Nutzung von Proxys zur Ergänzung von Datenbankfunktionen

Alexander Adam
Technische Universität Chemnitz
Straße der Nationen 62
09107 Chemnitz
alad@cs.tu-chemnitz.de

Sebastian Leuoth
Technische Universität Chemnitz
Straße der Nationen 62
09107 Chemnitz
lese@cs.tu-chemnitz.de

Wolfgang Benn
Technische Universität Chemnitz
Straße der Nationen 62
09107 Chemnitz
benn@cs.tu-chemnitz.de

ZUSAMMENFASSUNG

In dieser Arbeit präsentieren wir eine Möglichkeit, Funktionalitäten – in diesem Fall primär Indizes – in eine Datenbank einzubringen. Dies geschieht in einer Weise, dass weder die Datenbank noch eine Anwendung, welche die Datenbank nutzt, etwas davon bemerken soll. Auf diese Weise soll die aufwendige Anpassung der Anwendungen an eine zusätzliche Komponente vermieden werden. Zugleich wird auch verhindert, dass die Datenbank durch das Einbringen neuer Funktionalitäten instabiler wird.

Keywords

Datenbankerweiterung, Indizierung, Proxy

1. EINLEITUNG

Datenbanken sind allgegenwärtig und nicht mehr aus dem täglichen Leben wegzudenken. Sie verwalten von vertraulichen Personendaten bis hin zum Zahlungsverkehr von Banken sehr viele Bereiche der Gesellschaft. Daraus ergeben sich einige Anforderungen, denen Datenbanksysteme gerecht werden müssen. So kann erwartet werden, dass Zugangsrichtlinien eingehalten und die Daten sicher – d. h. vor fremdem Zugriff und Verlust – verwahrt werden.

Datenbanken, wie wir sie heute kennen, haben sich dabei über viele Jahre hinweg entwickelt. Angefangen von den ersten Systemen, wie z. B. System R [4] und IMS [8], bis zu den heutigen großen relationalen Datenbanksystemen – bspw. Oracle [14] und IBM DB2 [3], um nur einige zu nennen – wurden immer wieder neue Funktionalitäten hinzugefügt. Entwicklungen an Datenbanksystemen finden dabei meist direkt bei den Datenbankherstellern statt. Diese integrieren neue Techniken in ihre Datenbanken, wenn diese ausreichend getestet, ausgereift und für eine breite Öffentlichkeit nutzbar und nützlich sind. Selbst einige Spezialgebiete wurden bereits abgedeckt, so z. B. die Speicherung und schnelle Abfrage von räumlichen Daten mittels Oracle Spacial [17] oder dessen Pendant bei IBM, des DB2 Spatial Extenders [12]. Beide bauen einen neuen Index – einen R-Baum [9] bei Oracle, ein Grid bei DB2 [2] – ein.

All das führt dazu, dass heutige relationale Datenbanken für sehr viele Bereiche Lösungen in Form von Indizes bereits zur Verfügung stellen. Allerdings gibt es auch Anwendungsgebiete – jene, die nicht genügend Nachfrage bieten können – bei denen diese Standardlösungen nicht anwendbar sind. Der Anwender steht dann vor der Wahl, eine komplett eigene Datenbanklösung zu erstellen oder aber eine bestehende Datenbank zu erweitern. Eine Eigenentwicklung kommt dabei für die wenigsten in Frage, da diese immens zeit- und damit kostenintensiv ist. Sie muss darüberhinaus auch selbst weiterentwickelt und mit Aktualisierungen versorgt werden, ein weiterer sehr kostenintensiver Punkt, der gegen eine solche Entwicklung spricht.

Bei der zweiten Möglichkeit – der Erweiterung eines Datenbanksystems – wird über vom Datenbankhersteller bereitgestellte Schnittstellen die Funktionalität einer bestehenden Datenbank um die geforderten Eigenschaften erweitert. Es ist schwierig, diese Erweiterungen dauerhaft in Datenbanken zu integrieren. Zum einen haben die Datenbankhersteller ein berechtigtes Interesse daran, nur gut getestete Komponenten in ihre Produkte einfließen zu lassen, zum anderen müssen diese weitestgehend allgemein einsetzbar sein.

In einigen Umgebungen ist es allerdings nicht möglich, Eingriffe in die Datenbank oder die Anwendungen, die auf sie Zugriff nehmen, vorzunehmen. Ein Szenario ist hier der Betrieb eines Rechenzentrums, welches Datenbankdienstleistungen zur Verfügung stellt. Hier können weder Eingriffe in die Datenbanken vorgenommen werden, noch ist dies bei den Anwendungen möglich, da diese den Kunden gehören.

Aus diesem Grund wollen wir einen Weg schaffen, der sich in eben diesen Situationen einsetzen lässt und der dann auch für andere offensteht. Es soll dabei möglichst erreicht werden, dass das Zusatzmodul, ein Proxy, komplett transparent in die bestehende Infrastruktur integriert werden kann.

Im nächsten Abschnitt werden wir zunächst auf andere Techniken eingehen, die genutzt werden können, neue Funktionalitäten, speziell Indizes, in Datenbanken einzufügen. Anschließend werden wir unser Konzept aufzeigen und einige Spezialfälle diskutieren.

2. STAND DER TECHNIK

2.1 ADT

Klassischerweise können neue Funktionalitäten in Datenbanksysteme mittels Stored Procedures und eigenen abstrakten Datentypen (ADT) implementiert werden. Die meisten Datenbanksysteme bieten diese Möglichkeiten. [11, 15, 18] Diese Erweiterungsmöglichkeiten sind jedoch begrenzt, da mit ihnen immer die Ablage der Daten in den vorgegebenen Strukturen der Datenbank einhergeht. Auch lassen sich so die internen Abläufe nur schwer beeinflussen. Postgres bietet beispielsweise die Möglichkeit die verfügbaren Indizes mittels Callback-Funktionen auf den ADT anwendbar zu machen. Eine komplett neue Indexstruktur lässt sich so allerdings nicht einbauen. Außerdem wird bei einer Anfrage nicht die Selektionsbedingung des `WHERE`-Teiles mit übergeben. Das bedeutet, dass immer der komplette Tabelleninhalt zurückgeliefert werden muss und die Datenbank die Bedingungen des `WHERE`-Teiles schließlich selbst auswertet.

2.2 Plugins

Um nun diese Einschränkungen zu umgehen, wurden tiefergreifende Möglichkeiten geschaffen, auch eigene Indizes in Datenbanksysteme einzubringen. Allgemein wird diese Möglichkeit auch als „Extensible Indexing“ bezeichnet. Oracle setzt dies im *Datacartridge Framework* [5], IBM DB2 mit dem *DB2 Extender* [19] um. Weitere Ausführungen zu u. a. Informix DataBlades [20] und GiST [10] sind auch in [1] sowie auf allgemeinerem Niveau in [13] zu finden.

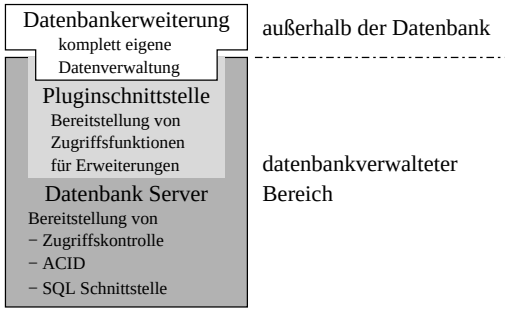

Abbildung 1: Mittels eines Datenbankplugins wie hier dargestellt lassen sich neue Funktionen in eine Datenbank integrieren, insbesondere Indizes (vgl. [5]).

Diese Pluginschnittstellen ermöglichen es, komplett neben der Datenbank Indexfunktionalitäten bereitzustellen und die Datenbank so auf eine reine Verwaltungskomponente zu reduzieren. Die Datenbank macht weiterhin die Zugriffskontrolle, SQL-Auswertung, Transaktionen, gibt dann die eigentliche Ausführung aber an das Plugin ab. Dieses kann z. B. speziell strukturierte Daten sehr effizient außerhalb der Datenbank vorhalten.

Neben konkreten Pluginschnittstellen, steht bei Opensource-Projekten auch der Weg des direkten Eingriffes in den Datenbankkern offen. Da sich interne Strukturen bei solchen Projekten allerdings häufig ändern, ist hier der Wartungaufwand für eine solche Lösung enorm. Des weiteren müssen an dieser Stelle auch alle Funktionen, die angepasst werden, und deren Seiteneffekte beachtet werden.

Beide Verfahren – Plugins und ADTs – haben einen gemeinsamen Schwachpunkt: sie müssen immer in die Datenbank selbst integriert werden. Ein solcher Eingriff, in die u. U. für ein Unternehmen kritischen Datenbestände ist, schon allein aufgrund von Sicherheitsbedenken, schwierig bis unmöglich umzusetzen. Bei einer komplett neuen Datenbank, die erst ihren Regelbetrieb aufnehmen muss, ist dieser Weg jedoch gangbar.

2.3 GreenSQL

Einen anderen Weg geht das GreenSQL-Projekt. Ziel von GreenSQL ist es, eine Datenbank vor schädlichen SQL-Befehlen zu schützen. Dabei nutzt es auch einen Proxyansatz. Die GreenSQL kennt dabei die folgenden Betriebsmodi:

- *Simulationsmodus:* Es findet hierbei kein Eingriff in die Kommunikation statt. Es werden ausschließlich alle Anfragen untersucht und bei verdächtigen Anfragen ein Verantwortlicher informiert.
- *Blockierung von verdächtigen Anfragen:* Hier werden Heuristiken angewandt, um die eingehenden Anfragen zu beurteilen. Wird eine Anfrage als „verdächtig“ eingestuft, so wird noch in einer Positivliste nachgeschaut, ob die Anfrage doch zulässig ist. Ist sie zulässig, so wird sie an den Datenbankserver weitergeleitet, ansonsten wird direkt ein leeres Ergebnis zurückgeliefert.
- *Lernmodus:* Um nicht auf die Heuristiken angewiesen zu sein, kann GreenSQL in diesem Modus eine Liste zugelassener Anfragen erzeugen.
- *Schutz vor unbekannten Anfragen:* Nachdem alle zugelassenen Anfragen „gelernt“ wurden, werden in diesem Modus alle unbekannten Anfragen blockiert.

Die Anfragen werden dabei mittels Mustererkennung untersucht. Eine Anfrage wird also nicht geparst und anhand eines Syntaxbaumes untersucht, was genau bewirkt werden soll. Damit ist der Nutzen auf Anfragen beschränkt, die keine komplexe syntaktische Struktur aufweisen. Des weiteren ist GreenSQL auf MySQL zugeschnitten. Es kennt die Tabellennamen des Systemkataloges, die sich aber bei anderen Datenbanken zu denen von MySQL unterscheiden.

3. UNSER ANSATZ

3.1 Allgemein

Da weder die Datenbank noch die Anwendung angepasst werden dürfen, muss also eine externe Lösung angestrebt werden. In Abbildung 2(a) ist nocheinmal das Ausgangsszenario dargestellt. Im weiteren werden wir uns auf SQL [6] als Kommunikationssprache hin zur Datenbank beschränken. Diese Einschränkung ist insofern unkritisch, da die meisten Datenbanksysteme auf diese Art angesprochen werden und sich der folgende Ansatz auch auf andere Anfragemöglichkeiten übertragen lässt.

Der einzig verbleibende Punkt, um noch etwas zu integrieren, ist die Kopplung zwischen Anwendung und Datenbank,

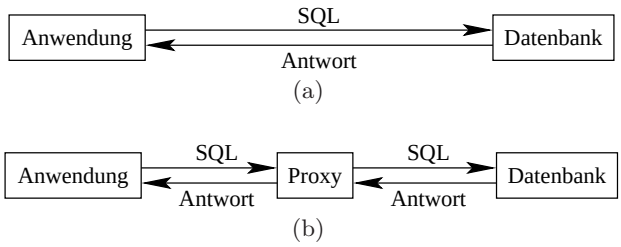

Abbildung 2: Integrationsszenario. In Teil (a) ist dargestellt, wie sich der bisherige Ablauf bei der Kommunikation von Anwendung und Datenbank darstellt. Teil (b) zeigt, wie sich der Proxy einfügt und die Kommunikation auf ihn umgeleitet wird.

sprich die Kommunikationsstrecke. Abbildung 2(b) zeigt, wie ein solcher Eingriff aussehen könnte. Ein Proxy nimmt den Datenverkehr von der Anwendung entgegen und wertet diesen aus. Er miemt also die Datenbank. Zur Beantwortung der Anfrage kann der Proxy trotzdem mit der Datenbank kommunizieren.

Es gibt mehrere Möglichkeiten, einen Proxy in eine bestehende Systemlandschaft zu integrieren:

- Nicht-transparente Integration, indem der Anwendung der Proxy als neuer Datenbankserver mitgeteilt wird. Im Idealfall ist dies nur ein einziger Eintrag in der Anwendung.
- Ersetzen des Datenbankservers durch den Proxy. Hierbei übernimmt der Proxy die Adresse des Datenbankservers und der Datenbankserver bekommt eine neue Adresse zugeteilt.
- Transparentes Einklinken des Proxys. Dabei werden auf Netzwerkebene die Datenpakete, die für den Datenbankserver bestimmt sind, auf den Proxy umgeleitet (Destination Network Address Translation) [7]. Diese Variante ist vom Anpassungsaufwand für die Anwendung als auch für den Datenbankserver am geringsten (im Idealfall Null). Dabei ist jedoch zu beachten, dass es sich hierbei um einen Eingriff in eventuell vertrauliche Kommunikation handelt.

Ziel soll es sein, die letzte Möglichkeit zu realisieren. Bei einer Neuinstallation ist ein neues Indizierungsverfahren, egal ob Datenbank intern oder extern, verhältnismäßig einfach zu integrieren. Die Integration in bestehende Systemlandschaften aber gestaltet sich schwierig. Eine transparente Lösung ist hier die einzig anwendbare.

3.2 Bearbeitung einer Anfrage

Egal für welche Integrationsvariante sich entschieden wurde, im Betrieb sind die Aufgaben der Kernkomponenten des Proxys gleich. Diese Kernkomponenten und ihr Zusammenspiel sind in Abbildung 3 dargestellt. Sendet die Anwendung eine Anfrage an die Datenbank, wird diese zunächst von der Verwaltungskomponente des Proxys entgegengenommen.

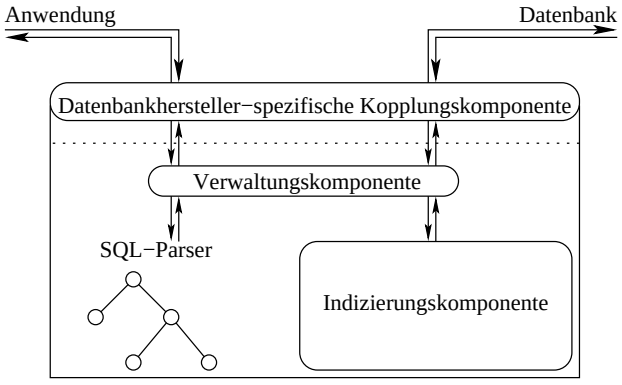

Abbildung 3: Interner Aufbau des Proxys. Der Proxy besteht im Wesentlichen aus einem SQL-Parser, dem eigentlichen Index und einer Verwaltungskomponente, die diese Komponenten miteinander koppelt. Um die Datenbankanfragen einzulesen, wird noch eine Datenbankhersteller-spezifische Schnittstelle benötigt.

Anschließend wird geprüft, ob der Proxy überhaupt etwas mit dieser Anfrage anfangen kann. Im Falle eines Index werden also die Tabellen, auf die sich die Anfrage bezieht, mit den indizierten Tabellen verglichen. Wenn keine indizierte Tabelle getroffen wurde, so wird die Anfrage unbehandelt an die Datenbank weitergereicht, von dieser beantwortet und das Ergebnis an die Anwendung zurückgeliefert.

Wird die Anfrage bearbeitet, so muss der SQL-Parser diese zuerst in Unteranfragen zerlegen. Abbildung 4 zeigt eine solche zerlegte Anfrage, die eine Unteranfrage beinhaltet. Bei der Beantwortung muss der Baum, der sich aus dieser Struktur ergibt, von den Blättern her zur Wurzel hin abgearbeitet werden. Es muss also zunächst die Anfrage „`SELECT mid FROM abt WHERE nr = 3`“ abgearbeitet werden. Die Ergebnisse dieser Teilanfrage werden dann in den darüberliegenden Knoten eingebaut. Sobald alle Kindknoten eines Knotens abgearbeitet wurden, kann der Knoten selbst bearbeitet werden.

```
(0) SELECT * FROM mitarb WHERE mitarb.id IN (①)
(1) SELECT mid FROM abt WHERE abt.nr = 3
```

Abbildung 4: Anfragebaum für eine kleine geschachtelte SQL-Anfrage. Anfrage 1 ist in Anfrage 0 eingebettet, ihre Ergebnisse sind für die Beantwortung von Anfrage 0 entscheidend.

In eine Anfrage, die Daten aus Unteranfragen benötigt, werden die Ergebnisse dieser Unteranfrage zunächst direkt eingesetzt. Im vorigen Beispiel würde also bspw. eine Anfrage wie „`SELECT * FROM mitarb WHERE id IN (17,23,42)`“ erzeugt werden.

Im Falle eines Index ist es nicht sinnvoll, jeden Zweig im Anfragebaum zu verfolgen. Es müssen nur Blätter ausgewertet

werden, die auch tatsächlich an einem für den Index auswertbaren Vaterknoten direkt oder indirekt angeschlossen sind.

Bei der Auswertung eines Anfragebaumes ergibt sich das in Abbildung 5 dargestellte Kommunikationsschema. Es ist zu erkennen, dass während der Beantwortung einer Anfrage u. U. mehrere Anfragen an die Datenbank abgeschickt werden müssen, während die Anwendung auf die Antwort wartet. Dieses Wechselspiel von Proxy und Datenbank kann bei entsprechend tief geschachtelten Anfragen sehr lange dauern. Es muss noch untersucht werden, bis zu welcher Tiefe sich die Ausführung der Auflösung tatsächlich lohnt und ab wann die gesamte Anfrage direkt an die Datenbank weitergeleitet werden sollte.

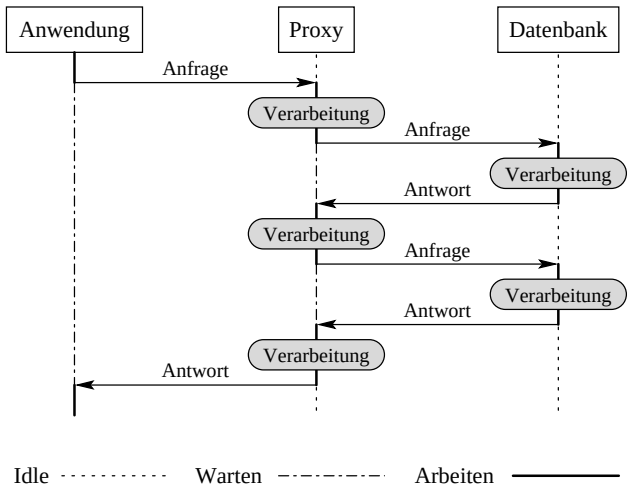

Abbildung 5: Kommunikationsschema. Die Abbildung zeigt das Kommunikationsverhalten der einzelnen Teilnehmer. Zu bemerken ist, dass nicht jede Anfrage der Anwendung auch nur eine Anfrage des Proxys bei der Datenbank bedingt, sondern auch eine Reihe solcher Anfragen an die Datenbank auslösen kann.

Ein Index muss nicht die Daten selbst, sondern nur Identifikatoren speichern, die einzelne Datensätze eindeutig identifizieren. Sein Ergebnis sind also idealerweise Primärschlüssel der eigentlichen Tupel in der Datenbank. Werden einer Datenbank die Tupelidentifikatoren (TIDs) der gesuchten Datensätze mitgeteilt, kann diese sie sehr schnell finden. Es kann allerdings vorkommen, dass die von einem Index zurückgelieferten Ergebnisse unscharf, es also zu viele sind. Deshalb müssen alle `WHERE`-Bedingungen der originalen Anfrage erhalten bleiben.

Eine Möglichkeit der Datenbank, die Ergebnisse eines Index mitzuteilen, ist die Erweiterung der ursprünglichen Anfrage um seine Ergebnisse. Zum Beispiel kann so aus einem einfachen

```
SELECT * FROM mitarb WHERE gehalt > 2000
```

eine um den Primärschlüssel `id` erweiterte Anfrage werden:

```
SELECT * FROM mitarb WHERE gehalt > 2000
AND id IN (4, 18)
```

Problematisch an dieser Lösung kann die begrenzte Länge von SQL-Ausdrücken sein, die eine Datenbank verarbeiten kann. Für den Fall einer Überschreitung dieser Maximallänge muss ein anderer Weg gegangen werden. So können die Ergebnisse, die der Index zurückliefert, in der Datenbank in eine seperate, temporäre Tabelle eingefügt werden. Die neue SQL-Anfrage hätte dann die Form:

```
SELECT * FROM mitarb WHERE gehalt > 2000
AND id IN (SELECT * FROM temp_tabelle)
```

4. AUSBLICK

Das beschriebene Verfahren befindet sich noch in Entwicklung. Von den vorgestellten Komponenten wurde der SQL-Parser umgesetzt. Ein weiterer Punkt, das Verändern von SQL-Anfragen, stellt eine Herausforderung dar, da die Übertragungswege von und zu den Datenbanken herstellerabhängig sind. Erste Tests zeigten Prüfsummen, die sich allerdings nur auf die Länge der Anfrage bezogen. Eine Veränderung wurde erfolgreich vorgenommen. Dieser Punkt ist allerdings ein weites Feld und wir werden uns zunächst auf eine Schnittstelle festlegen, im speziellen auf das Oracle Call Interface (OCI) [16].

Auch muss die Leistungsfähigkeit der beschriebenen Veränderungsverfahren noch untersucht werden. Anfängliche Tests mit manueller Erweiterung der Anfragen zeigten bereits Geschwindigkeitssteigerungen.

Der entwickelte Proxy kann prinzipiell nicht nur einen datenbankexternen Index ermöglichen. Er erlaubt auch eine Fülle anderer Funktionen. So kann er dazu genutzt werden, Antworten auf Anfragen an die Datenbank zwischenzuspeichern, er kann Adapterfunktionen zu anderen SQL-Dialekten oder zu ganz anderen Anfrageparadigmen sein. Diese stehen nicht im direkten Fokus der Arbeit, sollen aber definitiv evaluiert werden.

5. LITERATUR

[1] R. Acker, R. Pieringer, and R. Bayer. Towards Truly Extensible Database Systems. In K. Andersen, J. Debenham, and R. Wagner, editors, *DEXA 2005*, pages 596–605. Springer-Verlag Berlin Heidelberg, 2005.

[2] D. W. Adler. DB2 Spatial Extender - Spatial data within the RDBMS. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, pages 687–690, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[3] R. Ahuja. *DB2 9 Unveiled: Overview and New Enhancements*. IBM Software Group, July 2006.

[4] M. M. Astrahan, M. W. Blasgen, D. D. Chamberlin, K. P. Eswaran, J. N. Gray, P. P. Griffiths, W. F. King, R. A. Lorie, P. R. McJones, J. W. Mehl, G. R. Putzolu, I. L. Traiger, B. W. Wade, and V. Watson. System R: relational approach to database

management. *ACM Trans. Database Syst.*, 1(2):97–137, 1976.

[5] E. Belden, T. Chorma, D. Das, Y. Hu, S. Kotsovolos, G. Lee, R. Leyderman, S. Mavris, V. Moore, M. Morsi, C. Murray, D. Raphaely, H. Slattery, S. Sundara, and A. Yoaz. *Oracle Database Data Cartridge Developers Guide, 11g Release 2 (11.2)*. Oracle, July 2009.

[6] D. D. Chamberlin and R. F. Boyce. SEQUEL: A structured English query language. In *SIGFIDET '74: Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control*, pages 249–264, New York, NY, USA, 1974. ACM.

[7] K. B. Egevang and P. Francis. *RFC 1631 – The IP Network Address Translator (NAT)*. Cray Communications, NTT Software Lab, May 1994.

[8] R. L. Gilliam. *The State of IMS*. IBM Corporation, Department DQZA, Route 100 Box 100, Sommers NY, USA, 06877, Apr. 2005.

[9] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *SIGMOD '84: Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57, New York, NY, USA, 1984. ACM.

[10] J. M. Hellerstein, J. F. Naughton, and A. Pfeffer. Generalized Search Trees for Database Systems. In *VLDB '95: Proceedings of the 21th International Conference on Very Large Data Bases*, pages 562–573, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[11] IBM. *SQL Reference, Volume 1*. IBM Corporation, Nov. 2009.

[12] IBM Deutschland GmbH. *DB2 Spatial Extender und Geodetic Data Management Feature – Benutzer- und Referenzhandbuch*, July 2006.

[13] H.-P. Kriegel, M. Pfeifle, M. Pötke, and T. Seidl. The Paradigm of Relational Indexing: A Survey. In *Proceedings of the 10th Conference on Database Systems for Business, Technology, and the Web (BTW'03)*, GI-Edition Lecture Notes in Informatics, pages 285–304, Leipzig, Deutschland, 2003.

[14] K. Loney. *Oracle Database 11g The Complete Reference*. McGraw-Hill, Inc., New York, NY, USA, 2009.

[15] D. Lorentz and M. B. Roeser. *Oracle Database SQL Language Reference, 11g Release 2 (11.2)*. Oracle, Oct. 2009.

[16] J. Melnick. *Oracle Call Interface Programmer's Guide, 11g Release 2 (11.2)*. Oracle, Oct. 2009.

[17] C. Murray. *Oracle Spatial Developers Guide, 11g Release 2 (11.2)*. Oracle, Dec. 2009.

[18] The PostgresSQL Global Development Group. *PostgreSQL 8.4.3 Documentation*, 2009.

[19] K. Stolze and T. Steinbach. DB2 Index Extensions by example and in detail, IBM Developer works DB2 library. Dec. 2003.

[20] M. Ubell. The Montage extensible DataBlade architecture. *SIGMOD Rec.*, 23(2):482, 1994.