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

(10) **DE 602 14 862 T2** 2007.05.10

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 410 566 B1**
(21) Deutsches Aktenzeichen: **602 14 862.6**
(86) PCT-Aktenzeichen: **PCT/GB02/03095**
(96) Europäisches Aktenzeichen: **02 747 560.7**
(87) PCT-Veröffentlichungs-Nr.: **WO 2003/009531**
(86) PCT-Anmeldetag: **04.07.2002**
(87) Veröffentlichungstag der PCT-Anmeldung: **30.01.2003**
(97) Erstveröffentlichung durch das EPA: **21.04.2004**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **20.09.2006**
(47) Veröffentlichungstag im Patentblatt: **10.05.2007**

(51) Int Cl.[8]: ***H04L 12/26*** (2006.01)
H04L 12/24 (2006.01)

(30) Unionspriorität:
910676 **20.07.2001** **US**

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

(73) Patentinhaber:
Micromuse Ltd, London, GB

(74) Vertreter:
derzeit kein Vertreter bestellt

(72) Erfinder:
KERRISON, Adam Geoffrey, Esher Surrey KT10 9NN, GB; BENNETT, Jonathan, Andrew, London SW13 0AL, GB; STEWART, Kristian Jon, London SW18 2JW, GB; BANYARD, S., Nicholas, Woking GU21 3LY, GB

(54) Bezeichnung: **METHODE FÜR DIE VERBESSERTE VERWALTUNG VON EINER EREIGNISDATENBASIS UND SYSTEM FÜR EREIGNISMELDUNG IN EINEM NETZWERK**

Beschreibung

COPYRIGHT-VERMERK

[0001] Ein Teil der Offenbarung dieser Patentschrift enthält Material, das dem Urhebenechtsschutz unterliegt. Der Urhebenechtsinhaber hat keinerlei Einwände gegen die Faxsimilereproduktion der Patentschrift oder der Patentoffenbarung durch jedermann, wie sie in den Akten oder Unterlagen des Patentund Markenamtes erscheint, behält sich andernfalls jedoch alle wie auch immer gearteten Rechte aus dem Copyright vor.

VERWEIS AUF VERWANDTE ANMELDUNGEN

[0002] Diese Anmeldung ist verwandt der US-Patentschrift 2003014462 A1 mit Datum 16.01.2003, mit dem Titel "METHOD AND SYSTEM FOR EFFICIENT DISTRIBUTION OF NETWORK EVENT DATA".

ALLGEMEINER STAND DER TECHNIK

[0003] Die hier offenbarte Erfindung betrifft im Allgemeinen Netzüberwachungssysteme. Im Besonderen betrifft die vorliegende Erfindung verbesserte Verfahren und Systeme zum effizienten Speichern von Ereignisdaten in einer Datenbank und Verteilen der Daten an verschiedene Benutzer, wobei die Ereignisdaten sich auf Ereignisse beziehen, die auf einem Computernetz auftreten.

[0004] Den ordnungsgemäßen Betrieb von über ein Netz angebotenen Diensten zu erhalten ist gewöhnlich eine wichtige aber schwierige Aufgabe. Verwalter von Diensten sind oftmals aufgefordert, auf eine Störung eines Dienstes zu reagieren, indem sie das Problem, das die Störung verursacht hat, identifizieren und dann Schritte setzen, um das Problem zu korrigieren. Die Kosten der Ausfallzeit von Diensten, die begrenzte Anzahl von Netztechnikern und die wettbewerbsorientierte Natur des modernen Marktes haben Diensteanbieter gezwungen, sich immer mehr auf Software-Tools zu verlassen, um ihre Netze höchstmöglich effizient in Betrieb zu halten und vereinbarte Serviceleistungen an einen wachsenden Kundenbestand zu liefern. Folglich ist es wesentlich geworden, dass diese Software-Tools im Stande sind, ein Netz so effizient wie möglich zu verwalten und zu überwachen.

[0005] Eine Anzahl von Tools ist verfügbar, um Verwalter bei der Erfüllung dieser Aufgaben zu unterstützen. Ein Beispiel ist die NETCOOL® Applikationssuite, verfügbar von Micromuse Inc. San Francisco, Kalifornien, die Netzverwaltern erlaubt, Aktivität in Netzen, wie etwa in drahtgebundenen und drahtlosen Sprachkommunikationsnetzen, Intranets, weiträumigen Netzen oder im Internet zu überwachen. Die NETCOOL® Suite enthält Probes und Monitore, die Netzereignisdaten protokollieren und sammeln, einschließlich Ereignisauftreten im Netz, wie etwa Warnungen, Alarme oder andere Störungen, und die Ereignisdaten in einer Datenbank auf einem Server speichern. Das System meldet dann Netzverwaltern die Ereignisdaten in grafischen und textorientierten Formaten in Übereinstimmung mit besonderen durch die Verwalter getätigten Anforderungen. So können Verwalter gewünschte Netzereignisse auf Echtzeitbasis beobachten und schneller darauf antworten. Die NETCOOL® Software erlaubt Verwaltern, Ereignisdaten entsprechend einer gewünschten Metrik oder Formel zusammengefasst anzufordern, und sie erlaubt Verwaltern weiter, Filter zu wählen, um ihre eigenen Servicesichten und Serviceberichte kundenspezifisch zu gestalten.

[0006] Bekannte Versuche, Ereignismanagement in Netzsystemen zu verbessern, umfassen die Schriften WO 00/39674, US 6255943 und US 6131112. Im Besonderen lehrt WO 00/39674 ein Ereignismanagementsystem, wobei ein Ereignismanager Ereignisdaten empfängt und für einen Ereigniskorrelator bereitstellt, der die Ereignisdaten korreliert, in Übereinstimmung mit Alarmregeln, die in einem Alarmregelspeicher gespeichert sind. Eine Antwortmaschine ist vorgesehen, eine Antwortstrategie auszuführen, auf Basis der Korrelation der Ereignisse durch den Ereigniskorrelator. US 6255943 beschreibt ein Verfahren zum Filtern von verteilten Objekten. Das umfasst Erzeugen von Alarmen von mehreren Netzwerkmanagementservern, Zuordnen von strategiebasierten Filtern zu den Netzwerkmanagementservern und dazugehörigen Netzwerkmanagementapplikationen und Anwenden der zugeordneten strategiebasierten Filter auf die Alarme. Wenn ein Alarm die Filter passiert, wird eine Alarmbenachrichtigung erzeugt und weitergeleitet an die dazugehörige Netzwerkmanagementapplikation. US 6131112 offenbart ein Verfahren zum Teilen von Information zwischen einem ersten und einem zweiten Managementsystem. Das umfasst Empfangen einer Ereignisnachricht durch das erste Managementsystem und Bestimmen, ob die Ereignisnachricht sich auf eine Entität bezieht, die von dem zweiten Managementsystem verwaltet wird. Wenn das der Fall ist, wird die Nachricht formatiert, um kompatibel zu sein mit dem zweiten Managementsystem, und das zweite Managementsystem wird benachrichtigt, dass eine Ereignisnachricht verfügbar ist von dem ersten Managementsystem.

[0007] In einer anspruchsvollen Umgebung gibt es mehrere zig oder gar hundert Clients, die im Wesentlichen dieselben gefilterten oder zusammengefassten Ereignisdaten sichten. Außerdem gibt es in einem großen Netz tausende Geräte, die an einer Anzahl geografisch entfernter Orte überwacht werden, und Ereignisse treten an diesen Geräten mit großer Häufigkeit auf. Folglich sind die Datenbanken, die Ereignisdaten speichern, sehr groß und werden permanent aktualisiert. Neu eingehende Ereignisdaten werden verzögert, bevor sie in der Datenbank gespeichert oder verarbeitet werden, z.B. während eines Zeitraums, in dem die Datenbanken gesperrt sind. Auch wenn solche Verzögerungen nur einen Sekundenbruchteil oder wenige Sekunden ausmachen, kann das die Fähigkeit der Verwalterclients beeinträchtigen, Ereignisdaten zeitgerecht zu empfangen. Diese und ähnliche Probleme verschärfen sich mit zunehmender Größe des Netzes, wodurch die Skalierbarkeit des Netzwerkmanagementsystems limitiert wird.

[0008] Folglich besteht eine Notwendigkeit an Verbesserungen in der Art, wie derartige Netzereignisdatenbanken mit Ereignissen aktualisiert werden, und in der Art, wie sie verwaltet werden, um größere Skalierbarkeit und Effizient zu bieten. Weiter besteht eine Notwendigkeit an verbesserten Methoden zum effizienten Koordinieren des Verarbeitens von Ereignisdaten, die sowohl von lokalen als auch von entfernten Netzen bezogen werden.

KURZDARSTELLUNG DER ERFINDUNG

[0009] Die vorliegende Erfindung sieht verbesserte Verfahren und Systeme zum Verwalten einer Ereignisdatenbank in einem Netzüberwachungssystem vor. Die Verbesserungen steigern die Effizienz in der Art und Weise, wie Ereignisdaten durch die Datenbank gehandhabt werden, steigern die Leistung und Geschwindigkeit der Ereignisdatenbank und verbessern die Skalierbarkeit der Ereignisdatenbank, um zu erlauben, dass sie ein größeres Netz bedient. Die Verbesserungen umfassen Verfahren, Benutzern zu erlauben, Trigger zu setzen zum automatischen Vorverarbeiten von Ereignisdaten, die von überwachten Standorten empfangen werden, ein verteiltes System lokaler Master- und Replikdatenbanken und Methoden zum Koordinieren der Ereignisdaten über diese, sowie ein Benachrichtigungssubsystem zum Bedienen wiederholter Clientanforderungen nach unverarbeiteten und verarbeiteten Ereignisdaten. Diese Verbesserungen wirken zusammen, um eine verbesserte Leistung, wie hier beschrieben, zu erreichen.

[0010] Unter Verwendung des ersten Aspekts der Verbesserungen wird Benutzern die Möglichkeit geboten, Trigger in die Ereignisdatenbank einzufügen. Die Trigger prüfen automatisch auf bestimmte Ereignisbedingungen, Zeiten oder Datenbankereignisse, einschließlich Systemereignisse oder benutzerdefinierter Ereignisse, einschließlich eines Namens und eines gewählten Parameters, und initiieren eine programmierte Aktion bei Erkennen einer derartigen Bedingung, Zeit oder eines Ereignisses. Ein derartiger Trigger ist ein vorverarbeitender Trigger, der Ereignisdaten prüft, bevor sie in die Ereignisdatenbank eingefügt werden, während sie noch gespeichert sind, z.B. in einem Ereignispuffer oder einer Warteschlange. Eine mögliche Verwendung eines derartigen Triggers besteht darin, Duplikate von Ereignisdaten in der Ereignisdatenbank zu verhindern oder zu reduzieren. Das heißt, dieselben oder duplizierte Ereignisdaten können von einem oder mehreren Monitoren gemeldet werden, und die Vorverarbeitung der Ereignisdaten erkennt das Auftreten von duplizierten Daten durch Vergleich mit Ereignisdaten, die schon in der Ereignisdatenbank gespeichert sind, oder anderen Ereignisdaten, die ebenfalls empfangen worden sind zum, jedoch vor, Eintrag in die Ereignisdatenbank. Duplikate zu verhindern hält die Ereignisdatenbank schlank und begrenzt auch die Zeit, in der die Ereignisdatenbank in einem gesperrten Zustand ist, wie das während Lese/Schreib-Operationen notwendig sein kann.

[0011] Außerdem stellt die Erfindung Ereignisdaten sowohl von lokalen als auch von entfernten Überwachungsorten effizient bereit, wobei ein System lokaler Repliken und ihrer Vereinigung verwendet wird. Jeder Überwachungsort kann seine eigenen Ereignisdaten lokaler überwachter Orte führen, sowie Repliken von Ereignisdaten, die an anderen, entfernten Überwachungsorten gespeichert sind. Unter Verwendung einer Vereinigung oder kombinierenden Operation können die lokalen und entfernten Daten kombiniert werden, um eine vereinheitlichte Ereignisdatenzusammenfassung zur Verwendung durch den Client bereitzustellen. Die Überwachungsorte aktualisieren einander, wenn ihre Ereignisdaten sich ändern.

[0012] Die Kombination der vorverarbeitenden Trigger und die Verwendung von Repliken und Vereinigung bietet wesentlich verbesserte Skalierbarkeit des Netzüberwachungssystems. Da Daten in lokalen Ereignisdatenbanken ausgebreitet werden zu entsprechenden Repliken an entfernten Orten, ist es besonders vorteilhaft, ein Aktualisieren jeder lokalen Datenbanktabelle zu verzögern oder zu vermeiden, außer es ist gewünscht oder notwendig. Die Verwendung von Triggern erleichtert es, diese Auswahl zu erreichen, bevor jede Datenbank überfordert wird.

[0013] Die verbesserte Skalierbarkeit und Leistung der Ereignisdatenbank wird weiter durch ein Verfahren und eine Softwarekomponente zur Publish/Subscribe-Benachrichtigung erreicht. Die Benachrichtigungskomponente registriert anhaltende Clientanforderungen nach gefilterten unverarbeiteten Daten oder Zusammenfassungsdaten, ordnet ähnliche Anforderungen einander zu und liefert alle entsprechenden Anforderungen ungefähr zur gleichen Zeit an alle Benutzer, welche dieselben oder ähnliche Daten anfordern. Das erlaubt Benutzern, effizient mit neuen Ereignisdaten aktualisiert zu werden, ohne die Ereignisdatenbank exzessiv zu überfordern.

[0014] Eine gemeinsame Grundidee in der Verwendung von Triggern, besonders von vorverarbeitenden Triggern, einer Union/Replik-Architektur und eines Publish/Subscribe-Benachrichtigungsystems ist, dass es sich bei allen um ereignisbasierte Verfahrensweisen handelt. Trigger werden durch das Auftreten von Ereignissen, seien sie zeitlich oder anderer Art, aktiviert und initiieren Aktionen an den Ereignissen selbst, z.B. Einfügeverweigerung oder sofortige Kommunikation an einen Client. Die Union/Replik-Architektur bewegt sich weg von einem traditionellen Datenbankmodell, das sich darauf richtet, wo Ereignisse auftreten, und welche Orte sie betreffen könnten. Das Benachrichtigungsmodell richtet sich auf gesteigerte Effizienz im Liefern spezifischer Ereignisdaten oder Ereignisdatenmetriken an Clients, die sie anfordern. Verschieben der Ausrichtung eines Netzwerkmanagementsystems auf die Ereignisse, die im Netz auftreten, führt zu den hier beschriebenen Verbesserungen in Geschwindigkeit und Effizienz.

KURZBESCHREIBUNG DER ZEICHNUNGEN

[0015] Die Erfindung ist in den Figuren der beiliegenden Zeichnungen illustriert, die Beispielcharakter haben und nicht einschränken sollen, und in denen sich gleiche Bezugszeichen auf gleiche oder einander entsprechende Elemente beziehen sollen, dabei zeigt:

[0016] Fig. 1 ein Blockdiagramm, das funktionale Komponenten eines verbesserten Netzüberwachungssystems darstellt, gemäß einer Ausführungsform der vorliegenden Erfindung;

[0017] Fig. 2 eine Datenflussdarstellung, welche die Vorverarbeitung von Ereignisdaten illustriert, gemäß einer Ausführungsform der vorliegenden Erfindung;

[0018] Fig. 3 ein Flussdiagramm, das einen Prozess darstellt, in dem ein Trigger zum Vorverarbeiten von Ereignisdaten verwendet wird, gemäß einer Ausführungsform der vorliegenden Erfindung;

[0019] Fig. 4 eine Datenflussdarstellung, welche einen Prozess zum Verwalten von lokalen Ereignisdatenbanken und Repliken illustriert, gemäß einer Ausführungsform der vorliegenden Erfindung;

[0020] Fig. 5 einen Clientdisplay mit einer geordneten Sicht, gemäß einer Ausführungsform der vorliegenden Erfindung;

[0021] Fig. 6 einen Clientdisplay mit einer Landkarten-/geografischen Sicht, gemäß einer Ausführungsform der vorliegenden Erfindung; und

[0022] Fig. 7 einen Clientdisplay mit einer Listensicht, gemäß einer Ausführungsform der vorliegenden Erfindung;

DETAILLIERTE BESCHREIBUNG DER BEVORZUGTEN AUSFÜHRUNGSFORMEN

[0023] Gemäß der Erfindung werden hier Verfahren und Systeme beschrieben, mit Bezug auf die Figuren, zum Bereitstellen von Vorverarbeitung von Ereignisdaten, sowie von effizientem Liefern der Ereignisdaten an eine Anzahl von Clients. Im Besonderen richtet sich diese Beschreibung auf ein Netzüberwachungssystem, in dem Daten erfasst werden, die sich auf Ereignisse beziehen, wie etwa Störungen oder Alarme, die in einem Computernetz auftreten und verteilt werden an eine Anzahl von Verwalterclients, die für die Netzüberwachung und das Verhindern oder Korrigieren derartiger Störungen verantwortlich sind.

[0024] Fig. 1 zeigt ein Netzüberwachungssystem gemäß der vorliegenden Erfindung. Das System umfasst einen Objektserver **26**, welcher Ereignisdaten von einer Anzahl von Überwachungseinrichtungen, die Probes **2** und Monitore **4** umfassen, empfängt, die Ereignisdaten in einer oder mehreren Ereignisdatenbanken **28** speichert und die Ereignisdaten für eine Anzahl von Clients **8** bereitstellt, welche Anforderungen nach den Daten ausgeben. In einer Ausführungsform sind die Ereignisdatenbanken **28** im Objektserver **26** relationale Ereignis-

datenbanken **28**, wie etwa die Objektserverdatenbank, die als Teil des NETCOOL®/Omnibussystems verfügbar ist, das von Micromuse, Inc., San Francisco, Kalifornien verfügbar ist. In alternativer Ausgestaltung kann die Ereignisdatenbank **28** jeder andere geeignete Datenspeichertyp sein, wie etwa eine objektorientierte Datenbank, eine zweidimensionale Datenbank usw. Die Ereignisdatenbank **28** ist eine speicherresidente Datenbank, die periodisch für den Fall einer Störung als Datei abgelegt wird. Ereignisse kommen von Probes **2** und Monitoren **4** in der Form von SQL Einfügungen herein. Clients **8** greifen auf die Datenbank ebenfalls unter Verwendung von SQL zu. Wie unten erklärt, ist der Server leicht zu konfigurieren und weist gesteigerte Leistungsfähigkeit, Funktionalität und Flexibilität auf.

[0025] Die Probes **2** sind Codeabschnitte, die Ereignisse von Netzwerlananagementdatenquellen **6**, APIs, Datenbanken, Netzwerkgeräten **5**, Protokolldateien und von anderen Utilities sammeln. Monitore **4** sind Softwareapplikationen, die Netzbenutzer simulieren, zum Bestimmen von Antwortzeiten und Verfügbarkeit der Dienste **7** wie in einem Netz. Andere Überwachungseinrichtungen können verwendet werden, um Ereignisse zu sammeln und zu melden, die im Netz oder an entsprechenden Geräten oder Diensten auftreten.

[0026] Das Netzwerkmanagementsystem überwacht und meldet Aktivität in einem Computernetz, Telekommunikationsnetz oder in einem anderen Netztyp. In diesem Zusammenhang sind Clients **8** typischerweise Verwalter, die Ereignisdaten anfordern, die sie regelmäßig zum Überwachen benötigen. Clients können wählen, die gesamte Ereignisaktivität im Netz einzusehen. Typischer bei größeren Netzen wollen Clients nur Ereignisdaten sehen, die in besonderen Teilen des Netzes auftreten, für die sie verantwortlich sind, oder die ihren Netzabschnitt betreffen können. Außerdem können Clients nur Zusammenfassungen ihres relevanten Teils der Ereignisdaten sehen wollen, wie etwa Ereigniszahlen, Summen, Durchschnittswerte, Minimalwerte, Maximalwerte oder weitere Verteilungen von Ereignissen. Clients geben die verschiedenen Anforderungen in eine Ereignisliste **34** ein, wobei jede Anforderung eine besondere Sicht der Daten repräsentiert und hier gelegentlich als eine solche bezeichnet wird.

[0027] Ereignisdaten werden in der Ereignisdatenbank **28** einer Ausgestaltung in einer Anzahl von Reihen und Spalten gespeichert, wobei jede Reihe ein Ereignis repräsentiert, und die Spalten Datenfelder speichern, die sich auf das Ereignis beziehen, z.B. Ort, Typ, Zeit, Ernsthaftigkeit usw. Nach vorliegender Verwendung ist eine Sicht also im Allgemeinen ein Mechanismus, um Spalten aus der Datenbank auszuwählen, und kann optional auch einen Filter umfassen. Ein Filter ist im Allgemeinen ein Mechanismus zum Ausschließen von Datenreihen in der Datenbank auf der Basis von Spaltenwerten. Sichten können daher auf Filtern basieren. Filter können auch auf anderen Filtern und anderen Sichten basieren. Eine metrische Sicht ist im Allgemeinen ein Sichttyp, der eher zusammengefasste Information über die Anzahl von Reihen in einer Sicht bereitstellt als über die tatsächlichen Daten, und erfordert gewöhnlich eine gewisse rechnerische Verarbeitung an der Anzahl von Reihen.

[0028] Diese Clientanforderungen oder Sichten sind anhaltend und werden in Übereinstimmung mit einem Publish/Subscribe-Modell geliefert. Das heißt, da Netzereignisse regelmäßig auftreten, ändern sich die Daten in der Ereignisdatenbank **28** häufig, und Clients müssen prompt über die Aktualisierungen in Übereinstimmung mit ihren spezifizierten Anforderungen informiert werden, um die Daten geeignet verwenden zu können. Der Objektserver **26** verarbeitet die anhaltenden Anforderungen in einer gesetzten Frequenz, z.B. alle fünf oder zehn Sekunden, und liefert die Resultate an die Clients in Form eines Stroms oder von Ereignisdaten, die neu oder seit der letzten Verarbeitung der Anforderungen aktualisiert sind. Die Defaultfrequenz oder Initialfrequenz zum Verarbeiten von anhaltenden Anforderungen kann vorab als jede gewünschte Frequenz gesetzt werden, in jeder gewünschten Zeiteinheit, z.B. Sekunden oder Sekundenabschnitte, Minuten, Stunden usw., und in jedem gewünschten Umfang.

[0029] Was die am Objektserver **26** verwendeten Kommunikationen betrifft, umfassen Merkmalen der vorliegenden Erfindung Ereignislistenoptimierung, Repliken **42**, und Unionen **44**, sowie Cluster. Was Ereignislistenoptimierung betrifft, so hat eine typische Installation viele Benutzer mit derselben Ereignisliste. Das ist eine wesentliche Ursache von Belastung am Server. Ansätze umfassen einmaliges Auswerten einer Sicht, dann Publizieren der Ergebnisse auf dem Bus und Verwenden von Drosselung zum Verhindern von Überlastung. Eine Replik umfasst eine gecachte Kopie einer entfernten Tabelle. Einfügungen, Aktualisierungen und Löschungen werden zur Masterkopie einer Replik weitergeleitet. Eine Union ist ein Set von Tabellen und Repliken, die als eine einzige Tabelle dargestellt werden. Repliken und Unionen werden im Folgenden näher erläutert. Cluster sind in der oben genannten in gemeinsamem Besitz stehenden US-Patentschrift 2003014462 A1 beschrieben.

[0030] Gemäß einem Aspekt der Erfindung ist ein Benachrichtigungsprogramm oder Notifier **30** vorgesehen, der die Clientanforderungen nach Daten vom Objektserver **26** verwaltet, um die Antworten auf die Clientanfor-

derungen effizient zu verteilen. Das Benachrichtigungsprogramm **30** kann Teil des Objektservers **26** sein, wie dargestellt, oder kann eine separate, für sich allein stehende Komponente des Systems sein. Gemäß den in der oben erwähnten in gemeinsamem Besitz stehenden US-Patentschrift 2003014462 A1 detaillierter beschriebenen Verfahren verwaltet das Benachrichtigungsprogramm **30** die verschiedenen Clientanforderungen in einer Sichtenliste oder Sichtentabelle **32**, die eine Anzahl von Anforderungssets aufweist. Jedes Anforderungsset betrifft einen spezifischen Sichttyp oder Datenfilter und kann eine Anzahl von Metriken oder Formeln enthalten, die Daten im Objektserver **26** zusammenfassen, und die angefordert werden, um durch oder für Clients **8** verarbeitet zu werden.

[0031] Wenn der Notifier **30** eine Registrierungsanforderung von einem Client **8** empfängt, scannt er seine Liste bestehender Registrierungen. Wenn, wie in diesem Beispiel, eine identische Registrierung bereits existiert, wird die zweite Registrierung der ersten zugeordnet. Die erste Registrierung besonderer Zusammenfassungsdaten kann als "primäre" Registrierung oder Anforderung bezeichnet werden, während nachfolgende Registrierungen identischer Zusammenfassungsdaten als "sekundäre" Registrierungen oder Anforderungen bezeichnet werden können. Der Notifier **30** scannt periodisch seine Liste primärer Registrierungen, berechnet für jede die Zusammenfassungsdaten und sendet die Resultate an alle Clients, die Interesse an diesen Daten registriert haben. Wenn ein Client **18** also eine metrische Sicht in seiner Ereignisliste **34** wählt, registriert der Notifier **30** Interesse an dieser metrischen Sicht mit der Sichtentabelle **32** im Objektserver **26**. Wenn ein weiterer Client wählt, dieselbe metrische Sicht einzusehen, registriert der Notifier **30** auch das Interesse dieses anderen Clients an den Zusammenfassungsdaten in der Sichtentabelle **32**.

[0032] Folglich wird die Auswertung der Zusammenfassungsdaten durch dieses Benachrichtigungsprogramm **30** und Sichtenlistenlibrary **32** optimiert. Unter der spezifischen Annahme, dass jeder Client Sichten derselben M Metriken anfordert, liegt die Arbeit, die vom Objektserver **26** zu bewältigen ist, eher in einer Größenordnung von M als M*(Anzahl der Clients).

[0033] Weiter ist, gemäß der Erfindung, eine Automatisierungsmaschine **20** als Teil des Objektservers **26** vorgesehen. Die Automatisierungsmaschine **20** umfasst die Verwendung von "Triggern", um einem Verwalter zu erlauben, Ereignisse, Bedingungen und Aktionen miteinander zu verbinden. Der Objektserver **26** stellt das Automatisierungsmodul bereit, mit Ereignissen, Triggern und Aktionen, die völlig allgemein sind, zahlreiche Leistungssteigerungsanforderungen zu unterstützen, wie auch Fehlersuche, wobei einzelne Trigger in Debugmodus gesetzt werden können, und Debugeinträge in eine Protokolldatei geschrieben werden.

[0034] Ein großer Teil der Leistungsfähigkeit des Objektservers **26** kommt durch Automatisierung zu Stande. Automatisierung erlaubt es dem System, auf Ereignisse zu antworten, das sind Geschehen von Interesse, wie etwa die Modifikation von Daten in einer Tabelle, das Ausführen eines Befehles oder der Ablauf eines Zeitintervalls. Wenn ein Ereignis auftritt, wird eine Anfrage ausgeführt, dann wird eine Bedingung ausgewertet, die, wenn sie wahr ist, die Ausführung einer Aktion verursacht: z.B. eine Sequenz von SQL Anweisungen, ein externes Skript oder beides. Darüber hinaus hat jeder Trigger einen Kopplungsmodus, der anzeigt, wann die Aktion ausgeführt werden sollte, das heißt jetzt oder zu einem späteren Zeitpunkt. Trigger können kreiert, geändert oder entfernt werden. Die SQL Befehlssyntax zum Implementieren von Triggern in einer Ausführungsform der Erfindung wird im Anhang zusammengefasst, der Teil dieser Schrift ist.

[0035] Trigger erlauben, dass Aktionen ausgeführt werden, wenn ein Ereignis von Interesse auftritt. Das triggernde Ereignis kann beispielsweise ein Primitivereignis sein, ein Datenbankereignis oder ein zeitliches Ereignis. Wenn das Ereignis auftritt, wird eine optionale Evaluierung ausgeführt, dann wird eine Bedingung ausgewertet, und, wenn sie wahr ist, eine Aktion ausgeführt. Ein Primitivereignis ist ein Geschehen von Interesse, dem ein eindeutiger Name zugeordnet ist. Bei Auftreten tragen alle Ereignisse ein Set von Argumenten, das null oder mehr <name, type> Paare enthält. Die Klassen von Primitivereignissen umfassen eine Benutzerklasse für ein Ereignis, das von einem Benutzer kreiert ist, und eine Systemklasse für ein Auftreten im Objektserver, das von Interesse ist, z.B. das Einloggen eines Benutzers, der Ablauf einer Berechtigung usw.

[0036] Was Benutzerereignisse angeht, können diese benutzerdefinierte Ereignisse umfassen. Darüber hinaus kann ein Benutzerereignis in einer Befehlszeile kreiert werden, z.B: create event bob (a INT, b REAL). Ein Benutzerereignis kann in der Befehlszeile ausgelöst werden, z.B: raise event bob 1, 1.2.

[0037] Zeitliche Trigger signalisieren ein Ereignis bei einer bestimmten Frequenz von einer spezifizierten Startzeit bis zu einer spezifizierten Endzeit.

[0038] Gemäß einem Aspekt der Erfindung können Trigger in die Datenbank eingefügt werden zum Handha-

ben der Verarbeitung von Ereignisdaten, bevor sie der Datenbank hinzugefügt werden. Der Objektserver empfängt unverarbeitete Ereignisdaten von überwachten Orten im Netz, z.B. über Überwachungseinrichtungen **2**, **4**. Die unverarbeiteten Ereignisdaten werden in einem Puffer **12** gepuffert und von der Automatisierungsmaschine **20** vorverarbeitet, bevor sie in einer Datenbank gespeichert werden. Fig. 2 stellt diesen Prozess schematisch dar, wobei Trigger **22** Ereignisdaten in Puffer **12** liest, bevor sie in Ereignisdatenbanktabelle **40** eingefügt werden. Vorteilhafterweise können die Ereignisdaten **12** auch dann verarbeitet werden, wenn die Datenbank in einem gesperrten Lese/Schreib-Zustand ist, z.B. wenn die Datenbank aktualisiert wird und keine Lese- oder Schreiboperationen von oder zu externen Geräten durchführen kann. Das vermeidet Verzögerungen im Verarbeiten von Ereignisdaten und Kommunizieren entsprechender Mitteilungen an die Clients über einen Notifier.

[0039] Ein EECA (Event, Evaluate, Condition, Action) Modell wird für Trigger verwendet: Ereignis, Auswerten, Bedingung, Aktion. Wenn ein Ereignis auftritt, führe Auswerten aus, dann teste die Bedingung, und wenn wahr, führe Aktion aus. Übergangstabellen können verwendet werden, um zwischen den Phasen eines Triggers Resultate zu kommunizieren. Prozedurorientierte SQL Syntax kann in Aktionen verwendet werden. Es folgen einige exemplarische Triggertypen, die verwendet werden können.

[0040] Ein "up-down-correlation" zeitlicher Trigger kann wie folgt beschrieben werden:

NAME	up-down correlation
EVENT	every 10 seconds
EVALUATE	select Node from alerts.status where ... bind as tt
CONDITION	when %row_count > 0
ACTION	
	for each row r in tt update alerts.status set ... where Node = r.Node

[0041] Ein "PageOnRouterDown" zeitlicher Trigger kann wie folgt beschrieben werden:

NAME	PageOnRouterDown
EVENT	every 10 seconds
EVALUATE	select * from alerts.status where ... bind as tt
CONDITION	true
ACTION	
	if %trigger.row_count > 0 AND (%trigger.positive_row_count mod 5) = 0 system(/usr/bin/page_message 'some routers are down') end if if %trigger.row_count = 0 AND (%triggers.zero_row_count mod 5) = 0 system(/usr/bin/page_message 'no routers are down') end if

[0042] Ein Beispiel eines Datenbanktriggers, der neue Ereignisdaten vorverarbeitet, um Duplikate von Ereignisdaten in der Ereignisdatenbank zu verhindern, wird wie folgt beschrieben:

NAME	deduplication
EVENT	before reinsert on alerts.status
EVALUATE	<nothing>
CONDITION	true
ACTION	
	set old.Tally = old.Tally + 1, set old.Summary = new.Summary set old.LastOccunence = new.LastOccurrence

[0043] Ein Beispiel eines Ereignistriggers kann wie folgt beschrieben werden:

NAME	ConnectionWatch
EVENT	on event connect
EVALUATE	<nothing>
CONDITION	true
ACTION	
	insert into alerts.status ... %event.user, %event.time

[0044] Mit Bezug auf Fig. 3 ist ein Prozess zur Verwendung von Triggern zum Vorverarbeiten von Ereignisdaten, wie etwa in einem Deduplikationstrigger, wie folgt dargestellt. Ein Benutzer erzeugt den Trigger auf Basis eines gewünschten Ereignisses und fügt ihn in die Datenbank ein, Schritt **50**. Wenn neue Ereignisdaten empfangen werden, Schritt **52**, werden sie vom Trigger überprüft, während sie noch im Ereignispuffer gespeichert sind, und bevor sie der Datenbank hinzugefügt werden, Schritt **54**. Wenn das vom Trigger definierte Ereignis aufgetreten ist, Schritt **56**, z.B. vom Ereignis in den neuen Ereignisdaten repräsentiert wird, prüft der Trigger, ob eine (der) benutzerdefinierte(n) Bedingung(en) erfüllt ist/sind, Schritt **58**. Wenn das so ist, werden die neuen Ereignisdaten in Übereinstimmung mit der Aktion verarbeitet, die im Trigger spezifiziert ist, Schritt **60**. Zu diesem Zweck erzeugt der Trigger eine Übergangstabelle, das heißt eine anonyme Tabelle, die von der Aktion im Trigger verwendet wird, und verwendet eine oder mehrere Übergangsvariablen, das heißt Variablen, die von einem Datenbanktrigger erzeugt und von einer Aktion verwendet werden. Wenn die Aktion im Trigger Löschen der neuen Ereignisdaten oder auf andere Weise Vermeiden oder Verzögern des Einfügens der neuen Ereignisdaten in die Ereignisdatenbank, Schritt **62**, umfasst, werden die neuen Ereignisdaten nicht hinzugefügt. Andernfalls, oder wenn die Triggeraktion nicht eingeleitet wird, werden die Ereignisdaten der Ereignisdatenbank hinzugefügt, Schritt **64**.

[0045] Mit erneutem Bezug auf Fig. 1, gemäß einem weiteren Aspekt der vorliegenden Erfindung, wird die Ereignisdatenbank **28** verteilt in lokale Ereignisdatenbanken **40**, die Ereignisdaten speichern, die von einem gegebenen Ort im Netz erzeugt sind, und eine oder mehrere Replik-Ereignisdatenbanken **42**, die Ereignisdaten speichern, die gesammelt sind an und repliziert sind von einem oder mehreren entfernten Netzorten. Beim Liefern auf Clientanforderung wird eine Union **44** erzeugt durch Kombinieren der lokalen und der entfernten Datenbanken **40**, **42** unter Verwendung allgemein bekannter Vereinigungsmethoden.

[0046] Mit Bezug auf Fig. 4 wird beispielsweise eine verteilte Implementierung gezeigt mit einem Objektserver in London **26b**, einem in Hongkong **26c** und einem weiteren in New York **26a**. Jeder Objektserver **26a**, **26b**, **26c** ist so konfiguriert, dass er jeweils eine einzige Tabelle **40a**, **40b**, **40c** aufweist, die lokal erzeugte Alarme enthält, und eine Replik der vom anderen Objektserver geführten Tabelle. So enthält beispielsweise die lokale Datenbanktabelle **40a** New York lokal erzeugte Alarme oder andere Ereignisse in einer Tabelle, benannt alerts.NY, ferner enthält sie Repliken von entfernt erzeugten Alarmen oder anderen Ereignissen in alerts.LN **42a** und alerts.HK **43a**. Zusätzlich enthält sie eine Union **44a**, benannt alerts.status, die alerts.HK, alerts.LN, alerts.NY logisch enthält. Eine an einer lokalen Tabelle ausgeführte Modifikation wird von dieser Tabelle direkt angewandt. Eine Modifikation an einer Replik erfordert normalerweise, dass die Modifikation an ihren Anker geschrieben wird, beispielsweise führt eine Aktualisierung jeder Reihe in alerts.status in Hong Kong dazu, dass eine Nachricht nach New York gesendet wird, die anfordert, dass die dort enthaltenen Reihen von alerts.NY aktualisiert werden.

[0047] Allerdings ist es einem lokalen Überwachungsort möglich, die Stelle des Ankers einer Replik zu übernehmen, sodass darauffolgende Modifikationen daran kein "Hinüberschreiben" zu ihrem Anker erfordern, das heißt, keine Anweisungen von dem dazugehörigen entfernten Überwachungsort empfangen. Das erlaubt, dass "Follow-the-Sun"-Konfigurationen aufgebaut werden, in welchen jede lokale Überwachungsstation die Aktualisierung von lokalen Tabellen und Repliktabellen während ihres primären Betriebszeitrahmens steuert und dann die Steuerung übergibt an einen anderen Ort in einem anderen Zeitrahmen, der dann für seinen gegebenen Zeitrahmen des Geschäftsbetriebs die Steuerung übernimmt.

[0048] So stellt eine Replik gemäß der Erfindung eine Kopie einer Tabelle dar, die an einem entfernten Objektserver gehalten wird, und sie wird inkrementell aktualisiert, sodass sie den Zustand der Tabelle am Ankerort widerspiegelt. Versuche, Daten zu modifizieren, die in einer Replik gehalten werden, führen zu einem Befehl, der an die Masterkopie der Tabelle gesendet wird, welche die Modifikation anfordert. Ein Ort, der die Replik hält, kann, wenn erforderlich, der Anker werden. Die Attribute einer Replik umfassen ihren eindeutigen Namen in jeder Datenbank und ihre Speicherklasse (anhaltend oder temporär). Versuche, Daten zu modifizieren, die in einer Union gehalten werden, führen zu einem Befehl, der auf jede ihrer Komponenten in einer implementationsdefinierten Abfolge angewandt wird. Die Attribute einer Union umfassen ihren eindeutigen Namen in jeder Datenbank und ihre Speicherklasse (anhaltend oder temporär). Unionen werden durch Spezifizierung der

Namen der Tabellen und Repliken kreiert und können durch Hinzufügen oder Entfernen von Tabellen oder Repliken verändert werden.

[0049] Die hier beschriebenen Merkmale der vorliegenden Erfindung unterstützen ein leistungsfähiges, skalierbares und schnelles System zum Liefern von Ereignisdaten an Clients unter Verwendung zahlreicher verschiedener Clientsichten. Drei Beispielsichten sind in Fig. 5 bis Fig. 7 dargestellt. Fig. 5 zeigt einen Clientdisplay mit einer geordneten Sicht, gemäß einer Ausführungsform der vorliegenden Erfindung. Der Display wird am Ort des Client erzeugt als Antwort auf die Ereignisdaten, die ihm vom Server kommuniziert werden. Sowohl pure als auch metrische Sicht sind dargestellt (in den Bildschirmansichten als Monitore bezeichnet, was sich auf clienterzeugte Monitore bezieht und nicht auf spezifische Überwachungseinrichtungen wie Monitore oder Probes, die Daten aus spezifischen Netzen, Diensten oder Geräten sammeln), und der Benutzer kann die Sichten mit Drag und Drop organisieren. Links erscheinen als Sichten. Darüber hinaus kann ein Hintergrundbild/-muster dazugefügt werden.

[0050] Fig. 6 zeigt einen Clientdisplay mit einer Landkarten-/geographischen Sicht, gemäß einer Ausführungsform der vorliegenden Erfindung. Der Display kann auch am Ort des Client erzeugt sein als Antwort auf die Ereignisdaten, die ihm vom Server kommuniziert werden und Ereignisinformation geografisch angeordnet bereitstellen. Benutzer können Sichten durch Drag und Drop positionieren, Größenanpassungen an Sichten durch Ziehen der Ränder vornehmen, einen Link durch Auswählen von zu verlinkenden Monitoren hinzufügen oder eine Hintergrundlandkarte hinzufügen. Neue Monitore, die in geordneter Sicht kreiert werden, erscheinen in unverankerter Palette.

[0051] Fig. 7 illustriert einen Clientdisplay mit einer Listensicht gemäß einer Ausführungsform der vorliegenden Erfindung. Diese Sicht stellt Ereignisdaten auf Basis des Monitors bereit. Im Besonderen wird Monitorinformation als detaillierte Liste dargestellt. Jede Spalte kann durch Auswählen des Spaltenheaders sortiert werden.

[0052] Im Allgemeinen können die Ereignisdaten, die an den Ort des Clients kommuniziert werden, auf vielfältige Art dargestellt werden, wie es den Bedürfnissen des Client entspricht.

[0053] Folglich zeigt sich, dass die vorliegende Erfindung verbesserte und effizientere Methoden bietet zum Reduzieren des Arbeitsaufwands, der von einer Datenbank in einem Computernetz bewältigt werden muss, um Ereigniszusammenfassungsdaten an eine große Zahl von Verwalterclients zu verteilen. Darüber hinaus vermeidet oder reduziert die Erfindung Verzögerungen, die Ereignisdaten an einer Datenbank begegnen, z.B. aufgrund von Verzögerungen beim Datenbankzugriff. Ferner werden sowohl aus lokalen als auch aus entfernten Netzen bezogene Ereignisdaten unter Verwendung von Replik- und Unionsprozessen effizient koordiniert. Jeder Überwachungsort in dem Netz umfasst sowohl lokal erzeugte Ereignisse als auch eine Kopie entfernt erzeugter Ereignisse, die von einem oder von mehreren entfernten Überwachungsorten bereitgestellt und geführt werden.

[0054] Während die Erfindung im Zusammenhang mit bevorzugten Ausgestaltungen beschrieben und illustriert ist, können zahlreiche Variationen und Modifikationen, wie der Fachmann erkennen wird, vorgenommen werden, ohne dass der Rahmen der Erfindung dadurch verlassen wäre, und so soll die Erfindung nicht auf die präzisen Details der oben ausgeführten Verfahrensweise oder Konstruktion beschränkt bleiben, denn derartige Variationen und Modifikationen sollen im Rahmen der Erfindung mit einbezogen sein.

SQL-BEFEHLSSYNTAXZUSAMMENFASSUNG

triggers

```
CREATE [OR REPLACE] TRIGGER <name>                    // temporal triggers
                [DEBUGGING <bool>]
                [ENABLED <bool>]
                PRIORITY <int>                        // 1=min, 20=max
                [COMMENT] <text>
                [FROM <abs_time>] [UNTIL <abs_time>] EVERY <rel_time>
                [EVALUATE <select> BIND AS <name>]
                [WHEN <condition>]
                EXECUTE IMMEDIATE | DEFERRED | DETACHED
                DECLARE
                        <decls>
                BEGIN
                        <action>
                END

CREATE [OR REPLACE] TRIGGER <name>                    // database triggers
                [DEBUGGING <bool>]
                [ENABLED <bool>]
                PRIORITY <int>                        // 1=min, 20=max
                [COMMENT <text>]
BEFORE | AFTER
INSERT | UPDATE | DELETE | REINSERT
ON <table>
                FOR EACH {ROW | STATEMENT}
                [EVALUATE <select> BIND AS <name>] // for STATEMENT
triggers
                [WHEN <condition>]
                EXECUTE IMMEDIATE | DEFFERRED | DETACHED
                DECLARE                               // pre-triggers must be immediate
                        <decls>
                BEGIN
                        <action>
                END
```

```
CREATE [OR REPLACE] TRIGGER <name>                    // event triggers
        [DEBUGGING <bool>]
        [ENABLED <bool>]
        PRIORITY <int>                                // 1=min, 20=max
        [COMMENT <text>]
        ON EVENT <name>
        [EVALUATE <select> BIND AS <name>]
        [WHEN <condition>]
EXECUTE IMMEDIATE | DEFERRED | DETACHED
        DECLARE
                <decls>
        BEGIN
                <action>
        END

ALTER TRIGGER <name>
SET DEBUG <boolval>
SET ENABLED <boolval>
SET PRIORITY <int>

DROP TRIGGER <name>
```

Triggerattribute

These are read-only scalar values accessible in WHEN and action blocks.

%trigger.row_count	number of rows matched in evaluate clause
%trigger.last_condition	value of condition on last execution
%trigger.num_executions	number of times trigger has been run
%trigger.num_fires	number of executions where condition was true
%trigger.num_zero_row_count	number of consecutive fires with zero matches in eval
%trigger.num_positive_row_count	number of consecutive fires with >0 matches in eval

Patentansprüche

1. Verfahren zum Bereitstellen eines verbesserten Netzüberwachungssystems, wobei das Netzüberwachungssystem eine Ereignisdatenbank (**28**) zum Speichern von Ereignisdaten (**12**) umfasst, die Ereignisse darstellen, die auf dem Netz auftreten, wobei die Ereignisdaten (**12**) durch eine Vielzahl von Überwachungseinrichtungen (**2**, **4**) gesammelt werden, die an einer Vielzahl von unterschiedlichen, entfernten Orten (**26**) auf dem Netz angeordnet sind, wobei das Verfahren die folgenden Schritte umfasst:
Zulassen, dass Benutzer (**8**) einen oder mehrere Trigger (**22**) in die Ereignisdatenbank (**28**) einfügen, wobei die Trigger (**22**) automatisch eine programmierte Antwort bei der Erfassung einer Bedingung, die durch gesam-

melte Ereignisdaten (**12**) erfüllt wird, vor der Einfügung der gesammelten Ereignisdaten (**12**) in die Ereignisdatenbank (**28**) hinein initiieren;
gekennzeichnet durch ein Verteilen der Ereignisdatenbank (**28**) an eine Vielzahl von entfernten Netzorten (**26**), wobei jeder entfernte Netzort (**26**) eine lokale Tabelle (**40**), die Ereignisdaten (**12**) enthält, die an dem entfernten Ort (**26**) erzeugt werden, und eine oder mehrere Repliktabellen (**42**), die Ereignisdaten (**12**) enthalten, die an anderen entfernten Orten (**26**) erzeugt werden, speichert, und wobei eine Zusammenfassung (**44**) der lokalen (**40**) und Replik (**42**) Tabellen erzeugt wird, um eine kombinierte Datenbank (**28**) an dem entfernten Ort (**26**) zu bilden; und
Verwenden von Triggern (**22**) und lokalen (**40**) und Replik- (**42**) Tabellen-Zusammenfassungen während einer Lieferung von Ereignisdaten (**12**) an Benutzer (**8**) des Netzüberwachungssystems.

2. Verfahren nach Anspruch 1, ferner umfassend ein Bereitstellen einer Benachrichtigungskomponente (**30**) zum Registrieren von ähnlichen Client-Anforderungen (**34**) nach Ereignisdaten (**12**), und Liefern, zu im wesentlichen der gleichen Zeit, von angeforderten Ereignisdaten (**12**) an all Clients (**8**), die ähnliche registrierte Anforderungen (**34**) aufweisen.

3. Verfahren nach Anspruch 1, ferner umfassend die folgenden Schritte:
Empfangen von Ereignisdaten (**12**) an den entfernten Netzorten (**26**);
wenn empfangen an dem entfernten Netzort (**26**), Vorverarbeiten der Ereignisdaten (**12**), bevor die Ereignisdaten (**12**) in eine Ereignisdatenbank (**28**) hinein eingefügt werden, um zu bestimmen, ob eine Bedingung, wie in einem Trigger (**22**) aufgeführt, erfüllt wird; und
wenn die Trigger- (**22**) Bedingung erfüllt wird, Initiieren einer Aktion, die sich auf die Ereignisdaten bezieht, wobei die Aktion in dem Trigger (**22**) definiert wird.

4. Verfahren nach Anspruch 3, wobei ein Vorverarbeiten der Ereignisdaten (**12**) ein Bestimmen umfasst, ob die Ereignisdaten (**12**) ein Duplikat von anderen Ereignisdaten (**12**) in der Ereignisdatenbank (**28**) oder an dem entfernten Netzort (**26**) empfangen umfassen.

5. Verfahren nach Anspruch 4, wobei ein Initiieren der Aktion ein Abweisen einer Speicherung der Ereignisdaten (**12**) in der Ereignisdatenbank (**28**) umfasst, wenn sie ein Duplikat von anderen Ereignisdaten (**12**) umfasst.

6. Verfahren nach Anspruch 3, wobei dann, wenn die Ereignisdaten die Bedingung nicht erfüllen, sie vorübergehend außerhalb des Datenspeichers (**28**) gespeichert werden.

7. Verfahren nach Anspruch 3, wobei für Ereignisdaten (**12**), die an dem entfernten Netzort (**26**) empfangen werden, eine Anfrage ausgeführt wird, und eine Bedingung ausgewertet wird, die dann, wenn sie wahr ist, die Ausführung der Aktion verursacht.

8. Verfahren nach Anspruch 7, wobei die Aktion wenigstens eine Sequenz von Structured-Query-Language-(SQL) Anweisungen und/oder ein externes Skript umfasst.

9. Verfahren nach Anspruch 3, wobei der Trigger (**22**) einen Kopplungsmodus aufweist, der anzeigt, wann die Aktion ausgeführt werden sollte.

10. Verfahren nach Anspruch 3, wobei der Trigger (**22**) einem Verwalter des Netzes erlaubt, Ereignisse, Bedingungen und eine Aktion zu verbinden.

11. Verfahren nach Anspruch 3, wobei die Ereignisdaten (**12**) ein Primitiv-Ereignis umfasst.

12. Verfahren nach Anspruch 3, wobei die Ereignisdaten (**12**) ein Datenbankereignis umfassen.

13. Verfahren nach Anspruch 3, wobei die Ereignisdaten (**12**) ein zeitliches Ereignis umfassen.

14. Verfahren nach Anspruch 3, wobei der Trigger (**22**) einen Datenbank-Trigger umfasst.

15. Verfahren nach Anspruch 3, wobei der Trigger (**22**) einen zeitlichen Trigger umfasst.

16. Verfahren nach Anspruch 15, wobei der zeitliche Trigger ein Ereignis bei einer bestimmten Frequenz von einer spezifizierten Startzeit bis zu einer spezifizierten Endzeit signalisiert.

17. Verfahren nach Anspruch 3, wobei ein Initiieren einer Aktion ein Kommunizieren einer Nachricht in Übereinstimmung mit den Ereignisdaten an wenigstens einen Kundenort (**8**), der eine Teilnehmerberechtigung erworben hat, um die Ereignisdaten (**12**) zu empfangen, und ein Speichern der Ereignisdaten (**12**) in einem Datenspeicher (**28**) an dem entfernten Netzort (**26**) umfasst.

18. Verfahren nach Anspruch 17, wobei das Vorverarbeiten, wenigstens teilweise, während einer Periode auftritt, wenn auf den Datenspeicher (**28**) nicht zugegriffen werden kann.

19. Verfahren nach Anspruch 17, wobei die Nachricht, die in Übereinstimmung mit den Ereignisdaten (**12**) kommuniziert wird, eine Zusammenfassung (**44**) von wenigstens den Ereignisdaten (**12**) eines lokalen Netzes und Ereignisdaten (**12**) eines entfernten Netzes umfasst.

20. Verfahren nach Anspruch 1, wobei die entfernten Netzorte (**26**) einander mit ihren Ereignisdaten (**12**) aktualisieren.

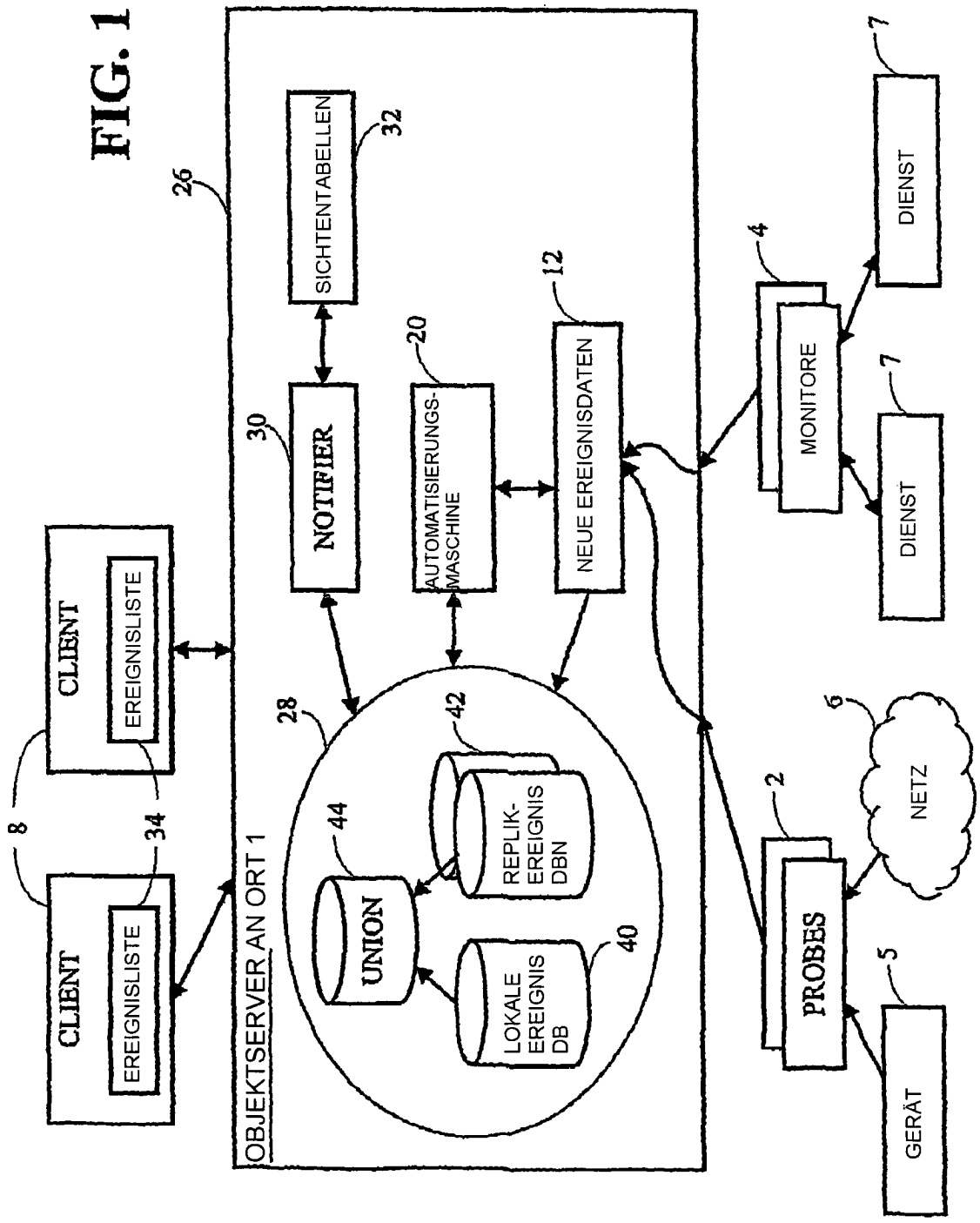
21. Verfahren nach Anspruch 1, wobei wenigstens ein entfernter Netzort (**26**) befähigt wird, die Eigentümerschaft einer Replik von entfernt erzeugten Ereignisdaten (**42**) zu übernehmen, um Modifikationen daran ohne Befehle von dem dazu gehörigen entfernten Netzort (**26**) vorzunehmen.

22. Ereignisdatenbank (**28**) zur Verwendung in einem Netzüberwachungssystem, wobei die Ereignisdatenbank (**28**) Ereignisdaten (**12**), die Ereignisse darstellen, die auf dem Netz auftreten, speichert, wobei die Ereignisdaten (**12**) durch eine Vielzahl von Überwachungseinrichtungen (**2**, **4**) gesammelt werden, die an einer Vielzahl von unterschiedlichen, entfernten Orten (**26**) auf dem Netz angeordnet sind, wobei die Ereignisdatenbank (**28**) umfasst:
eine Automatisierungsmaschine (**20**) zum Verarbeiten von einem oder mehreren Triggern (**22**), die in der Ereignisdatenbank (**28**) enthalten sind, wobei die Trigger (**22**) automatisch eine programmierte Antwort bei der Erfassung einer Bedingung, die durch gesammelte Ereignisdaten (**12**) erfüllt wird, vor der Einfügung der gesammelten Ereignisdaten (**12**) in die Datenbank (**28**) hinein initiiert;
eine lokale Tabelle (**40**), die an jedem entfernten Ort (**26**) gespeichert ist und die Ereignisdaten (**12**) enthält, die an dem entfernten Ort (**26**) erzeugt werden; und
eine oder mehrere Replik-Tabellen (**42**), die an jedem entfernten Netzort (**26**) gespeichert werden und die Ereignisdaten (**12**) enthalten, die an anderen entfernten Orten (**26**) erzeugt werden, wobei eine Zusammenfassung (**44**) der lokalen (**40**) und der Replik- (**42**) Tabellen erzeugt wird, um eine kombinierte Ereignisdatenbank (**28**) an dem entfernten Ort (**26**) zu bilden.

Es folgen 7 Blatt Zeichnungen

Anhängende Zeichnungen

FIG. 1



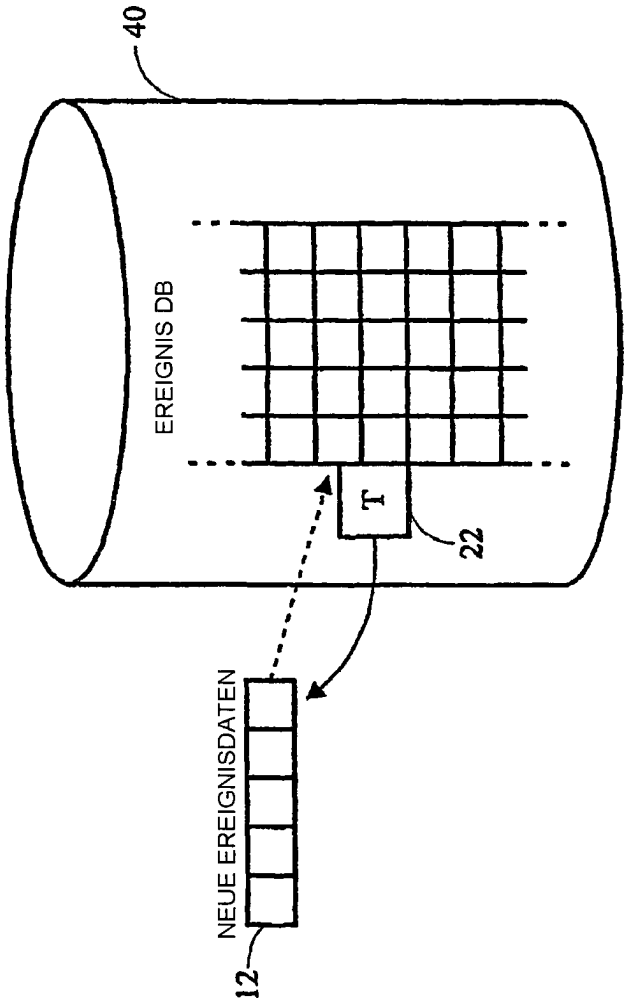


FIG. 2

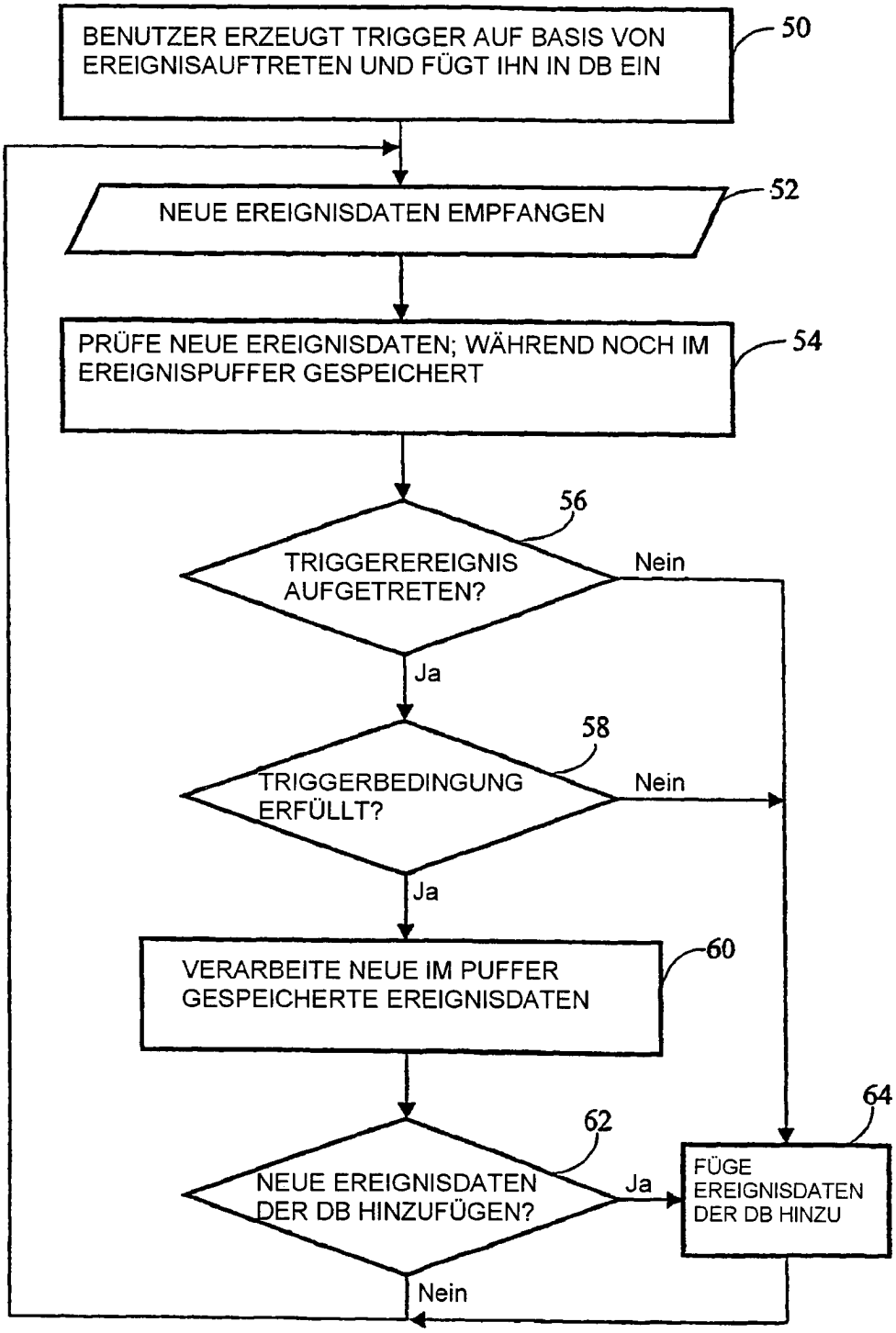

BENUTZER ERZEUGT TRIGGER AUF BASIS VON EREIGNISAUFTRETEN UND FÜGT IHN IN DB EIN
50
NEUE EREIGNISDATEN EMPFANGEN
52
PRÜFE NEUE EREIGNISDATEN; WÄHREND NOCH IM EREIGNISPUFFER GESPEICHERT
54
56
TRIGGEREREIGNIS AUFGETRETEN?
Nein
Ja
58
TRIGGERBEDINGUNG ERFÜLLT?
Nein
Ja
VERARBEITE NEUE IM PUFFER GESPEICHERTE EREIGNISDATEN
60
62
NEUE EREIGNISDATEN DER DB HINZUFÜGEN?
Ja
64
FÜGE EREIGNISDATEN DER DB HINZU
Nein

FIG. 3

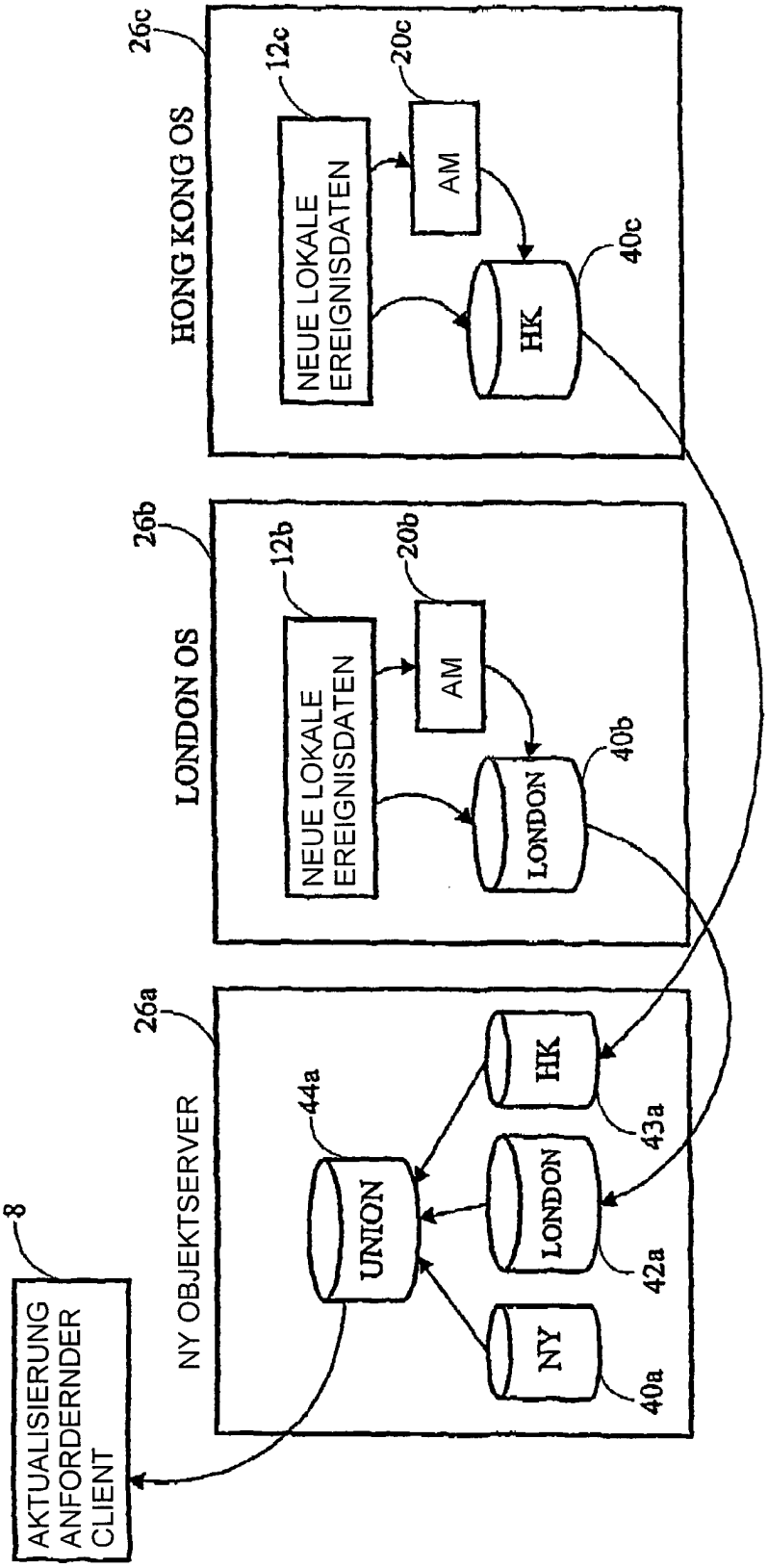

AKTUALISIERUNG ANFORDERNDER CLIENT
8
NY OBJEKTSERVER
26a
UNION
44a
NY
40a
LONDON
42a
HK
43a
LONDON OS
26b
NEUE LOKALE EREIGNISDATEN
12b
AM
20b
LONDON
40b
HONG KONG OS
26c
NEUE LOKALE EREIGNISDATEN
12c
AM
20c
HK
40c

FIG. 4

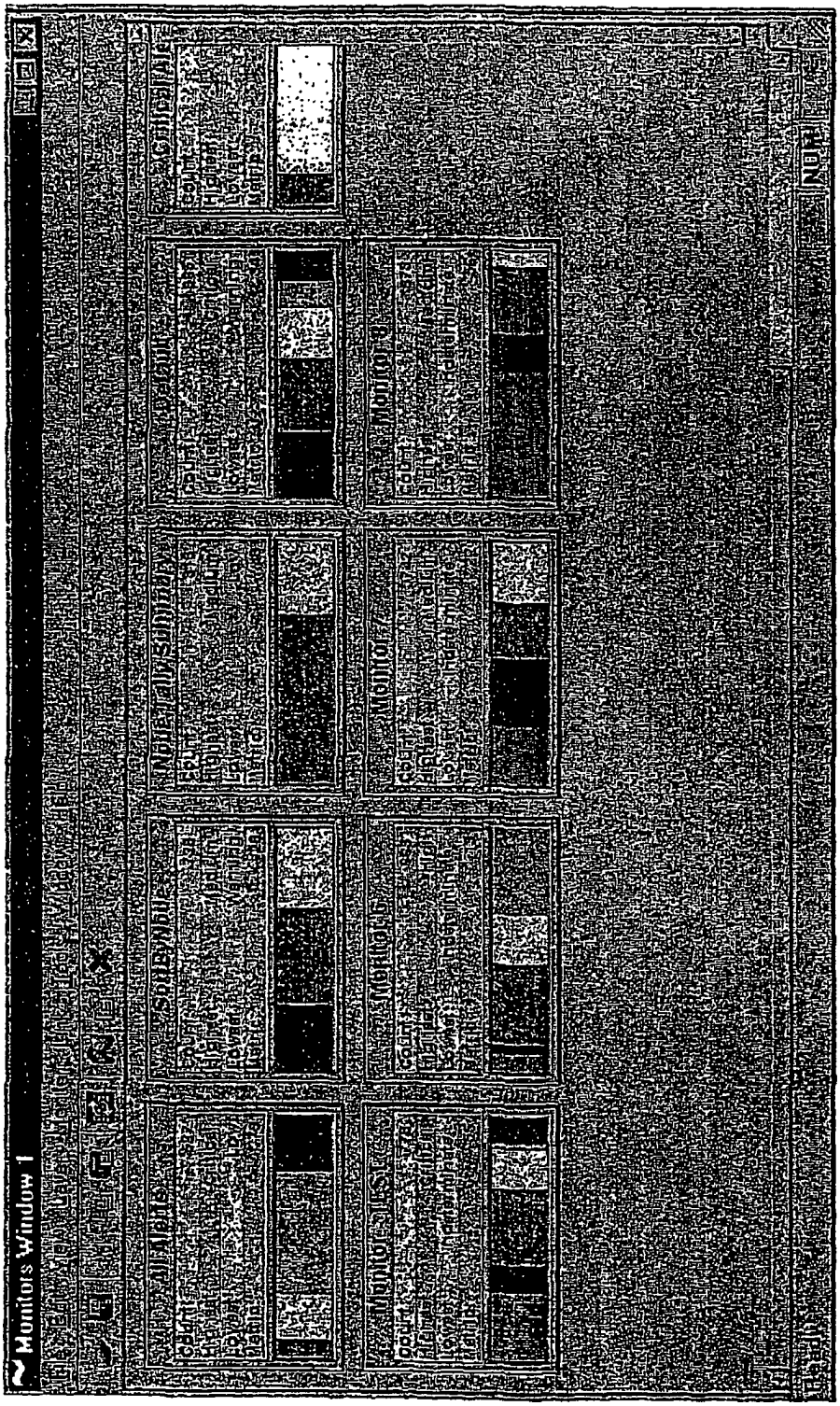


FIG. 5

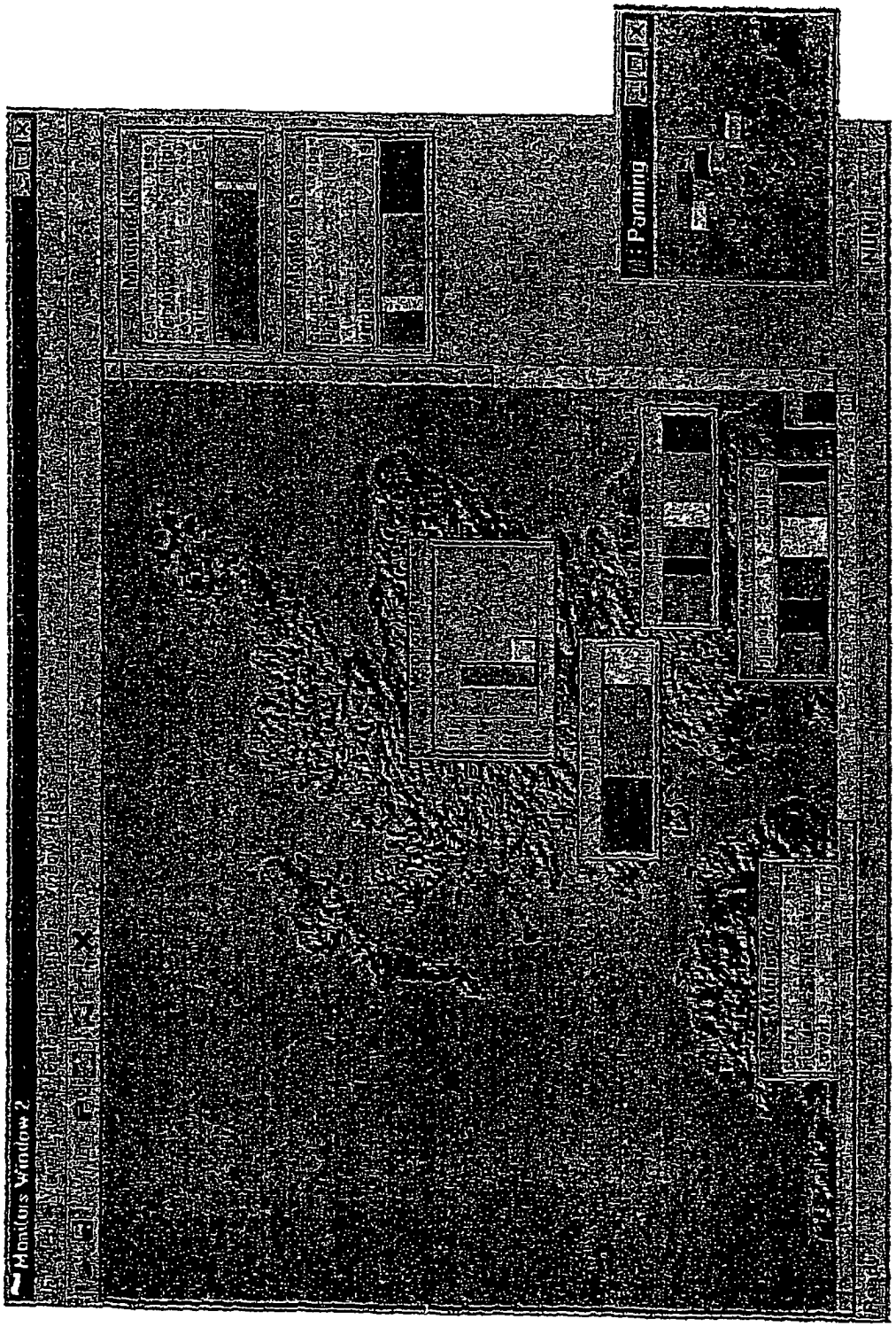


FIG. 6

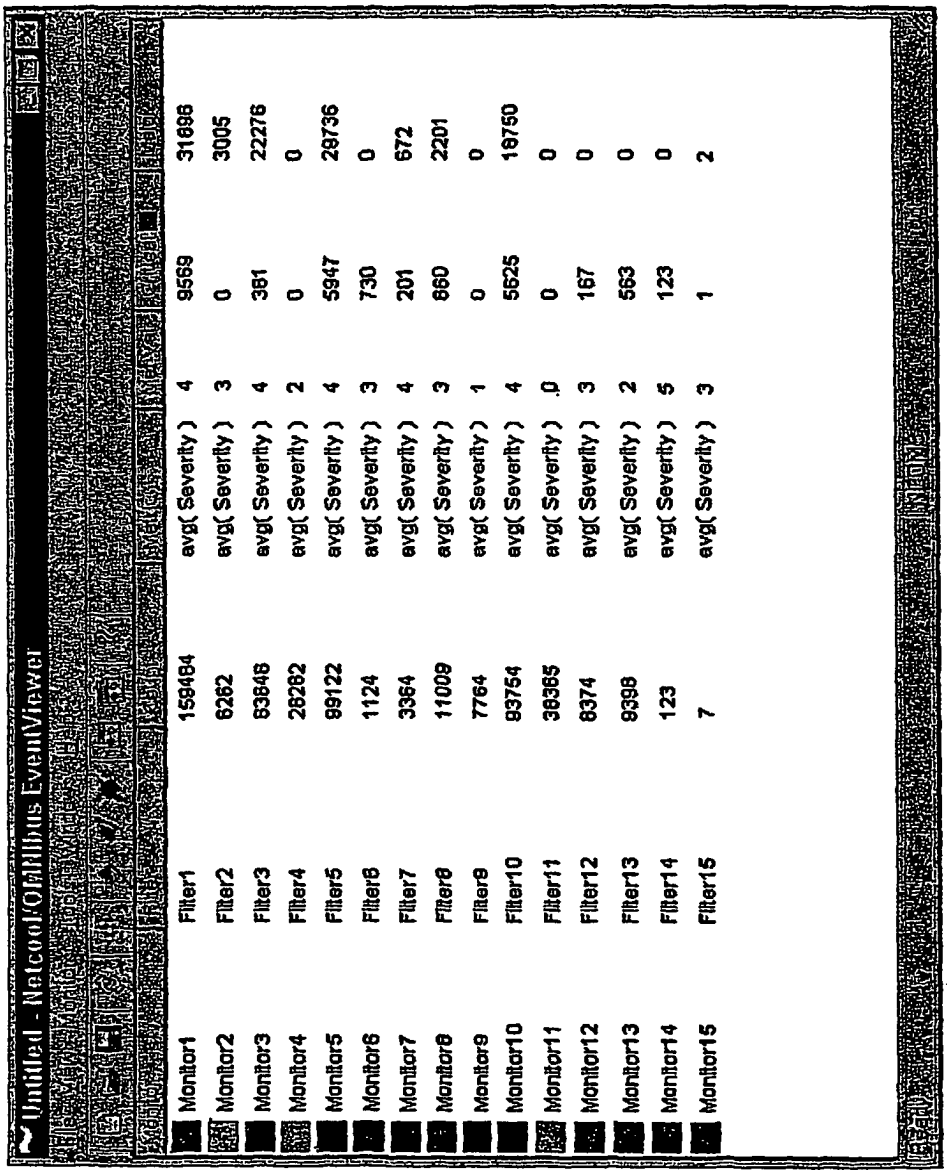

Untitled - NetcoolOmnibus EventViewer
Monitor1 Filter1 159484 avg(Severity) 4 9569 31686
Monitor2 Filter2 6262 avg(Severity) 3 0 3005
Monitor3 Filter3 83648 avg(Severity) 4 361 22276
Monitor4 Filter4 28262 avg(Severity) 2 0 0
Monitor5 Filter5 99122 avg(Severity) 4 5947 28736
Monitor6 Filter6 1124 avg(Severity) 3 730 0
Monitor7 Filter7 3364 avg(Severity) 4 201 672
Monitor8 Filter8 11009 avg(Severity) 3 660 2201
Monitor9 Filter9 7764 avg(Severity) 1 0 0
Monitor10 Filter10 83754 avg(Severity) 4 5625 18750
Monitor11 Filter11 38365 avg(Severity) 0 0 0
Monitor12 Filter12 8374 avg(Severity) 3 167 0
Monitor13 Filter13 9398 avg(Severity) 2 563 0
Monitor14 Filter14 123 avg(Severity) 5 123 0
Monitor15 Filter15 7 avg(Severity) 3 1 2

FIG. 7