nicht zwingend überein, denn der Schnitt einer über den Rand der Partition hinausragenden Linie mit einer Linie, die komplett außerhalb der Partition liegt, wird bei der Berechnung in dieser Partition nicht gefunden. Man betrachte z.B. die Situation in Abbildung 6. Während im Ergebnis der linken Partition (cyan) die Linie a am Schnittpunkt mit b in a_1 und a_2 unterteilt wurde, fehlt die Unterteilung am Schnittpunkt mit *c*. Für das Ergebnis der rechten Partition (magenta) hingegen ist die Situation genau umgekehrt.

Um Duplikate zu entfernen, verwerfen wir beim Rekomponieren einer Partition zunächst alle Linien, die komplett außerhalb des Partitionspolygons liegen, denn diese sind entweder bereits im Ergebnis enthalten oder werden beim Rekomponieren einer der nächsten Partitionen eingefügt. Für Situationen wie in Abbildung 6 bestimmen wir für die sich überlappenden Linien den gemeinsamen Teil durch eine geometrische *Durchschnittsoperation*. Z.B. fügen wir anstelle der zu langen Linien a_2 und a_3 nur deren Durchschnitt ins Gesamtergebnis ein.

4.3 Objektüberschneidung & Duplikate

Im Vergleich zur vorigen Variante können wir die Eliminierung der Duplikate bei der Rekomposition weiter vereinfachen, wenn wir bei der Partitionierung noch mehr Redundanz hinzufügen. Wir bezeichnen dazu die Linienobjekte, die das Polygon der Partition p schneiden, als p-Objekte. Um alle p-Objekte bei der Berechnung für diese Partition korrekt zu segmentieren, müssen wir bei der Partitionierung noch Linien von außerhalb der Partition hinzunehmen, die sich mit einem p-Objekt überschneiden.

Dadurch, dass bei der Partitionierung mehr Objekte redundant für mehrere Partitionen ausgewählt werden, entstehen auch mehr Duplikate, die bei der Rekomposition entfernt werden müssen (siehe Abbildung 7). Die meisten dieser Duplikate werden wir wieder los, indem wir (wie bei Variante 2) Linien außerhalb des Partitionspolygons verwerfen (z.B. a_1'/a_4'). Bei allen Linien im Ergebnis einer Partition, die den Rand des Partitionspolygons schneiden, können allerdings in dieser Variante nur echt übereinstimmende Duplikate auftreten, denn diese Linien wurden vollständig segmentiert. Anstatt Durchschnitte zu berechnen, reicht es somit aus, von den am Rand der Partition auftretenden Duplikaten (hier a_2/a_3) jeweils ein Objekt ins Ergebnis zu übernehmen.

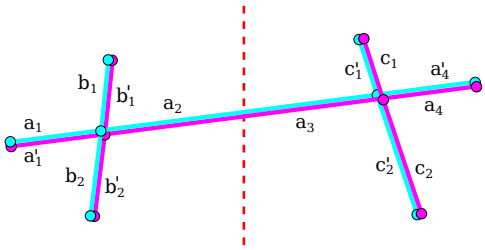

Abb. 7: Doppelte Linien

5. ERGEBNISSE

Um die Anwendbarkeit der in Abschnitt 4 vorgestellten Partitionierungs- und Rekompositionsoperationen zu demonstrieren und die Varianten miteinander zu vergleichen, führen wir Tests mit einem ca. 7,2GB großen kommerziell pro-

duzierten Datensatz durch, der das gesamte Bundesland Hessen umfasst. Diese Daten enthalten Informationen über Straßen und weitere für den Kraftfahrzeugverkehr relevante geographische Objekte in Form von Liniengeometrien, die für die kartographische Darstellung optimiert und insbesondere nicht schnittfrei sind. Partitionierung und Rekomposition sind in einer Datenbank (Oracle 11g) mit räumlicher Erweiterung (Oracle Spatial) implementiert. Zur Segmentierung verwenden wir ein Programm aus einer Java-Bibliothek für geometrische Berechnungen (JTS Topology Suite), das durch zahlreiche Optimierungen auf Kosten eines hohen Speicherverbrauchs sehr effizient arbeitet.

Wir führen für diesen Datensatz mit jeder der drei Varianten eine Segmentierung durch, wobei wir ein Partitionierungsschema aus 129 Quadraten mit jeweils 25km Kantenlänge verwenden. Wir messen und summieren dabei jeweils separat die Laufzeiten, die bei der Partitionierung, Segmentierung und Rekomposition für alle Partitionen benötigt werden. Diese Laufzeiten sind in Abbildung 8 dargestellt. Die Laufzeit für die Segmentierung ist erwartungsgemäß in

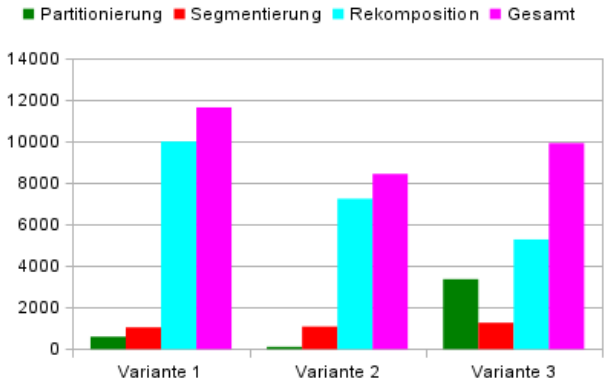

Abb. 8: Laufzeiten (in Sek.) der Prozessphasen

der ersten Variante am geringsten und steigt durch das Hinzufügen von mehr Redundanz in Variante 2 und 3 jeweils leicht an. Am meisten Zeit benötigt in allen Varianten die Rekomposition. Während diese in Variante 3 die geringste Laufzeit benötigt, ist dabei jedoch eine vergleichsweise aufwändige Partitionierung nötig. Die beste Gesamtlaufzeit hat somit Variante 2, bei der die Partitionierung am schnellsten geht und die Zeiten für Segmentierung und Rekomposition jeweils in der Mitte liegen.

6. FAZIT

Der in diesem Beitrag vorgestellte Partitionierungsdienst für geographische Daten in räumlichen Datenbanken besteht im Wesentlichen aus einer Sammlung von flexibel anwendbaren Partitionierungs- und Rekompositionsoperationen. Diese erlauben es, Geoprogramme mit sehr geringem Entwicklungsaufwand fit für die Bearbeitung großer Datenmengen zu machen. Dabei bleibt die Freiheit der Wahl einer Programmiersprache und von Software-Bibliotheken sowie die gute Wartbarkeit der Geoprogramme erhalten, da Zugriffe auf extern gespeicherte Daten nur während der Partitionierung und Rekomposition erfolgen müssen. Neben dem an dieser Stelle vorgestellten Anwendungsbeispiel eignet sich diese Vorgehensweise für eine Vielzahl weiterer geographischer Problemstellungen, sofern diese durch hinreichend lokale Algorithmen gelöst werden können, so dass die benötigte Redundanz bei der Partitionierung nicht zu groß wird.

Während für die in diesem Beitrag vorgestellten Experimente die Größe der Partitionen fest vorgegeben wurde, ist es für einen Anwender des Partitionierungsdienstes wünschenswert, stattdessen die Größe des verfügbaren Speichers angeben zu können, für die der Dienst dann ein geeignetes Partitionierungsschema berechnet. Daher arbeiten wir daran, Modelle aus der Literatur zum Schätzen der Partitionsgröße für räumliche Verbunde zu verallgemeinern, um diese auch auf andere Geoprogramme anwenden zu können. Dabei muss auch berücksichtigt werden, dass reale geographische Daten nicht gleichmäßig verteilt sind, und somit auch die Partitionsgröße innerhalb eines Partitionierungsschemas abhängig von der Datendichte variieren sollte.

Um das Spektrum von möglichen Anwendungen für den Partitionierungsdienst zu vergrößern, muss dieser um weitere Operationen und insbesondere weitere Repräsentationsmodelle erweitert werden. Außerdem werden wir genauer untersuchen, wie sich dieser Dienst möglichst gewinnbringend für die Fortführung von abgeleiteten Datensätzen einsetzen lässt. Dieser Ansatz basiert auf der Idee, bei Updates für Geobasisdaten zunächst möglichst kleine, aber räumlich zusammenhängende Gebiete zu identifizieren, in denen Änderungen stattgefunden haben. Für die Aktualisierung von abgeleiteten Datensätzen müssen dann unter Verwendung von Partitionierung und Rekomposition nur diese Gebiete an Stelle des kompletten Datensatzes neu berechnet werden.

7. DANKSAGUNG

Diese Arbeit wurde vom Bundesamt für Kartographie und Geodäsie im Rahmen des Projekts Wissensbasierter Photogrammetrisch-Kartographischer Arbeitsplatz (WiPKA) gefördert.

8. LITERATUR

[1] Berg, M. de ; Cheong, O. ; Kreveld, M. van ; Overmars, M. : *Computational Geometry: Algorithms and Applications*. 3.Aufl. Springer-Verlag, 2008

[2] Dittrich, J.-P. ; Seeger, B. : Data Redundancy and Duplicate Detection in Spatial Join Processing. In: *Proc. ICDE 2000*, San Diego, S. 535–546

[3] Güting, R. H. ; Schilling, W. : A Practical Divide-and-Conquer Algorithm for the Rectangle Intersection Problem. In: *Information Sciences* 42 (1987), Nr. 2, S. 95–112

[4] Hake, G. ; Grünreich, D. ; Meng, L. : *Kartographie*. 8.Aufl. Walter de Gruyter & Co., 2002

[5] Patel, J. M. ; DeWitt, D. J.: Partition Based Spatial-Merge Join. In: *Proc. SIGMOD 1996*, Montreal, S. 259–270

[6] Thiemann, F. ; Warneke, H. ; Sester, M. ; Lipeck, U. : A Scalable Approach for Generalization of Land Cover Data. In: *Proc. 14th AGILE Intl.Conf. on Geographic Information Systems*. Utrecht, 2011, S. 399–420

[7] Warneke, H. ; Schäfers, M. ; Lipeck, U. ; Bobrich, J. : Matching-Based Map Generalization by Transferring Geometric Representations. In: *Proc. Geoinformatik 2011*, Münster, S. 71–77

[8] Zhou, X. ; Abel, D. J. ; Truffet, D. : Data Partitioning for Parallel Spatial Join Processing. In: *GeoInformatica* 2 (1998), Nr. 2, S. 175–204