(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 699 10 219 T2** 2004.06.17

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 101 172 B1**
(21) Deutsches Aktenzeichen: **699 10 219.7**
(86) PCT-Aktenzeichen: **PCT/US99/05716**
(96) Europäisches Aktenzeichen: **99 912 573.5**
(87) PCT-Veröffentlichungs-Nr.: **WO 99/048029**
(86) PCT-Anmeldetag: **16.03.1999**
(87) Veröffentlichungstag der PCT-Anmeldung: **23.09.1999**
(97) Erstveröffentlichung durch das EPA: **23.05.2001**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **06.08.2003**
(47) Veröffentlichungstag im Patentblatt: **17.06.2004**

(51) Int Cl.[7]: **G06F 17/30**

(30) Unionspriorität:
39728 **16.03.1998** **US**

(84) Benannte Vertragsstaaten:
AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LI, LU, MC, NL, PT, SE

(73) Patentinhaber:
Microsoft Corp., Redmond, Wash., US

(72) Erfinder:
GRAEFE, Goetz, Bellevue, US; ALGER, Jeff, Redmond, US

(74) Vertreter:
Strehl, Schübel-Hopf & Partner, 80538 München

(54) Bezeichnung: **TRANSFORMATION DER PERSPEKTIVE AUF TABELLEN VON RELATIONALEN DATENBANKEN**

Beschreibung

Hintergrund der Erfindung

[0001] Die vorliegende Erfindung betrifft die elektronische Datenverarbeitung und insbesondere neue Anfrageoperationen für die Manipulation von Tabellen in relationalen Datenbanken.

[0002] Eine Datenbank ist eine Ansammlung von Daten in einer organisierten Struktur. Eine typische Datenbank ist in einem Computer als eine Gruppe von Datensätzen gespeichert, die jeweils eine Anzahl von Feldern zum Aufnehmen von Dateneinträgen einer bestimmten Art, wie Zeichenketten, Zahlen oder Zeigern auf Daten, die sich irgendwo anders befinden, aufweisen. Eine relationale Datenbank weist eine beliebige Anzahl rechteckiger Tabellen auf. Jede Tabelle weist eine Gruppe von Datensätzen auf, wobei jeder Datensatz als eine Zeile der Tabelle bezeichnet wird. Jeder Datensatz in derselben Tabelle weist die gleiche Anzahl von Feldern auf. (Einige Felder in einem Datensatz können jedoch keine Daten enthalten, was durch einen NULL-Wert angegeben wird.) Die Felder einer Tabelle bilden einen Satz von Spalten, die festgelegte Namen aufweisen können, die nicht Teil der Daten selbst sind. Die Datensätze weisen keine externen Angaben auf, um sie individuell zu identifizieren. Vielmehr wird auf sie durch einen Schlüssel zugegriffen, der aus dem Inhalt irgendeiner Kombination der Felder besteht, so daß eine relationale Datenbank als ein Software-implementierter, nach dem Inhalt adressierbarer Speicher angesehen werden kann.

[0003] Ein Datenbankmanagementsystem (DBMS oder Datenbanksystem) ist eine Computersoftware zum Speichern, Unterhalten und Durchsuchen der Daten in einer Datenbank. Ein DBMS weist gewöhnlich Einrichtungen zum Erhöhen der Leistungsfähigkeit, der Zuverlässigkeit und der Integrität, wie Indizes, Protokolle und Datensatzsperren, auf. Es weist stets eine oder mehrere Schnittstellen zum Finden bestimmter Daten aus der Datenbank, und um diese Anfragen einer Suchmaschine zu präsentieren, auf. Die Maschine durchsucht die Datenbank und gibt dem Benutzer ein Ergebnis, gewöhnlich in Form einer relationalen Tabelle, zurück, das die Spezifikationen der Anfrage erfüllt.

[0004] Die am weitesten verbreitete Schnittstelle für relationale Datenbanken ist die strukturierte Anfragesprache ("Structured Query Language") (SQL). Wenngleich viele Varianten dieser Schnittstellensprache existieren, wurden vom American National Standards Institute (ANSI) und der International Standards Organization (ISO) Standardversionen definiert. Die meisten gegenwärtigen kommerziellen Verwirklichungen von SQL folgen diesen Standardversionen, wenngleich viele von ihnen Sprachenkonstruktionen zusätzlich zu den im Standard definierten oder mit einem anderen Grad an Konformität aufweisen.

[0005] Relationale Datenbanken und relationale Anfragesprachen behandeln Daten als einen Satz rechteckiger Tabellen. Viele Datenbanken sind jedoch konzeptionell mehrdimensional und beruhen auf Achsen, wie der Zeit {Tag, Monat, Jahr}, dem Ort {Laden, Stadt, Staat}, der Kategorie {Produkt, Produktgruppe}, einem Agierenden {Angestellter, Abteilung, Zweig}, der Bezahlung {Bargeld, Scheck, Kredit} usw. Ein Benutzer findet es häufig nützlich, solche Daten als eine Ansammlung von Ansammlungen anzusehen, und möchte sie aus verschiedenen Perspektiven betrachten. In dem vorstehenden Beispiel ist eine Perspektive eine Ansammlung von Datensätzen, wobei jeder Datensatz einen Ort repräsentiert und eine Ansammlung monatlicher Verkaufsdaten für diesen Ort enthält, eine andere Perspektive sieht eine Ansammlung von Datensätzen (also Zeilen einer Tabelle), wobei jeder einen bestimmten Zeitpunkt bezeichnet und die Felder jedes Datensatzes (also die Spalten der Tabelle) Verkaufszahlen für die verschiedenen Kategorien enthalten.

[0006] In dieser Hinsicht wäre die Fähigkeit zum Transformieren einer Datenbanktabelle von einer Perspektive in eine andere, also zum Drehen der Dimensionen der Daten, ein wertvoller Zusatz zu den herkömmlichen Fähigkeiten einer Anfragesprache, wie SQL. In diesem Zusammenhang bedeutet das Drehen von Perspektiven oder Dimensionen das Tauschen einer in einer Tabelle als ein Spaltensatz dargestellten Dimension gegen eine als ein Zeilensatz dargestellte Dimension. Herkömmliche relationale DBMS-Produkte und -Normen weisen keine direkte Operation zum Drehen von Perspektiven auf. Wenngleich es möglich ist, SQL-Anfragen zum indirekten Erreichen dieser Wirkung zu formulieren, sind diese Anfragen groß, komplex, fehleranfällig, langsam und schwer zu wirksamen Ausführungsplänen zu optimieren, selbst wenn eine Parallelverarbeitung verfügbar ist.

[0007] Einige herkömmliche Tabellenkalkulationsprogramme ermöglichen es einem Benutzer, Daten in einem vom Benutzer gewählten Rechteck von Zellen in der gleichen Weise auszutauschen, in der eine Matrix-Algebra-"Transponierungsoperation" ein Matrixelement a_{ij} zu a_{ji} relokalisiert. Beim Pivot-Tabellen-Merkmal von Microsoft Excel wählt ein Benutzer beispielsweise ein Rechteck von Zellen, kopiert es in eine Zwischenablage, zeigt auf eine Zielzelle und führt nach dem Auswählen von "Transponieren" aus einem Optionsmenü eine "Einfüge-Speziell-Operation" aus. Mit einem Paket kompatibler Anwendungsprogramme, wie Microsoft Office, kann ein Benutzer sogar Daten aus einer Datenbanktabelle in der Datenbankkomponente von Microsoft Access auswählen, sie als ein einzelnes Objekt in die Excel-Komponente als ein Rechteck von Tabellenkalkulationszellen übertragen, die Zellen transponieren und die Zellen dann als eine Ansammlung von Datensätzen in dem transponierten Format in die Access-Datenbank zurückübertragen. Pivot-Operationen bei Tabellenkal-

kulationen sind in J. C. Nossiter, Using Excel 5 For Windows Que Corp, 1995 und in B. Desmarais, "Using the Microsoft Excel Pivot Table for Reliability Applications", IEEE 34th Annual Spring Reliability Symposium (18. April 1996), S. 79–81, beschrieben.

[0008] Das Transponieren von Dateneinträgen auf diese Weise ist sowohl umständlich als auch funktionell beschränkt. Selbst bei kleinen Datenbanken ist das Aufrufen eines anderen Anwendungsprogramms lediglich zum Ausführen einer einzigen Anfrage verschwenderisch. Für große Datenbanken macht die übliche Anforderung, daß transponierte Daten im Speicher vorhanden sind, dieses Verfahren unmöglich. Für Client-/Server-Architekturen, bei denen Host-basierte Suchmaschinen verwendet werden, gibt es keine Möglichkeit, eine Verbindung mit einem Tabellenkalkulationsprogramm herzustellen, um die Operation auszuführen. In jeder Umgebung erfordert eine Transposition über ein Tabellenkalkulations-Arbeitsblatt einen manuellen Eingriff, so daß es dabei nicht möglich ist, daß eine Transposition einen internen Teil einer Anfrage innerhalb eines Datenbankprogramms bildet. Diese externen Operationen können nicht an den hockentwickelten Reformulierungs-, Umschreib- und anderen Optimierungsprozeduren herkömmlicher Datenbank-Anfrageprozessoren und anderer Suchmaschinen teilnehmen. Auf einer eher konzeptionellen Ebene verhindern grundlegende Unterschiede zwischen Tabellenkalkulations-Arbeitsblättern und relationalen Datenbanktabellen die gewünschten Transpositionstypen. Beispielsweise sind die Namen der Spalten oder Felder in einer Datenbanktabelle kein Teil der Tabelle selbst, und sie bilden keinen Datensatz der Tabelle in der Weise, in der Spaltenköpfe in einem Tabellenkalkulations-Arbeitsblatt eine Zeile von Zellen innerhalb des Arbeitsblatts sind. Das Transponieren eines Rechtecks von Zellen in einem Tabellenkalkulations-Arbeitsblatt kann eine Zellenspalte demgemäß nicht in die Spaltennamen umwandeln, wenn die Zeilen des Tabellenkalkulations-Arbeitsblatts den Datenbankprogrammen Datensätze in einer Tabelle zurückgeben.

[0009] Einige nicht-relationale Datenbanksysteme weisen Operationen auf, die Pivotisierungsoperationen bei Tabellenkalkulations-Arbeitsblättern ähneln. OLAP (rechnergestützte analytische Verarbeitung – on-line analytical processing) kann eine Drehoperation an einem mehrdimensionalen "Datenkubus" ausführen, wie in U. Flohr, "OLAP by Web", Byte, September 1997, S. 81–84, in E. Lindholm u. a., "Datamation's Feature Summary: OLAP Servers", Datamation, Mai 1985, S. 70–71, in M. Frank, "BrioQuery 3.5", DBMS Online, Februar 1996, in "OLAP and OLAP Server Definitions", (OLAP Council, 1995) und in C. B. Darling "Think Outside the OLAP Box", Datamation, 15. April 1996, S. 88–92, erwähnt wurde.

[0010] Die Ausführungsmaschine des Microsoft-SQL-Server-Produkts hat eine streng interne Operation zum Aufteilen jedes Eintrags einer Tabellenaktualisierung mit der Form (Zeilenkennung, alte Werte, neue Werte) innerhalb eines Stroms von Aktualisierungseinträgen in einen "Löscheintrag" und einen "Einfügungseintrag", wodurch gewisse Zeilen- und Spaltenwerte ausgetauscht werden, und eine ähnliche Operation zum Zusammenfassen eines "Löscheintrags" und eines "Einfügungseintrags" zu einem "Aktualisierungseintrag". Diese Operationen stehen Benutzern nicht zur Verfügung und können nicht an Benutzeranfragen teilnehmen. Das heißt, daß der Anfrageprozessor sie nur intern verwendet, um die wirksame Ausführung bestimmter Funktionen zu erleichtern, die während des Aktualisierens von Datenbanken ausgeführt werden.

[0011] Demgemäß könnte die Datenbanktechnik durch Bereitstellen einer Einrichtung zur schnellen, wirksamen Drehung von Perspektiven, insbesondere für relationale Datenbanken, erheblich erweitert werden. Weiterhin besteht ein Bedarf an Drehungs- oder Transpositionsoperationen, deren Semantik und Syntax sich gut in Anfragesprachen, wie SQL, als natürliche Erweiterungen integrieren lassen und welche in herkömmlich organisierten Datenbank-Anfrageprozessoren und anderen Suchmaschinen optimiert und ausgeführt werden können, ohne zusätzliche komplexe oder idiosynkratische Einrichtungen hinzuzufügen.

Zusammenfassung der Erfindung

[0012] Die in den Ansprüchen 1, 11 und 17 definierte vorliegende Erfindung sieht eine "Pivotisierungsoperation" zum Transformieren der Zeilen (Datensätze) und Spalten (Felder) einer Tabelle vor, wobei dieser Begriff in einer relationalen Datenbank definiert ist, um verschiedene Perspektiven für die Dateneinträge in der Tabelle bereitzustellen. Die Operation akzeptiert eine Eingangstabelle und eine Pivotisierungsspezifikation und erzeugt eine Ausgangstabelle. Sie findet in der Schnittstellensprachen-Organisation so statt, daß sie leicht in herkömmliche Datenbank-Anfrageprozessoren, Suchmaschinen und Server integriert werden kann. Die Operation gibt Daten in den Feldern spezifizierter Tabellendatensätze in das gleiche Feld verschiedener Datensätze, wobei die Werte von einer oder mehreren festgelegten Tabellenspalten als die Namen der Felder selbst verwendet werden. Daten in allen weiteren Spalten werden in einer pivotisierten Tabelle entsprechend den Datenwerten gruppiert.

[0013] Es ist manchmal einfacher, andere relationale Operationen an einer Datenbanktabelle aus einer anderen Perspektive auszuführen, selbst wenn das Endergebnis die ursprüngliche Perspektive aufweist. Daher sieht die in den Ansprüchen 7, 15 und 19 definierte Erfindung auch eine "Entpivotisierungsoperation" als eine Umkehrung der Pivotisierungseinrichtung vor. Es ist weiterhin manchmal erwünscht, eine gespeicherte Tabelle oder ein Zwischenergebnis zu entpivotisieren.

[0014] Diese Operationen vereinfachen zusammen mit einer einfachen und intuitiven Art des Aufnehmens von ihnen in Datenbankanfragen das Schreiben von Anfragen und machen sie weniger fehleranfällig. Sie verringern oder beseitigen beispielsweise den Bedarf, Tabellen mit sich selbst zu verbinden. Das Verfahren zum Aufrufen der Operationen ermöglicht ein tiefes Einbetten mehrerer Operationen mit einer einfachen und mächtigen Syntaxerweiterung und einer wohldefinierten Semantik und wendet ein vertrautes Programmiersprachenparadigma an. Durch das Zulassen von Text als Verfahrensargumente in Anfragen werden die Mächtigkeit und die Einfachheit der Verwendung der erweiterten SQL-Sprache verbessert. Weiterhin kann das Erweitern des auf diese Weise verfügbaren Satzes relationaler Algebraausdrücke auf nichtprozedurale Anfrageausdrücke auch auf andere Operationen, wie Probe, Oben und Rang ("sample, top and rank") angewendet werden.
[0015] Pivotisierungs- und Entpivotisierungsoperationen gemäß der Erfindung sind schon an sich mit vielen Typen von Datenmanipulationsprogrammen und Systemarchitekturen, insbesondere solchen, die relationale Datenbanken aufweisen, kompatibel. Diese Operationen können sowohl auf der Sprachenebene (beispielsweise durch intuitive Erweiterungen von SQL und anderen Anfragesprachen) als auch auf der Verarbeitungsebene (beispielsweise Anfrageoptimierung und -ausführung) in solche Systeme integriert werden.
[0016] Beim Integrieren von Daten von mehreren Datenbanken in eine einzige Daten-Warenhaus-Datenbank tritt häufig eine "Impedanzfehlanpassung" auf, wenn die mehreren Datenquellen voneinander verschiedene Formen oder Zeilen-/Spaltenverhältnisse aufweisen. Fast nach Definition können solche Datenbanken sehr groß sein. Das Normieren dieser Daten kann vom Zusammenhang abhängen, wobei das Speichern von Daten in pivotisierter Form oder eine Perspektive für ein Schema optimal oder sogar erforderlich sein kann, während für ein anderes Schema die entpivotisierte Form bevorzugt oder erforderlich sein kann. Daher kann das Hinzufügen von Pivotisierungs- und Entpivotisierungsoperationen die Kombination von Daten von verschiedenen Quellen, insbesondere bei großen Datenmengen, sehr begünstigen.
[0017] Die gemäß der Erfindung vorgesehenen neuen Operationen beschleunigen, selbst bei begrenzten Systemressourcen, auch die DBMS-Verarbeitung niedrigerer Ebene. Die erweiterbare Syntax und die klare Semantik der neuen Operationen erleichtern das automatische Erzeugen und Optimieren komplexer Anfragen, insbesondere beim Umschreiben von Anfragen für eine wirksamere Ausführung. Selbst rein interne DBMS-Funktionen, wie die Aktualisierungsverarbeitung für die Index- und Integritätserhaltung und andere Zwecke, können profitieren. Die Verarbeitung von SQL-Anfragen, die IN-, ODER- und VEREINIGUNGS-Anfragen ("IN, OR and UNION queries") beinhalten, kann erweitert werden. Viele Optimierungstechniken, die bereits für GRUPPIEREN-NACH-Anfragen ("GROUP BY queries") verwendet werden, sind routinemäßig an das Verarbeiten von Pivotisierungs- und Entpivotisierungsanfragen anpaßbar. Herkömmliche Ausführungsalgorithmen, die Parallelverarbeitungstechniken für diese Anfragen aufweisen, gelten für das Pivotisieren von Tabellen oder Anfrageergebnissen unter Einschluß nicht sortierter und partitionierter Tabellen und Ergebnisse.
[0018] Andere Merkmale und Vorteile der Erfindung sowie Variationen, die innerhalb des Schutzumfangs der Erfindung liegen, werden Fachleuten beim Lesen der folgenden detaillierten Beschreibung einfallen.

Kurzbeschreibung der Zeichnung

[0019] **Fig.** 1 ist ein Blockdiagramm einer Computernetzwerkumgebung für die Erfindung.
[0020] **Fig.** 2 ist ein Diagramm eines Datenbankmanagementsystems zum Aufnehmen der Erfindung.
[0021] **Fig.** 3 ist ein Flußdiagramm der vom DBMS aus **Fig.** 2 ausgeführten Funktionen.
[0022] **Fig.** 4 zeigt Beispiele von Pivotisierungs- und Entpivotisierungsoperationen gemäß der Erfindung.
[0023] **Fig.** 5 ist ein Flußdiagramm einer Pivotisierungsoperation gemäß der Erfindung.
[0024] **Fig.** 6 ist ein Flußdiagramm einer Entpivotisierungsoperation.

Detaillierte Beschreibung

Als Beispiel dienende Betriebsumgebung

[0025] Datenbankmanagementsysteme werden in vielen verschiedenen Typen von Datenverarbeitungssystemen, einschließlich alleinstehender Personalcomputer, Mittelrechner und Großrechner, Peer-to-Peer- und Client/Server-Netzwerke und verteilter Weitbereichssysteme vieler Architekturen, implementiert. Alle Datenverarbeitungssysteme sind geeignete Umgebungen für die vorliegende Erfindung. Die Erfindung wird jedoch zu Erläuterungszwecken in Zusammenhang mit einem in **Fig.** 1 dargestellten herkömmlichen Client/Server-Computersystem **100** beschrieben. Netzwerkleitungen **110** verbinden eine Anzahl von Personalcomputern (PCs) **120** über Netzwerkadapter **121** und **131** mit einem Server **130**. Der Server **130** weist ein Speicheruntersystem **132** zum Aufnehmen der großen Datenmengen in typischen Unternehmensdatenbanken auf. Andere Systemarchitekturen sind auch geeignete Umgebungen für die Erfindung. Beispielsweise können die Einheiten **120** mit einem Großrechner oder einem Mittelrechner **130** verbundene Endgeräte sein, oder die Einheit **130** kann selbst einen PC aufweisen, der mit PCs **120** in einem Peer-to-Peer-Netzwerk gekoppelt ist. Für kleine

und mittlere Datenbanken kann das Gesamtsystem **100** einen einzigen PC aufweisen, der sowohl als Client als auch als Server wirkt. Ebenso kann der Dateispeicher unter einer Anzahl verschiedener Maschinen verteilt sein. **Fig.** 1 zeigt schematische Darstellungen eines externen Speichermediums **133**, das Client- und Server-Software zum Verteilen und Herunterladen zu Clients aufweist, und eines anderen Mediums **134** in der Art einer Diskette zum Speichern von Datenbanktabellen außerhalb des Rechners.

[0026] **Fig.** 1A und die folgende Erörterung sollen eine kurze allgemeine Beschreibung eines Personalcomputers **120** liefern. Wenngleich dies nicht erforderlich ist, wird die Erfindung im allgemeinen Zusammenhang Computer-ausführbarer Anweisungen, wie Programmodule, die von einem Personalcomputer ausgeführt werden, beschrieben. Generell umfassen Programmodule Routinen, Programme, Objekte, Komponenten, Datenstrukturen usw., die bestimmte Aufgaben ausführen oder bestimmte abstrakte Datentypen implementieren. Weiterhin werden Fachleute verstehen, daß die Erfindung zusammen mit anderen Computersystemkonfigurationen, einschließlich handgehaltener Vorrichtungen, Mehrprozessorsysteme, Mikroprozessor-basierter oder programmierbarer Endverbraucherelektronik, Netzwerk-PCs, Minicomputer, Mittelrechner und dergleichen, verwirklicht werden kann. Die Erfindung kann auch in verteilten Computerumgebungen verwirklicht werden, in denen Aufgaben durch Fernverarbeitungsvorrichtungen ausgeführt werden, die über ein Kommunikationsnetzwerk miteinander verbunden sind. In einer verteilten Computerumgebung können sich Programmodule sowohl in lokalen als auch in fernen Speichervorrichtungen befinden.

[0027] **Fig.** 2 ist ein Blockdiagramm eines typischen herkömmlichen Client/Server-Datenbankmanagementsystems **200**, das in dem System **100** aus **Fig.** 2 arbeiten kann. Ein Client-Anwendungsprogramm **210** wird innerhalb jedes PCs **120** unter einem PC-Betriebssystem **220**, wie Microsoft Windows **95**, ausgeführt. Unter anderen Funktionen enthält die Client-Anwendung **210** eine Einrichtung **211** zum Annehmen von Datenbankanfragen von einem Benutzer an einem PC **120**. Zusätzlich zu Benutzereingaben können andere Anwendungsprogramme **230**, die in einigen der PCs **120** ausgeführt werden, über vordefinierte Host-Sprachen-Anwendungsprogrammschnittstellen (APIs) **231** Anfragen an den DBMS-Client **210** richten.

[0028] Innerhalb des Servers **130** wird eine DBMS-Serveranwendung **240** in der Art des Microsoft-SQL-Servers unter einem Server-Betriebssystem **250**, wie Microsoft Windows NT, ausgeführt. Das DBMS-Programm **240** liefert Dienste zum Erzeugen, Anfragen, Unterhalten und Modifizieren einer Anzahl durch eine Datenbank **260** beispielhaft angegebener relationaler Datenbanken. Das Programm **240** kann die Dateisystemdienste **251** des Betriebssystems **250** einsetzen oder sein eigenes Dateisystem bereitstellen. Das Betriebssystem **250** könnte eine getrennte Ausprägung der gesamten DBMS-Anwendung für jede Anforderung von einem Client **210** ausführen. Zum Erzielen einer höheren Wirksamkeit weist das Programm **240** jedoch jeder Client-Verbindung einen getrennten Teilprozeß **242** im DBMS-Kern zu. Weiterhin kann dieser Teilprozeß ein maschinenspezifischer Betriebssystem-Teilprozeß sein, der alle Mechanismen von Windows NT zum Prozeßspeicherschutz, für einen besseren Zugriff auf Speichervorrichtungen usw. mitführt. Eine Suchmaschine **241** verarbeitet Anfragen und andere Anforderungen von einzelnen Clients **210**, die auf Tabellen **261** einer Datenbank **260** gerichtet sind, wie nachstehend vollständiger beschrieben wird. Sie erzwingt auch die Datenbankintegrität mit herkömmlichen Einrichtungen zum Sperren von Datensätzen, für atomare Transaktionen usw. Beim Microsoft-SQL-Server ist die Schnittstellensprache zwischen der Anfrageeinrichtung **211** und der Suchmaschine **241** Transact-SQL, welche die meisten Funktionen der Standard-ANSI-SQL-89- und ANSI-SQL-92-Sprachen zuzüglich Erweiterungen zum Bereitstellen einer größeren Flexibilität und Programmierbarkeit, bereitstellt.

[0029] **Fig.** 3 zeigt einige übliche Funktionen **300** der Suchmaschine **241** aus **Fig.** 2 zum Verarbeiten einer von einer der Client-Anwendungen **210** übertragenen Anfrage. SQL ist eine nicht-prozedurorientierte Sprache, weil eine SQL-Anfrage eine Spezifikation von Eigenschaften oder Prädikaten eines gewünschten Ergebnisses statt einer Folge von Schritten zum Erhalten des Ergebnisses ist. Das heißt, daß eine Anfrage wie WÄHLE Jahr, Quartal, Verkäufe AUS Narrow, WOBEI Verkäufe < (WÄHLE DURCHSCHNITT (Verkäufe) AUS Narrow) GEORDNET NACH Jahr, Quartal die Eigenschaften einer Ausgangstabelle spezifiziert. Die Spalten der Ausgangstabelle entsprechen den mit Jahr, Quartal und Verkäufe bezeichneten Spalten, die einer mit Narrow bezeichneten Eingangstabelle entnommen werden. Die Ausgangstabellen-Zeilen (Datensätze) sind nach dem Jahr und dann nach dem Quartal innerhalb jedes Jahr-Werts zu ordnen (also zu sortieren). Die Datensätze von der Eingangstabelle, die in der Ausgabe auftreten, sind nur jene, bei denen der Wert von Verkäufe kleiner ist als der Durchschnittswert aller Werte von Verkäufe in der mit Narrow bezeichneten Tabelle. Die eingebettete Unteranfrage WÄHLE DURCHSCHNITT (Verkäufe) AUS Narrow erzeugt eine Tabelle, die nur eine einzige Spalte und eine einzige Zeile aufweist, welche den Durchschnittswert von Verkäufe in der Narrow-Tabelle enthält . Die Art und die Folge, in der auf die Datensätze der Eingangstabelle zugegriffen wird, und andere Einzelheiten der Prozedur oder des Plans zum Aufbauen der Ausgangstabelle sind durch die Anfrage selbst nicht definiert.

[0030] Wenn die Suchmaschine **241** eine Anfrage empfängt, versetzt sie die Anfrage mit einem Parser in eine interne oder mit einem Token versehene Form, wie in Schritt **310** dargestellt ist. Der Prüfschritt **320** gewährleistet, daß die in der Anfrage benannten Daten in der Datenbank tatsächlich existieren, und es werden darin Daten- und Integritätsbedingungen geprüft. Er kann gewisse Teile der Anfrage, wie Makros und Ansichten, bei

321 erweitern. Der Ausgang **322** teilt dem Benutzer oder einer anderen Anfragequelle jegliche Fehler mit. Alle Suchmaschinen außer den sehr stark eingeschränkten führen bei der Anfrage eine umfangreiche Optimierung aus, wie in Schritt **330** angegeben ist. Die Optimierung kann ein Umschreiben der Anfrage durch Kombinieren oder Aufteilen von Abschnitten der Anfrage, Neuanordnen von Operationen und Unteranfragen usw. und andere Verfahren, wie eine Ablaufsteuerung von Zugriffen auf die Datensätze gespeicherter Datenbanktabellen und das Modifizieren von Funktionen, einschließen. Für jede Kandidatenausführungsstrategie wird ein Kostenwert berechnet, der die Rechenzeit oder Betriebsmittel darstellt, welche zum Ausführen der Anfrage unter Verwendung dieser Strategie erforderlich sind, und es wird dann eine Strategie unter allen möglichen Kandidaten ausgewählt. Wenngleich die Technik zum Entwickeln dieser Optimierer komplex und geheimnisvoll ist, sind Personen, die darin bewandert sind, in der Lage, herkömmliche Optimierer so anzupassen, daß sie neue Anfragefunktionen verschiedener Typen aufweisen, wobei Entwickler von Übersetzern für andere, eher prozedurorientierte Sprachen auch routinemäßig Optimierer dieser allgemeinen Klasse konstruieren. In einem Übersichtsartikel von M. Jarke und J. Koch "Query Optimization in Database Systems", ACM Computing Surveys 16, 2 (Juni 1984), S. 111 ist der Aufbau von Datenbankanfrageoptimierern in weiteren Einzelheiten erörtert.

[0031] Die Ausgabe von Schritt 330 ist ein Anfragebeurteilungsplan (oder einfach ein "Plan",) zum Beantworten der Anfrage. In Schritt **340** wird dieser Plan zu einer prozedurorientierten Form kompiliert, die gewöhnlich als ein herkömmlicher Funktionsbaum dargestellt wird. In Schritt **350** kann dann ein einfacher Baumdurchlaufalgorithmus zum Ausführen des Plans in bezug auf die Datenbankobjekte ablaufen gelassen werden. Die Ausgabe des Schritts **350** ist das Ergebnis der Anfrage in Form einer Ausgangstabelle, die an die Anfragequelle zurückgegeben wird. Andere Suchmaschinen als die hier beschriebene können die einzelnen Schritte **300** kombinieren oder aufteilen oder Schritte fortlassen oder hinzufügen. Ein anderer Übersichtsartikel von G. Graefe "Query Evaluation Techniques for Large Databases", ACM Computing Surveys 25, 2 (Juni 1993), S. 73, auf den hiermit verwiesen sei, richtet sich auf den Gegenstand der Anfrageausführung und zitiert eine Anzahl von Bezügen, in denen zusätzliche Beschreibungen und Erörterungen enthalten sind. Wiederum sind die Schritte der neuesten Suchmaschinen speziell für eine leichte Erweiterbarkeit ausgelegt, um neuer Syntax, neuen Anfragefunktionen, Optimierungskenntnissen und Ausführungstechnologie Rechnung zu tragen.

Pivotisierungs- und Entpivotisierungsoperationen

[0032] **Fig.** 4 zeigt die Struktur einer Pivotisierungsoperation gemäß der Erfindung. Diese Operation paßt in die Hierarchie von SQL-Operationen auf der Ebene einer relationalen Algebra.

[0033] Personen, die relationale Datenbanksysteme und Schnittstellen entwickeln, teilen die Anfrageverarbeitung in drei Ebenen ein. Weil die mathematische Relationstheorie den konzeptionellen Rahmen für diesen Typ von Datenbanken bereitstellt, werden die erste und die zweite Ebene häufig als die relationale Analysis und die relationale Algebra bezeichnet.

[0034] Die relationale Analysis beschäftigt sich wie jede Analysis mit der Beschreibung oder Spezifikation hoher Ebene eines gewünschten Ergebnisses, ohne jegliche Operationen, Prozeduren oder ein anderes Verfahren zum Erhalten des Ergebnisses zu benennen. Das heißt, daß sie lediglich die Definition einer gewünschten Ergebnisrelation (Tabelle) durch bestehende Relationen in einer Datenbank ausdrückt. Die Anfrage WÄHLE employee.name, department.name AUS employee, department WOBEI employee.dept_id = department.dept_id beschreibt beispielsweise die Eigenschaften und Bedingungen einer Ausgangstabelle in bezug auf eine oder mehrere Eingangstabellen für ein typisches Element der Ergebnisbeziehung sowie eine Qualifikation, die die definierende Eigenschaft der Ergebniselemente darstellt. Die relationale Analysis liefert die Grundlage für ein formales, exaktes Verständnis von Datenbanken, Tabellen ("Relationen") und Anfragen, und sie hat in den Anfragekomponenten der Datenbanksprache SQL, nun eine ANSI/ISO-Norm, eine kommerzielle Verwirklichung gefunden. Wegen der wichtigen Rolle, die SQL in Datenbankmanagementprodukten spielt, ist für das Erweitern der Datenbankfunktionalität auf die wirkliche Welt erforderlich, daß jede hinzugefügte Funktionalität zu einer syntaktisch und semantisch sauberen Erweiterung der SQL-Sprache wird.

[0035] Die relationale Algebra ist eher operationsorientiert als die relationale Analysis (jedoch damit äquivalent). Operationen oder Funktionen bei der relationalen Algebra benötigen eine oder mehrere Eingangstabellen und erzeugen entsprechend einer Regel eine Ausgangstabelle. Die relationale Operation VERBINDE [employee.dept_id = department.dept_id] (employee, department) kombiniert beispielsweise die Tabellen employee und department entlang einer in beiden Tabellen mit dept_id bezeichneten Spalte oder einem damit bezeichneten Feld. (Dies ist analog mit einer Operation in der Art der Addition, welche zwei Zahlen benötigt und eine dritte erzeugt, so wie die Operation "4 + 5" beispielsweise "9" erzeugt.) Schlüsselcharakteristiken der relationalen Algebra bestehen darin, daß (1) Operationen Objekte desselben Typs, nämlich Relationen, benötigen und erzeugen, (2) Operationen in beliebig komplexe Strukturen eingebettet werden können und (3) neue Operationen hinzugefügt werden können. Bei der relationalen Algebra weisen Eingangsobjekte nicht nur Eingaben auf, sondern sie können auch Kennzeichen mitführen, die zusätzliche Informationen bezeichnen. In dem unmittelbar zuvor angegebenen Beispiel spezifiziert die Verbindungsoperation nicht nur die zwei relatio-

nalen Algebraausdrücke, nämlich die zwei Tabellen (employee, department), die zu verbinden sind, sondern sie benennt auch ein "Verbindungsprädikat", welches spezifiziert, wie sie zu verbinden sind, nämlich entlang gleichen Werten einer bestimmten Spalte in jeder Tabelle [employee.dept_id = department.dept_id].

[0036] Einige sehr nützliche Anfrageoperationen lassen sich auf der Ebene der relationalen Analysis nur schwer ausdrücken, sie lassen sich jedoch leicht und sauber in die Ebene der relationalen Algebra integrieren. Beispielsweise läßt sich die Operation ÄUSSERES VERBINDEN, eine Variante der relationalen Verbindungsoperation, nicht leicht und sauber in die einfache WÄHLE ... AUS ... WOBEI-Anfragesyntax einpassen. Daher ermöglicht ANSI/ISO einen begrenzten Satz relationaler Algebraausdrücke an Stelle von Tabellen in der AUS-Klausel, beispielsweise WÄHLE employee.name department.name AUS employee LINKES ÄUSSERES VERBINDEN department BEI employee.dept_id = department.dept_id. Das heißt, daß es einen Vorläufer für das Erweitern einer relationalen Analysisanfrage mit einem relationalen Algebraausdruck gibt, wenngleich diese Erweiterungen bisher auf Variationen von Verbindungsoperationen beschränkt waren. Relationale Algebraoperationen nehmen häufig an der Optimierung von Anfragen mit Auswahlen, Projektionen, Aggregationen und anderen nicht prozeduralen Spezifikationen auf der relationalen Analysisebene teil, wie in Block **330** in **Fig.** 3 angegeben ist.

[0037] Anfrageausführungspläne bilden die dritte und niedrigste Ebene der Anfrageverarbeitung. Wenngleich das Einbetten relationaler Algebraoperationen eine Ausführungsreihenfolge angeben kann, treten Algorithmen oder Sätze bestimmter Anweisungen zum Erzeugen von Zwischenergebnissen auf der Ebene von Ausführungsplänen statt auf den höheren Ebenen auf. Es gibt beispielsweise drei Grundverfahren zum Ausführen relationaler Verbindungsoperationen, nämlich eingebettete Schleifen, Verschmelzen-Verbinden und Hash-Verbinden, und jedes Verfahren weist eine große Anzahl von Varianten auf. Ausführungspläne geben klar die Wahlmöglichkeiten unter diesen Alternativen an, und sie sind in Block **340** in **Fig.** 3 auf der niedrigsten Ebene der Anfrageverarbeitung formuliert.

[0038] Weil die relationale Anfrageverarbeitung sehr genau und innerhalb eines definierten Strukturrahmens festgelegt ist, ist es wichtig, jegliche neue Funktionalitäten auf allen drei Ebenen zu definieren, nämlich Sprachenerweiterungen, relationale Algebraoperationen und Ausführungspläne. Die Erfindung kann die Pivotisierungs- und Entpivotisierungsfunktionen als neue relationale Algebraoperationen bereitstellen, die als Erweiterungen der Sprache explizit an SQL-Anfragen teilnehmen.

[0039] Die formale Definition einer Pivotisierungsoperation für eine Eingabe Tabelle-Ausdruck in der ersten Normalform, die ein gültiger Anfrageausdruck ist, ist: Table.PIVOT (<value_column> FÜR <pivot_column> IN (<pivot_list)), und die pivotisierte Ausgangstabvelle ist dann auch eine gültige Tabelle mit der ersten Normalform. Der nächste Ausdruck zwischen den äußersten Klammern bildet die Spezifikation der Pivotisierungsoperation. Die ersten beiden Spalten in der Pivotisierungsspezifikation müssen Spalten in der Eingangstabelle der Pivotisierungsoperation sein. Diese Spalten erscheinen nicht in der Ausgangstabelle der Pivotisierungsoperation. Vielmehr definiert jeder Wert in der Pivotisierungsliste innerhalb der Pivotisierungsspezifikation eine neue Spalte in der Ausgangstabelle der Pivotisierungsoperation. In der Eingangstabelle erscheinen Elemente in der Pivotisierungsliste als Werte in der Pivot-Spalte. Entsprechende Werte in der Wertspalte werden zu Werten in den neuen Spalten in der Ausgangstabelle. Alle Spalten der Eingangstabelle, die nicht in der Pivotisierungsspezifikation enthalten sind und als "Gruppenspalten" bezeichnet werden, werden in die Ausgangstabelle übertragen.

[0040] In dem in **Fig.** 4 angegebenen Beispiel **400** wird durch Pivotisieren der Eingangstabelle **410** in Übereinstimmung mit der Spezifikation **420** eine Ausgangstabelle **430** erzeugt. Die mit Quartal bezeichnete Pivot-Spalte **411** in der mit Narrow bezeichneten Eingangstabelle **410** wird zu vier Spalten **431**, **432**, **433** und **434** in der Ausgangstabelle **430**. Die Namen dieser Spalten sind die bestimmten Werte Frühling, Sommer, Herbst, Winter, die als Werte in der Spalte **411** auftreten und die auch nach dem Schlüsselwort IN in **420** in der Wertliste auftreten. Die Verkaufszahlen in der Wertspalte **412** erscheinen als Werte in entsprechenden der vier Spalten **431**–**434**, die jedoch pivotisiert oder gedreht sind, so daß die Verkaufszahlen für denselben Gebiet und dasselbe Jahr in derselben Zeile liegen. Gruppierungsspalten **413**–**414** erscheinen als Spalten **435**– **436** in der Ausgangstabelle **430**. In der Ausgangstabelle sind die Zeilen nach gleichen Werten der ersten Gruppierungsspalte **413** und dann nach gleichen Werten der zweiten Gruppierungsspalte **414** gruppiert, als ob die Spezifikation **420** eine SQL-Klausel der Form GRUPPIEREN NACH Gebiet, Jahr enthalten würde. In diesem Beispiel besteht die Wirkung der Pivotisierungsoperation darin, die Perspektive zu modifizieren, aus der die Daten betrachtet werden. Die Eingangstabelle **410** stellt. Datentrends in erster Linie nach dem Jahr für die Narrow-Gebiete einer Firma dar, während die Ausgangstabelle **430** ein saisonales Verfolgen nach Quartalen ermöglicht. (Es sei hier daran erinnert, daß die Zeilen in einer relationalen Tabelle keine Namen aufweisen und keine Reihenfolge haben. Spalten haben keine Namen und sind sortiert, so daß sie in der Reihenfolge präsentiert werden, in der ihre Namen in einer Anfrage auftreten.) Die Pivotisierungsoperation wandelt eine Eingangstabelle mit relativ vielen Zeilen und relativ wenigen Spalten in eine Ergebnistabelle mit weniger Zeilen und mehr Spalten um.

[0041] Die pivotisierten Spalten in der Ausgangstabelle weisen denselben Datentyp (numerisch, varchar

usw.) wie die Daten in der Wertspalte der Eingangstabelle auf. Die Wertspalte, die Pivot-Spalte und pivotisierte Spalten weisen einfache Daten an Stelle berechneter Ausdrücke auf. Die Reihenfolge der Spalten in beiden Tabellen ist nicht wesentlich, wie bei ANSI SQL, wobei Spalten nur nach dem Namen und nicht nach der Position angesprochen werden können. Wenngleich die Tabelle 430 als nach Werten von Gruppierungsspalten Gebiet und Jahr sortiert dargestellt ist, beinhaltet die Pivotisierungsoperation keine bestimmte Sortierung oder Reihenfolge der Zeilen.

[0042] Wie vorstehend erwähnt wurde, erscheint eine Zeile in der Eingangstabelle nicht in der Ausgabe, falls ihr Wert nicht in der Pivot-Liste auftritt. Die Zeilen der Eingangstabelle sind in bezug auf die Definition der Gleichheit nach gleichen Werten beliebiger Gruppierungsspalten gruppiert. Innerhalb jeder Gruppe hat jede Zeile der Eingangstabelle in der Pivot-Spalte einen jeweils bestimmten Wert. Jede Gruppe führt zu einer Ausgangszeile. Für Ausgangsspalten, die keine entsprechende Eingangszeile aufweisen, ist der Wert NULL, ein in SQL definierter spezieller Wert.

[0043] **Fig.** 5 ist ein Flußdiagramm **500** der von den Modulen **300** in **Fig.** 3 von der Suchmaschine **241** in **Fig.** 2 für eine Pivotisierungsoperation ausgeführten Schritte. Ein Block **510** empfängt eine Anfrage von einem Benutzer an einem Client-Endgerät **120** in **Fig.** 1 oder von einer anderen Quelle, wie hier beschrieben wurde. In Schritt **520** wird identifiziert oder ausgewählt, welche Tabelle **261** in der Datenbank **260** als die Eingangstabelle der Operation dienen soll. In Schritt 521 wird identifiziert, welche Spalte der Eingangstabelle als die Pivot-Spalte dienen soll, und in Schritt **522** wird identifiziert, welche Pivot-Spaltenwerte auf der Pivot-Liste an der Pivotisierung teilnehmen, und in Schritt **523** wird ausgewählt, welche Spalte der Eingangstabelle die Wertspalte ist. In Schritt **530** wird der Ausgang als eine andere Tabelle **261** konstruiert. In Schritt **531** wird eine getrennte pivotisierte Spalte für jeden Dateneintragswert in die Pivot-Liste eingetragen. In Schritt **532** werden die Gruppierungsspalten konstruiert, falls vorhanden. (Wie vorstehend erwähnt wurde, sind dies jegliche zusätzliche Spalten der Eingangstabelle, die nicht in der Pivotisierungsspezifikation identifiziert sind.) In Schritt **540** werden die Dateneintragswerte der Wertspalte in die Zeilen der Ausgangstabelle eingefügt, wie zuvor beschrieben wurde, wobei ein Verfahren in einer Transposition besteht, wie durch Schritt **541** angegeben ist. Eine andere Möglichkeit zum Ausdrücken dieser Transposition besteht darin, daß jeder Dateneintrag in der Wertspalte in eine der pivotisierten Spalten gegeben wird, nämlich in die Spalte, deren Name dem Datenwert in der Pivot-Spalte der Eingangstabelle gleicht. In Schritt **550** werden die Zeilen der Ausgangstabelle nach gleichen Werten beliebiger Gruppierungsspalten gruppiert. Schließlich wird in Schritt **560** die Ausgangstabelle in der Datenbank **260** aus **Fig.** 2 gespeichert.

[0044] Die Entpivotisierungsoperation ist die Umkehrung der Pivotisierungsoperation und ist formal als (table_expression|query_expression>.UNPIVOT (<value_column> FÜR <pivot_column> IN (<column_list>) definiert. Die Bedeutungen der Terme gleichen denen bei der Pivotisierungsoperation. Durch Anwenden einer Pivotisierungs- und einer Entpivotisierungsoperation mit derselben Spezifikation auf eine Eingangstabelle wird die Eingangstabelle in ihren Ausgangszustand zurückversetzt. In dem in **Fig.** 4 dargestellten Beispiel wird die Tabelle 410 durch Anwenden der Entpivotisierungsoperation **440** auf die pivotisierte Tabelle **430** wiederhergestellt, wobei die zwei benannten Spalten Verkäufe **412** und Quartal **411** die Spalten **431**–**434** ersetzen.

[0045] Andere Datenbanksystemoperationen können als Umkehrungen jeder anderen angesehen werden, wie die Gruppierungs/Aufteilungs- und Verschmelzungs/Zusammenführungs-Paare, die in M. Gyssens u. a. "Tables as a Paradigm for Querying and Restructuring", PROCEEDINGS 1996 ACM SIGMOD INTL. CONF. ON MANAGEMENT OF DATA, Montreal, Que., Kanada, 3.–6. Juni 1996, S. 93–103 erwähnt sind. Pivotisierung/Entpivotisierung sind auch Umkehrungen voneinander.

[0046] Für jede Zeile in einer Eingangstabelle erzeugt die Entpivotisierungsoperation allgemein eine Zeile einer Ausgangstabelle für jede pivotisierte Spalte. (Ein Null-Wert in einer pivotisierten Spalte erzeugt jedoch keine Ausgangszeile.) Alle Spalten in der Pivot-Liste müssen denselben Datentyp bei der Eingabe aufweisen, und die Einträge in der Wertspalte der Ausgabe weisen diesen Typ auf. Durch das Entpivotisieren einer Tabelle wird die Anzahl ihrer Zeilen erhöht und die Anzahl ihrer Spalten verringert. Wiederum ist mit der Entpivotisierungsoperation keine Zeilensortierung bei der Ausgabe verbunden, wenngleich in **Fig.** 4 eine nach Gruppierungsspaltenwerten sortierte Tabelle **410** dargestellt ist.

[0047] **Fig.** 6 ist ein Flußdiagramm **600** der von den Modulen **300** in **Fig.** 3 der Suchmaschine **241** in **Fig.** 2 ausgeführten Schritte für eine Entpivotisierungsoperation. In Schritt **610** werden die Entpivotisierungsoperation und ihre Spezifikation empfangen. Weil die Entpivotisierung als die Umkehrung der Pivotisierung definiert ist und eine pivotisierte Tabelle genau zu ihrer entpivotisierten Form wiederherstellt, ist die Spezifikation einer Entpivotisierung nicht komplementär zu derjenigen für eine Pivotisierung, sondern sie weist vielmehr genau die gleiche Form auf wie diejenige der Spezifikation, die die pivotisierte Tabelle zuerst erzeugt hat, wie bei **440** in **Fig.** 4 dargestellt ist. Wiederum kann in Schritt **610** die Operation von einer Benutzeranfrage oder einer anderen Quelle empfangen werden. In Schritt **620** wird die pivotisierte Tabelle, wie **261**, die zu entpivotisieren ist, identifiziert oder ausgewählt. Diese Tabelle braucht nicht tatsächlich in einer vorhergehenden Operation pivotisiert worden zu sein, dies ist jedoch normalerweise der Fall, so daß die Pivotisierungsoperation im allgemeinen verwendet wird, um eine anfängliche Drehung von Perspektiven zu erreichen und das Entpivotisieren im

allgemeinen nur verwendet wird, um eine Tabelle in einer sauberen, einfachen Weise in eine ursprüngliche, entpivotisierte Form zurückzuführen. In Schritt **621** wird die Pivot-Liste der Spezifikation verwendet, um anzugeben, welche Spalten die zu drehenden oder transponierenden pivotisierten Spalten sind. In den Schritten **622** und **623** werden die Namen der Wertspalte und der Pivot-Spalte identifiziert. Diese Namen werden in die Spezifikation aufgenommen, weil sie an keiner Stelle innerhalb der pivotisierten Tabelle auftreten (zumindest sofern eine vorhergehende Pivotisierung keine Seitentabelle erhalten hat, die diese Informationen enthält).

[0048] In Schritt **630** wird die Pivot-Tabelle aufgebaut, die durch die Entpivotisierungsoperation zu erzeugen ist. In den Schritten **631** und **632** werden die Pivot- und Wertspalten in der Pivot-Tabelle unter Verwendung der in den Schritten **622** und **623** zugeführten Namen gebildet. In Schritt **633** werden Gruppierungsspalten in der Pivot-Tabelle eingerichtet, wobei jeweils eine für jede der Spalten der pivotisierten Tabelle eingerichtet wird, die nicht in der in Schritt **610** empfangenen Entpivotisierungsspezifikation enthalten sind. In Schritt **640** werden die Dateneinträge von der pivotisierten Tabelle in die in den vorhergehenden Schritten aufgebaute entpivotisierte Tabelle transponiert. Namen der pivotisierten Spalten werden zu Dateneinträgen in verschiedenen Zeilen, und die Dateneinträge in der pivotisierten Spalte gehen in die neue Wertspalte in den Zeilen mit den jeweils gleichen Pivot-Spaltenwerten wie der Name der pivotisierten Spalte in der Originaltabelle (pivotisierten Tabelle), in der sie sich befunden haben, über. In Schritt **650** werden Zeilen nach gleichen Werten der Gruppierungsspalten-Zeilen gruppiert. In Schritt **660** wird die Tabelle in der Datenbank **260** aus **Fig.** 2 gespeichert.

[0049] Korrelationsvariablen sind innerhalb der Spezifikation von Pivotisierungs- und Entpivotisierungsoperationen nicht zulässig, weil die Pivot-, Wert- und pivotisierten Spalten einfache Daten und keine berechneten Werte sind. Diese neuen Operationen haben keine Bedeutung dafür, ob oder welche Korrelationsvariablen in einem Anfrageausdruck, auf den die Operationen angewendet werden, zulässig sind. Falls ANSI SQL das Definieren von Tabellen- und Spalten-Aliases für einen Anfrageausdruck zuläßt, in dem eine Pivotisierungs- oder Entpivotisierungsoperation nicht auftritt, ist es annehmbar, solche Aliases für den Anfrageausdruck einschließlich einer Pivotisierungs-/Entpivotisierungsoperation, jedoch nicht für den Anfrageausdruck ohne die Operation, zu definieren. Falls beispielsweise Table1 AS Table2 (col1, col2) zulässig ist, dann ist Table1.PIVOT (...) AS Table2 (col1, col2, ...) zulässig, jedoch Table1 AS Table2 (col1, col2).PIVOT (...) nicht zulässig.

[0050] Die Pivotisierungs- und die Entpivotisierungsoperationen können in Block **330** in **Fig.** 3 eine. beliebige Anzahl herkömmlicher Optimierungstechniken verwenden. Weil diese Operationen Teil der relationalen Algabraebene sind, kann ein algebraischer Anfrageoptimierer am geeignetsten sein, um Optimierungstechniken zu verwirklichen. Es können auch andere Optimierungsrahmen anwendbar sein.

[0051] Einige zusätzliche Optimierungstechniken können spezifische Eigenschaften der neuen Operationen verwenden. Offensichtlich kann ein benachbartes Paar von Pivotisierungs- und Entpivotisierungsoperationen einander aufheben, und sie können dann aus einer Anfrage entfernt werden. Ein Optimierer sollte erkennen, daß die Gruppierungsspalten in der pivotisierten Ausgangstabelle die Pivot- und Wertspalten funktionell festlegen und daher einen relationalen Schlüssel der Ergebnistabelle bilden. (Dies ähnelt sehr den Gruppierungsspalten bei einer herkömmlichen GRUPPIERENNACH-Operation.) Bei einer entpivotisierten Ausgangstabelle bestimmen die Gruppierungsspalten zusammen mit der Pivot-Spalte die Wertspalte. Diese Eigenschaften können das Schätzen der Anzahl der Ausgangszeilen unterstützen, um eine Selektivitätsschätzung und eine Anfragekostenberechnung zum Vergleichen alternativer Ausführungspläne auszuführen. Sie können auch beim Erzeugen von Bedingungen zum Anwenden von Neuschreibregeln beim Vereinfachen der Ausführung einer Anfrage nützlich sein. Falls eine Tabelle vertikal partitioniert ist, können eine Operation zum Wiederzusammensetzen vollständiger Zeilen und eine nachfolgende Entpivotisierung einander aufheben, wodurch beide Operationen beseitigt werden.

[0052] Eine konzeptionelle Ähnlichkeit der Pivotisierungsoperation mit einer SQL-Klausel GRUPPIEREN NACH ermöglicht, daß viele Techniken und Regeln zum Optimieren von Anfragen, die diese Klausel aufweisen, ebenso für die neuen Operationen geeignet sind. Typische Beispiele umfassen: (1) das Ziehen einer Pivotisierung über ein Verbinden, um die Gruppierungs-Eingabegröße zu verringern oder um einen wirksameren Verbindungsalgorithmus zu ermöglichen, (2) das Drücken einer Pivotisierung unter ein Verbinden, um die Verbindungseingabe zu verringern oder um wirksamere Ausführungspläne für die Pivotisierung einzusetzen, (3) das Verschmelzen zweier benachbarter Pivotisierungen, wobei effektiv eine von ihnen beseitigt wird, und (4) das Aufteilen einer Pivotisierung in zwei Teile, wobei dann einer der Teile durch eine Verbindung oder über eine Prozeßgrenze als eine lokale/globale Aggregation in einer parallelen Ausführungsumgebung gedrückt wird. Im allgemeinen können ein Anfrageprädikat an den Gruppierungsspalten und eine Projektionsoperation unter Einschluß von Ausdrücken, die zusätzliche Spalten berechnen, ebenso wie eine Gruppierungsoperation, durch eine Pivotisierungs-/Entpivotisierungsoperation (entweder darüber oder darunter) bewegt werden.

[0053] Bestimmte Anfrageprädikate lassen sich wirksamer implementieren und auch einfacher ausdrücken, wenn sie als Prädikate (also als Qualifikationen) in bezug auf eine Pivot-Ergebnistabelle behandelt werden. Beispielsweise läßt sich das Vergleichen zweier pivotisierter Spalten miteinander einfach ausdrücken und wirksam implementieren, während dasselbe Prädikat, wenn es auf die Pivot-Eingangstabelle angewendet wird, komplexe und unwirksame, verschachtelte Anfragen erfordert. Daher kann das Umschreiben der Anfrage, so

daß sie eine Pivotisierungs/Auswahl/Entpivotisierungs-Operationsfolge enthält, zum Optimieren solcher Anfragen verwendet werden. Es sei beispielsweise eine Anfrage zum Auswählen von Tabellenzeilen betrachtet, bei der die Verkäufe im Herbst die Verkäufe im Frühling in der Tabelle **410** in **Fig.** 4 übersteigen. Die pivotisierte Tabelle **430** kann diese Anfrage als einen Vergleich zwischen den Frühling- und Herbst-Spalten **431** und **433** Rechnung tragen, während es bei der ursprünglichen Tabelle **410** erforderlich ist, die Tabelle mit sich selbst zu verbinden, um den Vergleich vorzunehmen. Unter Verwendung der Tabellen aus **Fig.** 4 könnte eine solche größere Anfrage unter Verwendung von Pivotisierungs- und Entpivotisierungsoperationen (in einem herkömmlichen Mehrzeilenformat) die folgende sein:

```
(WÄHLE * AUS
Narrow.PIVOT (Verkäufe FÜR Quartal IN (Frühling, Sommer, Herbst,
Winter))
WOBEI Fall_Sales > Spring_Sales)
.UNPIVOT (Verkäufe FÜR Quartal IN (Frühling, Sommer, Herbst, Winter))
```

[0054] Für den Block **340** in **Fig.** 3 nützliche Ausführungspläne können von Fachleuten aus herkömmlichen Plänen zum Gruppieren von Operationen abgeleitet werden. Insbesondere werden sofort Pläne einfallen, die auf dem Verschleifen, Indexieren, Streamen, Sortieren und Hashen beruhen. Ein frühes Aggregations-Sortieren und ein Hybrid-Hashen sind verwendbare Varianten. Pivotisierungs-/Entpivotisierungsoperationen sind für Parallelausführungsumgebungen, einschließlich paralleler Algorithmen, wie Cluster-Maschinen mit einem geteilten Speicher, einem verteilten Speicher und einer geteilten Platte, geeignet. Die lokale/globale Aggregation wurde bereits als eine Möglichkeit erwähnt.

[0055] Entpivotisierungsoperationen benötigen nur einen Plan mit einer einzigen Eingabe und einer einzigen Ausgabe, der mehrere Ausgangsdatensätze für jeden Eingangsdatensatz erzeugt. Diese Operation kann leicht parallel an Cluster-Maschinen mit einem geteilten Speicher, einem verteilten Speicher und einer geteilten Platte ausgeführt werden.

Variationen und Erweiterungen

[0056] Eine Anzahl von Varianten und Erweiterungen der vorstehenden Ausführungsform können für einige Anwendungen, ob für sich oder in Kombination mit jeder anderen, verwendbar sein. Die offensichtlichsten sind natürlich das Ersetzen oder Erweitern von Notationskonventionen, wie der Punktaufruftrenner, und das Neuanordnen von Komponenten.

[0057] Die bisher beschriebenen Pivotisierungs- und Entpivotisierungsoperation beschränken die Spaltennamen in der pivotisierten Tabelle oder Ausgangstabelle. Spaltennamen bei der Standard-SQL müssen Zeichenfolgen ohne Leerstellen sein. Weil Spaltenwerte Leerstellen aufweisen können und das Pivotisieren Werte zu Namen ändert, kann leicht ein Verfahren entwickelt werden, um angegebene Bezeichner und Literale als Spaltennamen zu verwenden. Es wäre in gleicher Weise einfach, Spaltenwerte mit anderen Datentypen als Zeichenketten als druckbare und lesbare Darstellungen für Spaltennamen darzustellen. Herkömmliche Namensmanipulationen, wie das Verketten von Namen, könnten verwendbar sein. Beispielsweise könnte eine Pivot-Liste Spalten-Aliases unter Verwendung eines Schlüsselworts, wie ALS aus SQL, enthalten. (Falls in **Fig.** 4 die Quartals-Werte "1" bis "4" an Stelle der Jahreszeitennamen wären, könnte die Spezifikation von **420** folgendermaßen aussehen: (Verkäufe FÜR Quartal IN (1 ALS "Frühling", 2 ALS "Sommer", 3 ALS "Herbst", 4 ALS "Winter").) zusätzlich könnte ALS verwendet werden, um Pivot-Ergebnisspalten in einem beliebigen Zusammenhang umzubenennen, und benutzerdefinierte Funktionen könnten zum Umwandeln komplexer Spaltennamen oder zum Anpassen von Namen an spezifische Beschränkungen einer SQL-Implementation zugeführt werden.

[0058] Semantische Erweiterungen könnten das Pivotisieren und Entpivotisieren mehrerer Spalten in einem einzigen Schritt enthalten. Beispielsweise ersetzt die Operation **450** in **Fig.** 4 die drei Spalten **411**, **412** und **414** durch acht Spalten an Stelle der vier Spalten **431**–**434** der Ergebnistabelle **430**. Das heißt, daß der Satz von Pivot-Spalten ein kartesisches oder äußeres Produkt aller Pivot-Listen darstellt. Eine Konvention zum Bezeichnen der Pivot-Spalten könnte lediglich das Verketten der Namen, wie Verkäufe_1996_Frühling usw. beinhalten. Eine Mehrspalten-Entpivotisierungsoperation mit derselben Spezifikation könnte solche Namen wieder in ihre ursprüngliche Form decodieren, um eine wahre Umkehrung für diese Erweiterung bereitzustellen. Eine weitere Erweiterung würde ein Mehrspalten-Pivotisieren in Schritten ermöglichen. Es könnte beispielsweise erwünscht sein, eine weitere Pivotisierung auf die bereits pivotisierte Tabelle **430** um die Spalte **436** anzuwenden, um die vorstehend beschriebenen acht Spalten zu erzeugen. Statt die Tabelle **430** zu entpivotisieren und dann eine Mehrspalten-Pivotisieren anzuwenden, ermöglicht eine erweiterte Form eine Liste von Spalten an Stelle der Wertspalte, beispielsweise Wide.PIVOT ((Frühling, Sommer, Herbst, Winter) FÜR Jahr IN (1996,

1997)). Der Optimierer **340** und der Compiler **350** können diese Operationen einfach zu einem einzigen Ausführungsplan zusammenfassen.

[0059] Die Pivotisierungsoperation (jedoch nicht die Entpivotisierungsoperation) kann herkömmliche SQL-Aggregations- oder Gruppierungsfunktionen, wie MIN, SUM, AVG und sogar COUNT (Minimum, Summe, Durchschnitt und Zählwert) für die Wertspalte in Schritt **540** unterstützen. In diesem Fall kann die Beschränkung auf eine einzige Zeile je Gruppe aufgehoben werden. Natürlich könnte sich der Typ der neuen Spalte von demjenigen der ursprünglichen unterscheiden. Das folgende Beispiel, das auf **Fig.** 4 beruht, zeigt eine Anfrage unter Verwendung einer Aggregation:

(WÄHLE Jahr, Quartal, Verkäufe AUS Narrow) PIVOT (SUM(Verkäufe) FÜR Quartal IN (Frühling, Sommer, Herbst, Winter)

[0060] Diese Anfrage faßt Verkäufe für die Gebiete Ost und West zu einer einzigen Summe zusammen, die die gesamte Firma für jedes Jahr darstellt. Bei Versionen, bei denen Gruppierungsfunktionen zulässig sind, könnte die Implementation die implizite Anwendung einer bestimmten Funktion, wie SUM, in allen Fällen spezifizieren, in denen eine Pivotisierungsoperation andernfalls duplizierte Primärschlüssel in verschiedenen Zeilen der pivotisierten Tabelle erzeugen würde. Pivotisierungen mit einer Gruppierung können nicht umgekehrt werden, weil bei der Aggregation Informationseinzelheiten verlorengehen, und die gruppierte Ausgabe bei der Standard-SQL kann aus dem gleichen Grunde nicht umgekehrt werden. Wenngleich diese Erweiterung das Funktionieren einer Entpivotisierung als eine wahre Umkehrung verhindert, bewahrt eine Ausführungsform diese Fähigkeit durch Hinzufügen einer internen "Seitentabelle", die alle ursprünglichen Werte speichert.

[0061] Eine andere mächtige Erweiterung bietet die weitere Möglichkeit, die Liste von Literalspaltennamen bei einer Pivotisierungs- oder Entpivotisierungsoperation mit einer WAHL-Anfrage zu ersetzen. Bei einer komplexeren Verarbeitung würde die Hilfsanfrage zuerst ablaufen und dann die Liste pivotisierter Spalten unter Verwendung des Ergebnisses der Hilfsanfrage gebunden werden, so daß die Hilfsanfrage eine verschachtelte Compilierung und Ausführung in den Blöcken **340** und **350** in **Fig.** 3 erfordert. Die Ausführung erfordert das Berechnen des Anfrageausdrucks, der zu pivotisieren ist, sowie das Ausführen einer Anfrage in bezug auf das Ergebnis.

[0062] Die Pivot-Spezifikation könnte die Pivot-Liste vollständig fortlassen, und sie könnte stattdessen eine Standard-Anfrage bereitstellen. Beispielsweise könnte durch Fortlassen der Klausel IN (Frühling, Sommer, Herbst, Winter) in der Anfrage **420** eine Standard-Anfrage WÄHLE BESTIMMTES Quartal AUS Narrow ersetzt werden. Weil hierdurch bewirkt wird, daß die Anfrage **420** zweimal auf die Eingangstabelle Bezug nimmt, wäre es nützlich, einen zweckgebundenen Namen für eine Operations-Eingangstabelle, analog dem Namen "this" in C++, einzuführen. Die Operation **420** könnte dann zu folgendem werden:

Narrow.PIVOT (Verkäufe FÜR WÄHLE BESTIMMTES Quartal AUS EINGABE).

[0063] Statt eine Pivot-Liste zu benötigen, könnte die Entpivotisierungsoperation eine Spezifikation von allen außer den pivotisierten Spalten in der Eingabe der Operation ermöglichen. In dem wie vorstehend angegeben modifizierten Beispiel **400** könnte die inverse Operation folgendermaßen spezifiziert werden (siehe **440** in **Fig.** 4):

Wide.UNPIVOT (Verkäufe FÜR Quartal IN (Frühling, Sommer, Herbst, Winter))

oder folgendermaßen (**460**, **Fig.** 4):

Wide.UNPIVOT (Verkäufe ÜBER (Gebiet, Jahr)).

[0064] Das Unterstützen von UBER in diesem Zusammenhang erfordert das Bestimmen des Satzes pivotisierter Spalten von der Eingangstabelle und damit die Fähigkeit zum Verarbeiten von Hilfsanfragen, wie vorstehend beschrieben wurde. Die IN- und ÜBER-Klauseln können kombiniert werden, wodurch ermöglicht wird, daß eine oder mehrere Spalten zu einer pivotisierten Spalte sowie einer Gruppierungsspalte werden. Eine Situation, in der dies aus der Perspektive der Anwendung sinnvoll sein könnte, besteht im Einschluß der Frühlingsverkäufe in jede ausgegebene Zeile, um die Berechnung des Verkaufszuwachses seit dem ersten Quartal für jedes folgende Quartal zu ermöglichen.

[0065] In manchen Tabellen kann ein Spaltensatz mehr oder weniger orthogonal oder unabhängig sein, und es ist beispielsweise wahrscheinlich, daß die mit "Stadt" und "Monat" bezeichneten Spalten für alle Städte für alle Monate Tabelleneinträge aufweisen. Andere Spaltensätze sind hierarchisch, wie beispielsweise in einer "Orte"-Tabelle mit "Staat"-, "Stadt"- und "Laden"-Spalten, und ihre Daten sind spärlich, so daß sehr wenige Städte in mehreren Staaten auftreten und wenige Städte mehrere Läden aufweisen. Im letztgenannten Fall führt die Verwendung von zwei IN-Klauseln zu einer plumpen Syntax und Semantik bei einer Pivotisierungsoperation. Die Verwendung einer Liste von Pivot-Spalten an Stelle einer einzigen Pivot-Spalte vermindert dieses Problem jedoch. Das ANSI-SQL-Konzept von "Zeilenwerten" ist für diesen Fall geeignet. Typischerweise, wenn auch nicht immer, ist es bequemer, eine Anfrage als die Pivot-Liste statt als eine Liste von Literalspaltennamen zu spezifizieren. Eine als Beispiel dienende Form könnte beispielsweise die folgende sein: Locations.PIVOT (Verkaufsvolumen FÜR (Stadt, Laden) IN (WÄHLE Stadt, Laden AUS Geschäften)).

[0066] Die Pivotisierungs- oder Entpivotisierungsoperationen könnten auch bei der internen Operation von Anfrageprozessoren **300** in **Fig.** 3 verwendbar sein. Zusätzlich müssen Referenzintegritätskonstanten nur für

gelöschte Kandidatenschlüssel und neue Fremdschlüssel erzwungen werden, und unter Verwendung eines Pivotisierens bzw. eines Entpivotisierens oder einer ähnlichen Drehung könnten Lösch- und Einfügungseinträge, die zum selben Schlüsselwert gehören, zusammengefaßt werden, wodurch einige Integritätsprüfungen als redundant beseitigt werden könnten. Weiterhin kann bei einer herkömmlichen Anfrage mit einer sehr großen IN-Klausel unter Verwendung herkömmlicher Verbindungsverfahren, wie Schleifen, Indexierungen, Verschmelzungen und Hash-Verbindungen und ihrer Parallelverarbeitungsvarianten, eine von der Suchmaschine implizit aufgerufene interne Entpivotisierungsoperation eine einzige sehr komplexe Zeile, die viele Literale oder Parameter als Spalten enthält, in einen Zeilensatz abbilden, der mit Datenbanken abgeglichen werden kann. Ähnliche interne Aufrufe von Pivotisierungs- bzw. Entpivotisierungsoperationen können in ODER- und VEREINIGUNGS-Anfragen ("OR and UNION queries"), die häufig mit IN-Klauseln äquivalent sind, nützlich sein.

Patentansprüche

1. Von einem Computer (**500**) durchgeführtes Verfahren zum Transformieren von Daten von einer unpivotisierten Eingangstabelle (**410**) einer relationalen Datenbank in eine pivotisierte Ausgangstabelle (**430**),
dadurch gekennzeichnet, daß alle folgenden Schritte vollständig innerhalb des relationalen Datenbankmanagementsystems (**200**) ausgeführt werden, um Tabellen zu verarbeiten, die Zeilen und Spalten von Einträgen mit Datenwert-Einträgen sowie getrennt von den und in einem anderen Format als die Datenwert-Einträge gespeicherte Spaltennamen aufweisen, ohne daß auf ein und von einem anderen Gesamttabellenformat konvertiert wird, mit folgenden vom Computer ausgeführten Schritten:
Auswählen (**521**) des getrennt gespeicherten Namens einer ersten Spalte der Eingangstabelle als Pivot-Spalte,
Wählen (**523**) des getrennt gespeicherten Namens einer zweiten Spalte der Eingangstabelle als Wertspalte,
Umwandeln (**531**) eines Satzes von Einträgen, die als Datenwerte in der Pivot-Spalte gespeichert sind, direkt in Einträge, die als Spaltennamen-Einträge in der Ausgangstabelle gespeichert sind, die pivotisierte Spalten in der Ausgangstabelle benennen,
Anordnen (**540**) von Einträgen, die als Datenwerte in der Pivot-Spalte gespeichert sind, in entsprechende Datenwert-Einträge in entsprechend unterschiedlichen pivotisierten Spalten der Ausgangstabelle, und
Speichern (**560**) der Ausgangstabelle direkt in dem Datenbankmanagementsystem, ohne sie in ein anderes oder aus einem anderen Format zu konvertieren.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß der Anordnungsschritt für jeden Datenwert-Eintrag in der Wertspalte, der in der gleichen Zeile der Eingangstabelle wie ein bestimmter aus dem Satz an Datenwert-Einträgen in der Pivot-Spalte angeordnet ist, ein Anordnen des bestimmten einen Datenwert-Eintrags in einer gewissen der pivotisierten Spalten der pivotisierten Ausgangstabelle umfaßt, wobei die gewisse pivotisierte Spalte einen Namen aufweist, der dem bestimmten einen Datenwert-Eintrag in der Pivot-Spalte entspricht.

3. Verfahren nach Anspruch 1, gekennzeichnet durch einen weiteren Schritt zum Auswählen (**522**) gewisser aus dem Satz an Datenwert-Einträgen in der Pivot-Spalte als Pivot-Liste, wobei der Umwandlungsschritt lediglich Datenwert-Einträge in der Pivot-Liste in die Spaltennamen-Einträge umwandelt.

4. Verfahren nach Anspruch 3, dadurch gekennzeichnet, daß der Umwandlungsschritt keine Zeilen in der Eingangstabelle mit NULL-Datenwert-Einträgen in die Pivot-Spalte umwandelt.

5. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß die Eingangstabelle mindestens eine andere Spalte (**413**) als die Pivot- und die Wertspalte aufweist.

6. Verfahren nach Anspruch 5, gekennzeichnet durch ein Gruppieren (**532**) der Zeilen der Ausgangstabellen im Hinblick auf gleiche Werte der Datenwert-Einträge in der genannten mindestens einen anderen Spalte.

7. Von einem Computer ausgeführtes Verfahren (**600**) zum Transformieren von Daten von einer pivotisierten Eingangstabelle (**430**) einer relationalen Datenbank in eine unpivotisierte Ausgangstabelle (**410**),
dadurch gekennzeichnet, daß alle folgenden Schritte direkt innerhalb eines relationalen Datenbankmanagementsystems (**200**) ausgeführt werden, um Tabellen zu verarbeiten, die Zeilen und Spalten von Einträgen mit Datenwerten sowie Einträge mit Spaltennamen, die getrennt von den und in einem anderen Format als die Datenwert-Einträge gespeichert sind, aufweisen, ohne daß eine Konvertierung in ein anderes oder aus einem anderen Gesamttabellenformat durchgeführt wird, mit den folgenden vom Computer ausgeführten Schritten:
Auswählen (**623**) des getrennt gespeicherten Namens einer ersten der Spalten der Eingangstabelle als Pivot-Spalte,

Auswählen (**621**) mehrerer der getrennt gespeicherten Namen der Spalten der Eingangstabelle als Pivot-Liste,
Erzeugen (**632**) einer Wertspalte in der Ausgangstabelle mit einem ausgewählten Namen, der in einem Spaltennamen-Eintrag gespeichert ist, und mit mehreren getrennt gespeicherten Datenwert-Einträgen,
Anordnen (**640**) von Einträgen, die als Spaltennamen-Einträge in der Pivot-Liste gespeichert sind, in entsprechenden Datenwert-Einträgen in entsprechend unterschiedlichen der Wertspalte der Ausgangstabelle, und
Speichern (**660**) der Ausgangstabelle direkt in dem Datenbankmanagementsystem, ohne daß sie in ein anderes oder aus einem anderen Format konvertiert wird.

8. Verfahren nach Anspruch 7, dadurch gekennzeichnet, daß der Anordnungsschritt für jeden bestimmten Datenwert-Eintrag in jeder bestimmten Spalte der Eingangstabellen-Spalten in der Pivot-Liste ein Anordnen des bestimmten Datenwert-Eintrags in einem solchen Datenwert-Eintrag der Wertspalte der unpivotisierten Ausgangstabelle in einer Zeile umfaßt, die auch einen Datenwert-Eintrag in der Pivot-Spalte entsprechend der bestimmten Spalte der Eingangstabelle enthält.

9. Verfahren nach Anspruch 7, dadurch gekennzeichnet, daß die Eingangstabelle mindestens eine andere Spalte (**435**) als die Spalten in der Pivot-Liste aufweist.

10. Verfahren nach Anspruch 9, dadurch gekennzeichnet, daß ein Gruppierungsschritt die Zeilen der Ausgangstabelle im Hinblick auf gleiche Werte der Datenwert-Einträge in der genannten mindestens einen anderen Spalte gruppiert.

11. Relationales Datenbanksystem (**200**) mit einer Anzahl an Clients (**210**) und einer Suchmaschine (**240**) mit Modulen zum Parsen (**310**), Optimieren (**330**) und Ausführen (**350**) einer Anfrage (**420**, **450**) von einem der Clients, die eine Pivotisierungsoperation enthält, die angibt:
eine relationale Eingangstabelle (**410**) mit Zeilen und Spalten von Datenwerten und mit Spaltennamen, die getrennt von den Datenwerten in den Spalten gespeichert sind,
einen Namen einer Pivot-Spalte (**411**) in der Eingangstabelle, und
einen Namen einer Wertspalte (**412**) in der Eingangstabelle,
dadurch gekennzeichnet, daß die Suchmaschine Datenwerte in der Wertspalte um die Pivot-Spalte transponiert, wobei sie mindestens einige der Datenwerte in getrennt gespeicherte und formatierte Spaltennamen umwandelt, so daß sie eine pivotisierte Ausgangstabelle direkt aus der Eingangstabelle aufbaut, ohne eine der beiden Tabellen in ein anderes oder aus einem anderen Format zu konvertieren.

12. System nach Anspruch 11, wobei die Pivotisierungsoperation außerdem eine Pivot-Liste (**522**) von Datenwerten in der Pivot-Spalte angibt, dadurch gekennzeichnet, daß die Datenwerte in der Pivot-Spalte aufgrund der Dateneinträge in der Pivot-Liste transponiert werden.

13. System nach Anspruch 12, wobei die Eingangstabelle andere Spalten (**413**) als die Pivot- und die Wertspalte aufweist, dadurch gekennzeichnet, daß die Suchmaschine Zeilen der Ausgangstabelle im Hinblick auf gleiche Werte der Datenwert-Einträge in den anderen Spalten gruppiert (**550**).

14. System nach Anspruch 12, wobei eine Anfrage (**440**, **460**) von einem der Clients eine Entpivotisierungsoperation umfaßt, die angibt:
eine pivotisierte relationale Eingangstabelle (**430**) mit Zeilen und Spalten von Datenwerten und mit Spaltennamen, die getrennt von den Datenwerten in den Spalten gespeichert sind,
einen Namen (**411**) für eine Wertspalte,
einen Namen (**412**) für eine Pivot-Spalte, und
eine Pivot-Liste mit Namen von mindestens einigen der Spalten (**431** bis **434**) in der pivotisierten Eingangstabelle,
dadurch gekennzeichnet, daß die Suchmaschine die Spaltennamen in der Pivot-Liste um die Pivot-Spalte transponiert (**600**), wobei sie diese Spaltennamen in getrennt gespeicherte und formatierte Datenwerte umwandelt, so daß sie eine unpivotisierte Ausgangstabelle direkt aus der Eingangstabelle aufbaut, ohne eine von beiden Tabellen in ein anderes oder aus einem anderen Format zu konvertieren.

15. Relationales Datenbanksystem (**200**) mit einer Anzahl an Clients (**210**) und einer Suchmaschine (**240**) mit Modulen zum Parsen (**310**), Optimieren (**330**) und Ausführen (**350**) einer Anfrage (**440**, **460**) von einem der Clients, die eine Entpivotisierungsoperation enthält, die angibt:
eine relationale Eingangstabelle (**430**) mit Zeilen und Spalten von Datenwerten und mit Spaltennamen, die getrennt von den Datenwerten in den Spalten gespeichert sind,
einen Namen für eine Pivot-Spalte (**412**), und

eine Pivot-Liste mit Namen von mindestens einigen der Spalten (**431** bis **434**) in der Eingangstabelle,
dadurch gekennzeichnet, daß die Suchmaschine die Spaltennamen in der Pivot-Liste um die Pivot-Spalte transponiert (**600**), wobei sie diese Namen in getrennt gespeicherte und formatierte Datenwerte umwandelt, so daß sie eine pivotisierte Ausgangstabelle direkt aus der Eingangstabelle aufbaut, ohne eine von beiden Tabellen in ein anderes oder aus einem anderen Format zu konvertieren.

16. System nach Anspruch 15, wobei die Eingangstabelle andere Spalten (**435**) als die Pivot- und die Wertspalte aufweist, dadurch gekennzeichnet, daß die Suchmaschine Zeilen der Ausgangstabelle im Hinblick auf gleiche Werte der Datenwert-Einträge in den anderen Spalten gruppiert.

17. Medium (**133**) mit Darstellungen von Anweisungen, um einen geeignet programmierten Computer zu veranlassen, ein Verfahren (**500**) zum Transformieren von Daten aus einer unpivotisierten Eingangstabelle (**410**) einer relationalen Datenbank in eine pivotisierte Ausgangstabelle (**430**) auszuführen,
dadurch gekennzeichnet, daß alle folgenden Schritte vollständig innerhalb eines relationalen Datenbankmanagementsystems (**200**) ausgeführt werden, um Tabellen zu verarbeiten, die Zeilen und Spalten von Einträgen mit Datenwert-Einträgen sowie Einträge mit Spaltennamen, die getrennt von den und in einem anderen Format als die Datenwert-Einträge gespeichert werden, aufweisen, ohne daß eine Konvertierung in ein anderes oder aus einem anderen Gesamttabellenformat stattfindet, mit den folgenden Schritten:
Auswählen (**521**) des getrennt gespeicherten Namens einer ersten der Spalten der Eingangstabelle als Pivot-Spalte,
Auswählen (**523**) des getrennt gespeicherten Namens einer zweiten der Spalten der Eingangstabelle als Wertspalte,
Umwandeln (**531**) eines Satzes an Einträgen, die als Datenwerte in der Wertspalte gespeichert sind, direkt in Einträge, die als Spaltennamen-Einträge in der Ausgangstabelle, gespeichert sind, die pivotisierte Spalten in der Ausgangstabelle benennen,
Anordnen (**540**) von Einträgen, die als Datenwerte in der Pivot-Spalte gespeichert sind, in entsprechende Datenwert-Einträge in entsprechend unterschiedlichen der pivotisierten Spalten der Ausgangstabelle, und
Speichern (**560**) der Ausgangstabelle direkt in dem Datenbankmanagementsystem, ohne sie in ein anderes oder aus einem anderen Format zu konvertieren.

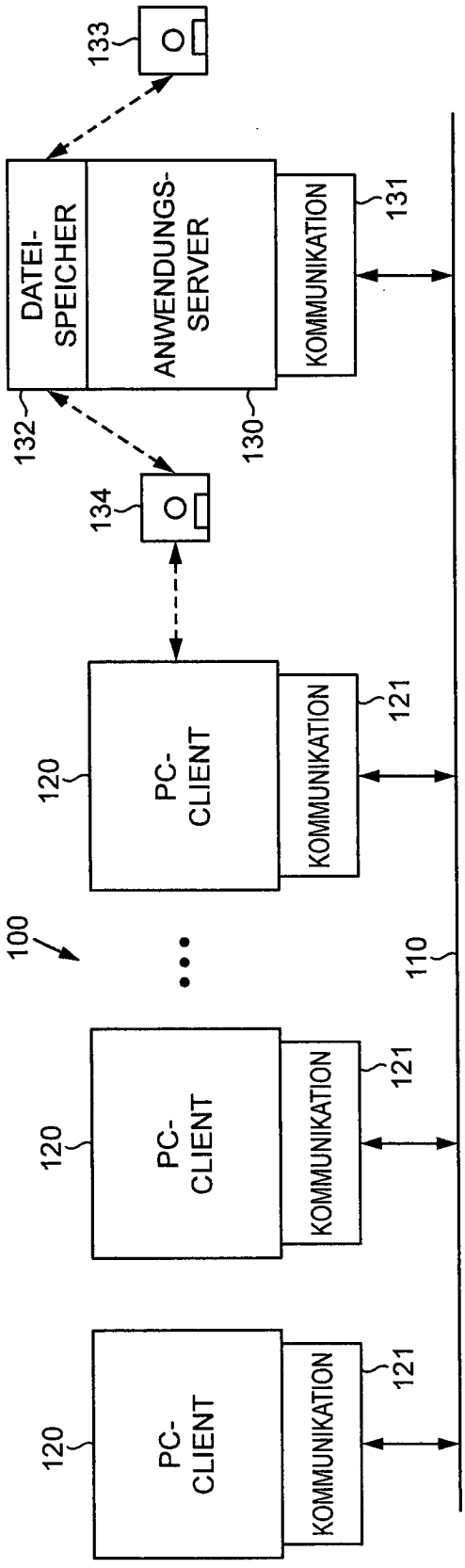
18. Medium nach Anspruch 17 mit Darstellungen von Anweisungen, um einen geeignet programmierten Computer zu veranlassen, ein Verfahren zum Transponieren von Daten aus einer pivotisierten Eingangstabelle einer relationalen Datenbank in eine unpivotisierte Ausgangstabelle in der gleichen relationalen Datenbank zu transformieren,
dadurch gekennzeichnet, daß alle folgenden Schritte vollständig innerhalb eines relationalen Datenbankmanagementsystems ausgeführt werden, um Tabellen zu verarbeiten, die Zeilen und Spalten von Einträgen mit Datenwerten sowie Datenwerte mit Spaltennamen, die getrennt von den und in einem anderen Format als die Datenwerteinträge gespeichert sind, aufweisen, ohne daß eine Umwandlung in ein anderes oder aus einem anderen Gesamttabellenformat stattfindet, wobei die Schritte aufweisen:
Auswählen des getrennt gespeicherten Namens einer ersten der Spalten der Eingangstabelle als Pivot-Spalte,
Auswählen einer Vielzahl der getrennt gespeicherten Namen der Spalten der Eingangstabelle als Pivot-Liste,
in der Ausgangstabelle Erzeugen einer Pivot-Spalte mit einem ausgewählten Namen, der in einem Spaltennamen-Eintrag gespeichert ist, und einer Vielzahl getrennt gespeicherten Datenwert-Einträge,
Umwandeln der Spaltennamen-Einträge in der Pivot-Liste in Datenwert-Einträge in der Pivot-Spalte,
in der Ausgangstabelle Erzeugen einer Wertspalte mit einem ausgewählten Namen, der in einem Spaltennamen-Eintrag gespeichert ist und einer Vielzahl separat gespeicherter Datenwert-Einträge,
Anordnen von Einträgen, die als Spaltennamen-Einträge in der Pivot-Liste gespeichert sind, in entsprechende Datenwert-Einträge in entsprechend unterschiedlichen der Wertspalte der Ausgangstabelle, und
Speichern der Ausgangstabelle direkt in dem Datenbankmanagementsystem, ohne sie in ein anderes oder aus einem anderen Format umzuwandeln.

19. Medium (**133**) mit Darstellungen von Anweisungen, um einen geeignet programmierten Computer zu veranlassen, ein Verfahren (**600**) zum Transformieren von Daten aus einer pivotisierten Eingangstabelle (**430**) einer relationalen Datenbank in eine unpivotisierte Ausgangstabelle (**410**) umzuwandeln,
dadurch gekennzeichnet, daß alle folgenden Schritte vollständig innerhalb eines relationalen Datenbank-managementsystems (**200**) ausgeführt werden, um Tabellen zu verarbeiten, die Zeilen und Spalten von Einträgen mit Datenwerten sowie Einträge mit Spaltennamen, die getrennt von den und in einem anderen Format als die Datenwert-Einträge gespeichert sind, aufweisen, ohne daß eine Umwandlung in ein anderes oder aus einem anderen Tabellenformat stattfindet, mit den folgenden Schritten:
Auswählen (**623**) des getrennt gespeicherten Namens einer ersten der Spalten der Eingangstabelle als Pi-

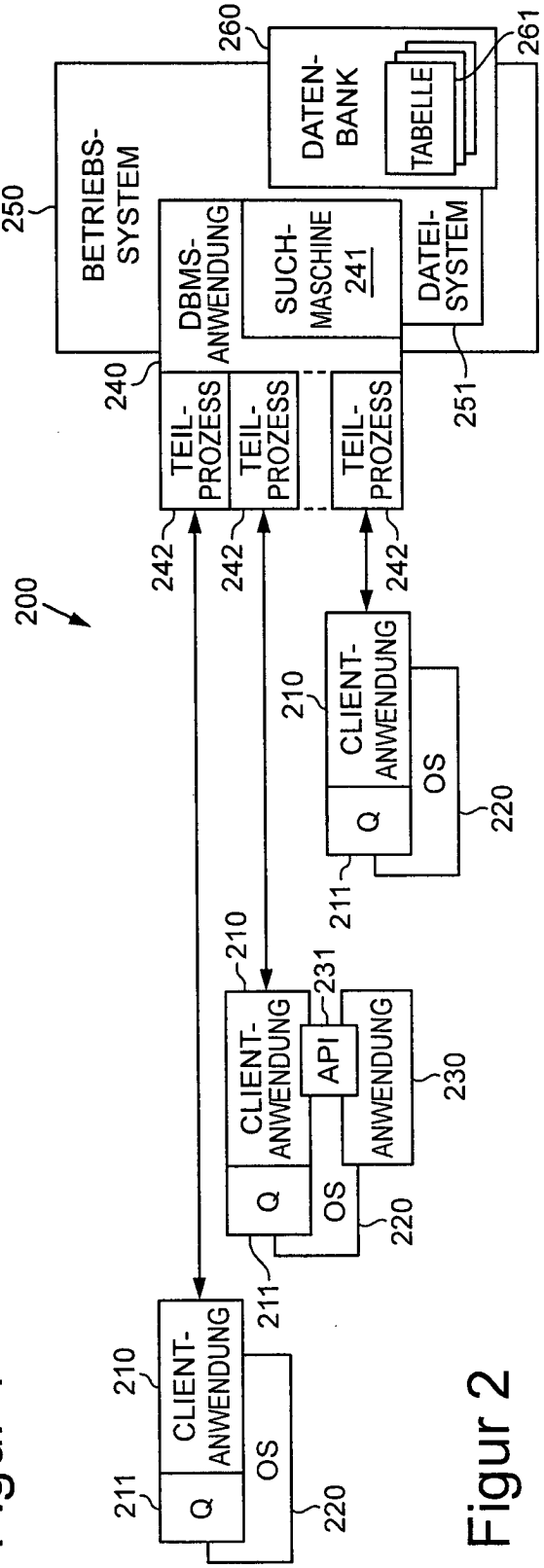
vot-Spalte,
Auswählen (**621**) einer Vielzahl der getrennt gespeicherten Namen der Spalten der Eingangstabelle als Pivot-Liste,
in der Ausgangstabelle Erzeugen **(632**) einer Wertspalte mit einem ausgewählten Namen, der in einem Spaltennamen-Eintrag gespeichert ist, und einer Vielzahl getrennt gespeicherter Datenwert-Einträge,
Anordnen (**640**) von Einträgen, die als Spaltennamen-Einträge in der Pivot-Liste gespeichert sind, in entsprechende Datenwert-Einträge in entsprechenden anderen der Wertspalte der Ausgangstabelle, und
Speichern (**660**) der Ausgangstabelle direkt in dem Daten bankmanagementsystem, ohne sie in ein anderes oder aus einem anderen Format zu konvertieren.

Es folgen 4 Blatt Zeichnungen

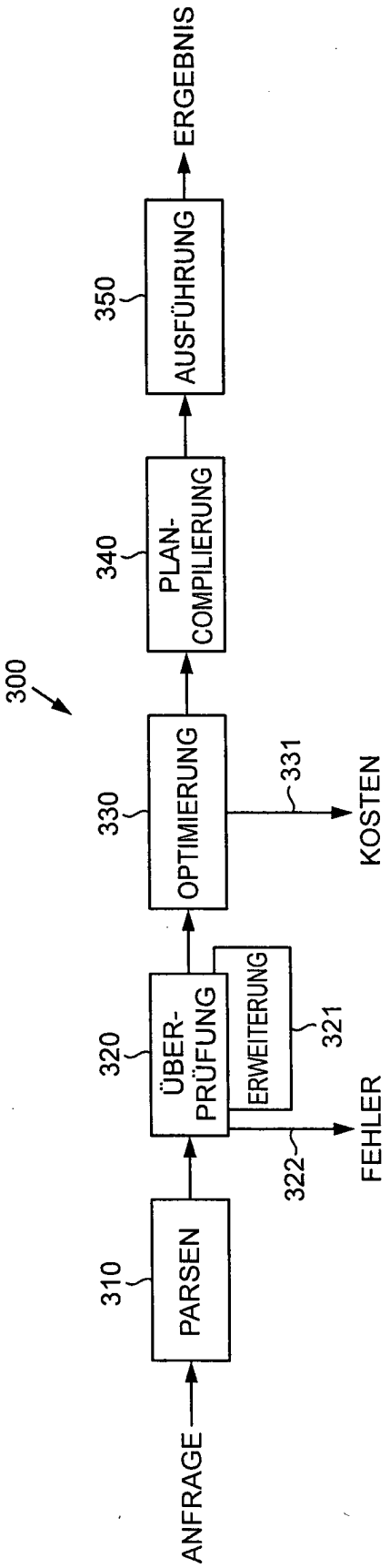
Anhängende Zeichnungen



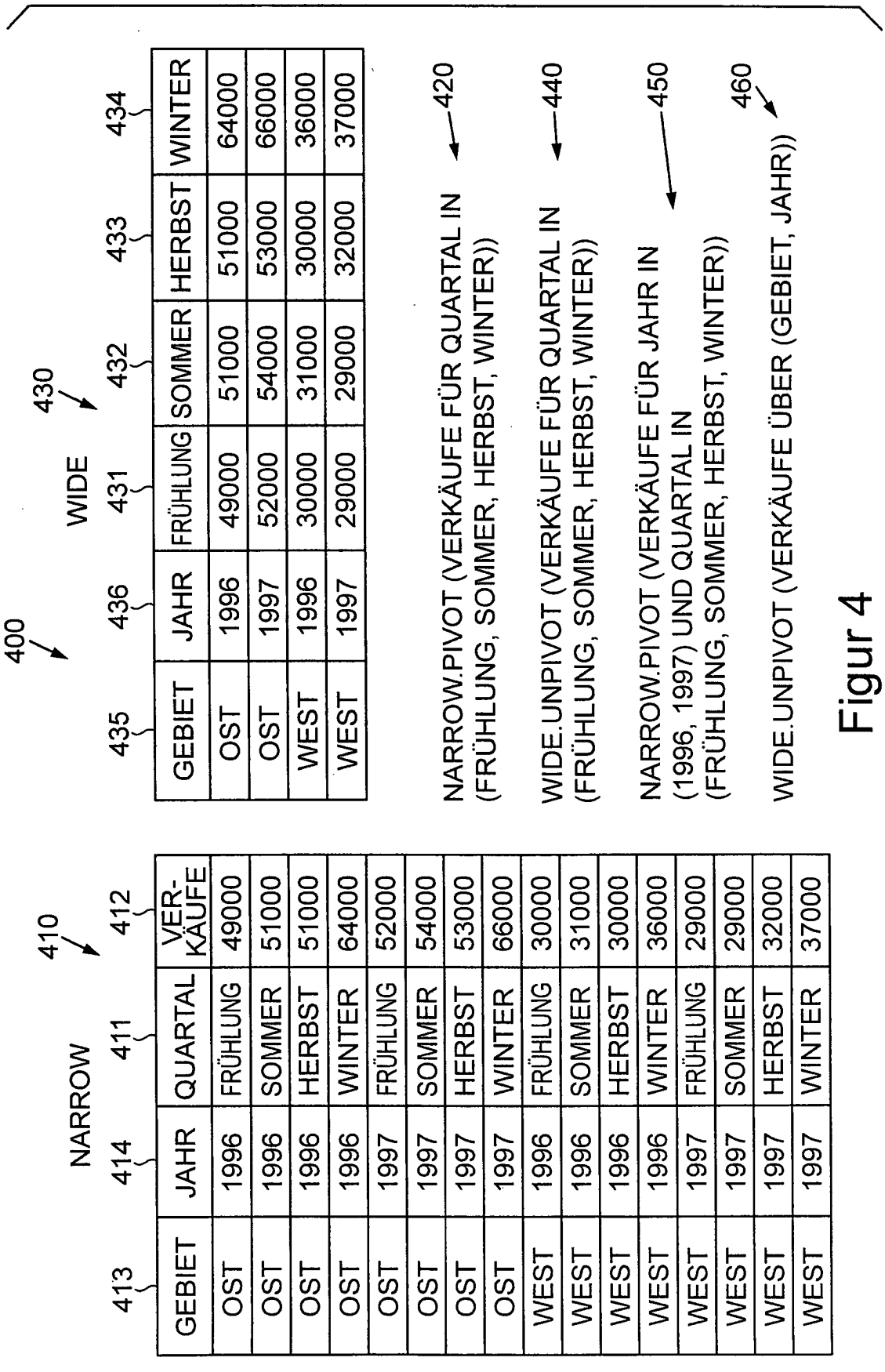
Figur 1



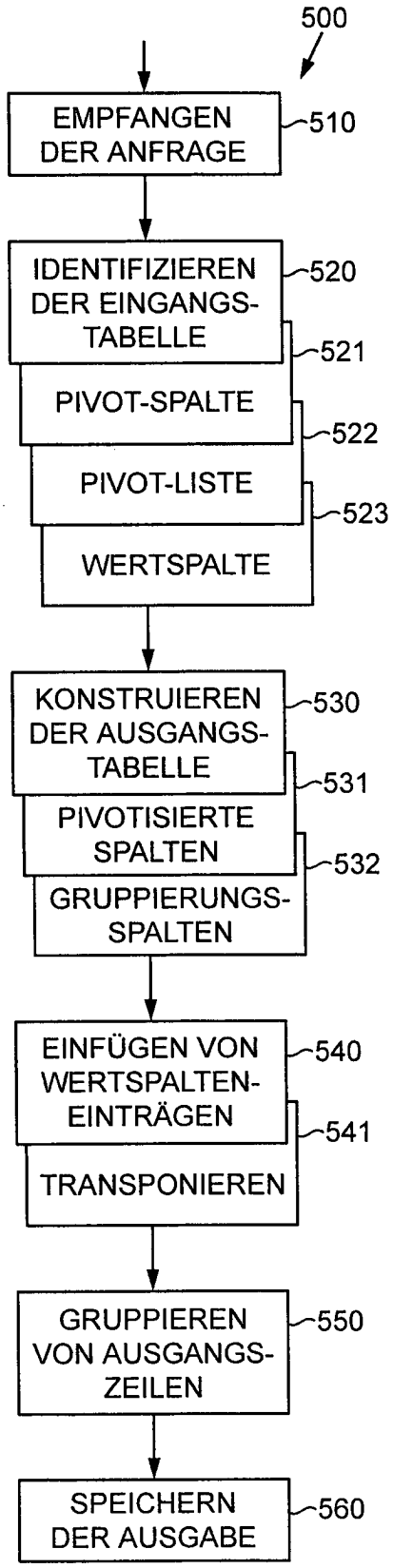
Figur 2


ANFRAGE
310
PARSEN
320
ÜBER-PRÜFUNG
ERWEITERUNG
321
322
FEHLER
330
OPTIMIERUNG
331
KOSTEN
340
PLAN-COMPILIERUNG
350
AUSFÜHRUNG
ERGEBNIS
300

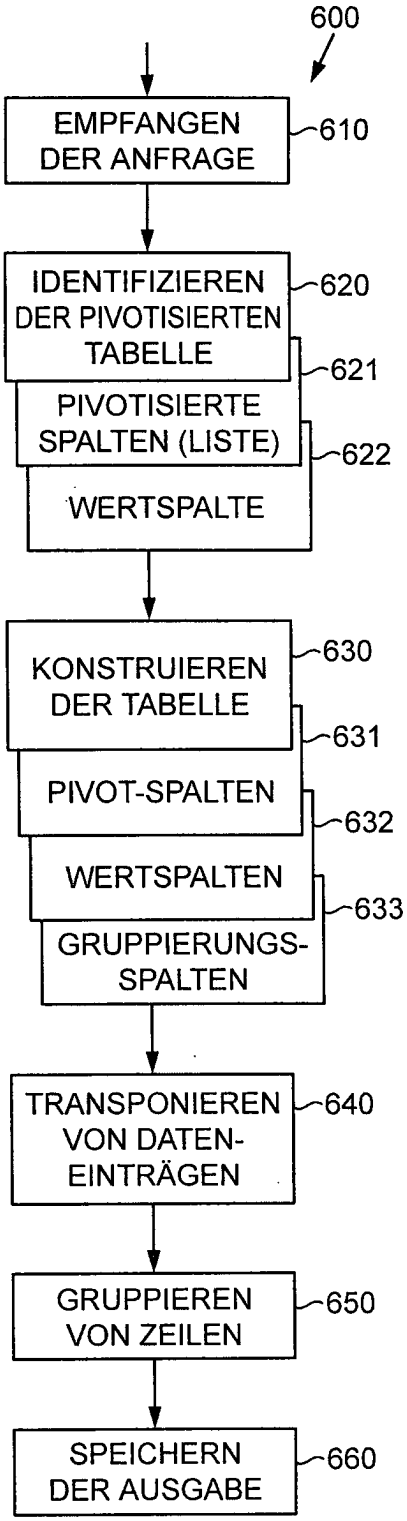
Figur 3


400
NARROW
410
413
414
411
412
GEBIET
JAHR
QUARTAL
VER-KÄUFE
OST 1996 FRÜHLUNG 49000
OST 1996 SOMMER 51000
OST 1996 HERBST 51000
OST 1996 WINTER 64000
OST 1997 FRÜHLUNG 52000
OST 1997 SOMMER 54000
OST 1997 HERBST 53000
OST 1997 WINTER 66000
WEST 1996 FRÜHLUNG 30000
WEST 1996 SOMMER 31000
WEST 1996 HERBST 30000
WEST 1996 WINTER 36000
WEST 1997 FRÜHLUNG 29000
WEST 1997 SOMMER 29000
WEST 1997 HERBST 32000
WEST 1997 WINTER 37000
WIDE
430
435
436
431
432
433
434
GEBIET
JAHR
FRÜHLUNG
SOMMER
HERBST
WINTER
OST 1996 49000 51000 51000 64000
OST 1997 52000 54000 53000 66000
WEST 1996 30000 31000 30000 36000
WEST 1997 29000 29000 32000 37000
NARROW.PIVOT (VERKÄUFE FÜR QUARTAL IN (FRÜHLUNG, SOMMER, HERBST, WINTER))
420
WIDE.UNPIVOT (VERKÄUFE FÜR QUARTAL IN (FRÜHLUNG, SOMMER, HERBST, WINTER))
440
NARROW.PIVOT (VERKÄUFE FÜR JAHR IN (1996, 1997) UND QUARTAL IN (FRÜHLUNG, SOMMER, HERBST, WINTER))
450
WIDE.UNPIVOT (VERKÄUFE ÜBER (GEBIET, JAHR))
460

Figur 4



Figur 5



Figur 6