[19]中华人民共和国国家知识产权局

[51]Int. Cl[7]
G06F 17/30

[12]发明专利申请公开说明书

[21]申请号 00810565.0

[43]公开日 2002年7月31日

[11]公开号 CN 1361890A

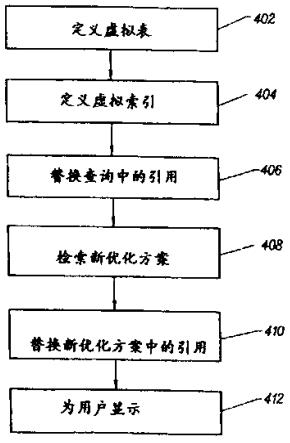
[22]申请日 2000.7.19 [21]申请号 00810565.0
[30]优先权
[32]1999.7.20 [33]US [31]09/356,797
[86]国际申请 PCT/US00/40424 2000.7.19
[87]国际公布 WO01/06417 英 2001.1.25
[85]进入国家阶段日期 2002.1.18
[71]申请人 计算机联合思想公司
地址 美国纽约
[72]发明人 爱德华·科斯克尤科 斯利库玛·曼诺
黄-文·沃
[74]专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 李德山

权利要求书2页 说明书21页 附图页数4页

[54]发明名称 观察改变索引对查询优化方案的影响的数据库系统

[57]摘要

观察改变数据库表索引对诸如SQL语句的数据库查询的优化方案的 影响的方法和装置创建模拟数据库中原始表(308)的结构的虚拟表 (312)。通过复制原始表(308)来创建虚拟表(312),其中排除原始表中 的任何数据。复制任何与原始表(308)相关的现有原始索引(310)以便定 义与虚拟表(312)相关的虚拟索引(314)。查询中针对原始表的引用被针 对虚拟表(408)的引用替换。数据库管理系统则为查询确定一个新的优 化方案(324)。



ISSN1008-4274

权　利　要　求　书

1.在数据库管理系统中观察一个查询的初始优化方案的变化的方法，上述查询具有一个针对某个原始表的引用，上述原始表具有在一个数据库中存储的数据，上述方法包括:

复制排除了数据的原始表以定义一个虚拟表;

提供一个与虚拟表相关的虚拟索引;

在查询中用针对虚拟表的引用替换针对原始表的引用;

向查询增加针对虚拟索引的引用;

确定查询的一个新优化方案；和

2.如权利要求 1 所述的方法，其中还包括比较新优化方案和初始优化方案的步骤。

3.如权利要求 2 所述的方法，其中在比较新优化方案和初始优化方案之前还包括:

在新优化方案中用针对原始表的引用替换针对虚拟表的引用；和

向用户显示具有针对原始表的引用的新优化方案。

4.在数据库管理系统观察查询的初始优化方案的变化的方法，上述查询具有：(i)针对原始表的引用，上述原始表具有在一个数据库中存储的数据；和(ii)针对与原始表相关的原始索引的引用，上述方法包括:

复制排除了数据的原始表以定义一个虚拟表;

改变原始索引以定义一个与虚拟表相关的虚拟索引;

确定查询的一个新优化方案；和

5.如权利要求 4 所述的方法，其中还包括比较新优化方案和初始优化方案的步骤。

6.如权利要求 5 所述的方法，其中在比较新优化方案和初始优化方案之前还包括：在新优化方案中分别用针对原始表和原始索引的引用替换针对虚拟表和虚拟索引的引用；和

向用户显示具有针对原始表和原始索引的引用的新优化方案。

7.在数据库管理系统观察查询的初始优化方案的变化的方法，上述查

询具有：(i)针对原始表的引用，上述原始表具有在一个数据库中存储的数据；和(ii)针对与原始表相关的原始索引的引用，上述方法包括:

复制排除了数据的原始表以定义一个虚拟表;

在查询中用针对虚拟表的引用替换针对原始表的引用;

从查询中删除针对虚拟表的引用;

确定查询的一个新优化方案；和

8.如权利要求 7 所述的方法，其中还包括比较新优化方案和初始优化方案的步骤。

9.如权利要求 8 所述的方法，其中在比较新优化方案和初始优化方案之前还包括:

在新优化方案中用针对原始表的引用替换针对虚拟表的引用；和

向用户显示具有针对原始表的引用的新优化方案。

说　　明　　书

观察改变索引对查询优化方案的影响的数据库系统

技术领域

本发明涉及观察改变数据库表索引设计的影响。更具体地，本发明涉及在改变数据库表的索引时使用虚拟表和虚拟索引确定数据库查询的优化方案。

背景技术

实现并保持计算机数据访问的简单有效是多数计算机用户的共同目标。另外，随着现代计算机处理能力的增加，用户可能需要组织和存储更多的数据。一种用于组织计算机数据的系统是数据库，数据库通常被看作一组在某种可记录介质中以没有不必要的冗余的方式共同存储的逻辑相关信息对象或文件。数据库为各种应用或程序提供良好的服务，并且利于这些应用或程序的访问。

在多数数据库中，从外部观察数据被组织成表。各个表通常包含一系列定义表的列的字段。表的每个行包括一个单独的记录。对于表中的每行数据，该数据的具体内容被物理存储在数据库中。因而当数据库用户从表中请求具体信息时，存储数据的适当部分被检索出来并且提供给用户。

一个被称作"数据库管理系统"(" DBMS ")的程序为用户提供一个针对数据库的接口。DBMS 为数据库提供允许用户访问数据库中存储的信息对象的结构。DBMS 根据用户的信息请求或 "查询 "识别并检索某些信息对象。具体信息对象的检索取决于信息对象中存储的信息与用户对系统的请求之间的相似程度。通过比较信息对象和信息请求所附的某些属性的数值来测量相似程度。

例如，如果表" Employee "包含字段" Name "，" Dept "，" Age "和" Salary "，并且用户期望找到在电子部门工作的雇员子集，可以使用下面查询:

```
SELECT    Name, Salary, Age
FROM  Employee
WHERE    Dept = "Electronics"
```

为了利于检索处理，经常对数据库中的信息对象加"索引"以便通过分配标识对象内容的描述符来概括对象的特征。提取这些信息对象的特征的处理可以引导 DBMS 根据用户的具体查询检索出数据库中的具体内容，上述处理被称作"索引"。

为了建立表索引， DBMS 通常扫描表，检索表中各行和各列的数据，并且向索引添加数据，其中索引经常具有 B-树结构。更多有关 B-树结构的信息参见 Patrick ONeil "数据库 - 原理，编程，性能" Morgan Kaufmann Publishers (1994)，这里参考引入了上述著作。DBMS 顺序读取表中的每个数据记录，将各个数据记录复制到一个临时空间，在必要时对数据记录进行排序，并且最终创建一个索引数据结构。

然而建立表索引的处理通常消耗大量时间和资源。例如，创建具有数百万行的表的索引可能需要几天时间。此外，创建这种表的索引通常需要使用数百兆字节的临时工作区以便在创建索引之前复制和排序数据。自然地，创建或改变索引的处理会相应占用大量的时间。诸如数据仓库中使用的数据库表可以具有数十亿甚至数千万亿行数据。用户可能需要数周甚至数月来创建这种规模的表的索引。

其它因素使得涉及创建索引的时间问题更加复杂。具体地，在创建索引之后，数据库需要时间测试索引并且返回性能统计结果，用户也需要时间分析结果。

对于一个典型的索引数据库系统，完成一个查询通常需要两个步骤。第一个步骤是确定在索引中具有关联描述符或索引记录的查询子句以便检索那些索引记录，并且预先限制要考虑的信息对象集合。第二个步骤通常需要从第一步骤得到信息对象集合并且依次检查各个信息对象以确定是否满足查询条件。

结构化查询语言（SQL)已经演变成数据库查询或语句的标准语言。SQL 接口允许用户通过批处理文件或嵌入到诸如 C， COBOL 等等的宿主语言中的方式交互表述数据库表的关系操作。 在 SQL 中提供允许用户处理数据的操作符，其中各个操作符操作一或多个表并且产生一个新表作为结果。

在调整 SQL 语句或查询的处理中，用户经常希望知道改变索引会怎样影响查询的性能。如上所述，索引可以被加到数据库中以利于提高执行查询的速度，尤其是在较大的表中，其中索引可以从根本上改进性能。另一方面，由于数据库中的数据量，增加或改变索引需要大量的时间和资源以供数据库建立索引。因而用户经常面临两难的选择，即或者花费建立索引所需的时间和资源，风险是新索引并没有显著改进性能，或者是不建立索引，风险是无法确定索引是否能够改进性能。

Oracle 数据库管理系统为用户提供了观察 SQL 语句的"优化方案"的能力。当数据库分析 SQL 语句时数据库自动确定 SQL 语句的优化方案。优化方案在没有实际执行 SQL 语句的情况下显示数据库会如何检索满足 SQL 语句需求所需的数据。具体地，优化方案显示这样的信息，例如什么表会被首先访问，中间结果集合会如何被联接，是否会使用索引，如果是这样的话会如何解释索引。因而通过观察具体 SQL 语句的优化方案，用户可以获得关于 SQL 语句在数据库中执行的效率的估测。

在较大的数据库管理系统中，查询优化对于时间和资源消耗最小化尤其重要。因而，用户观察 SQL 语句的优化方案并且确定索引变化对优化方案的影响的能力变得同等重要。

图 1 是关于常规方法 100 的流程图，其中上述常规方法 100 观察数据库表的索引变化对 SQL 语句优化方案的影响。在步骤 110，针对 SQL 语句产生一个初始优化方案。在步骤 120 中， 创建，丢弃或修改 SQL 语句中引用的表的索引。在步骤 130，针对 SQL 语句产生一个新的优化方案。最终在步骤 140，用户比较新优化方案和初始优化方案以确定改变索引会改进性能还是降低性能。

然而如上所述，图 1 的常规方法 100 需要过多的时间和资源来创建，

放弃或改变索引。此外，数据库为建立各种优化方案而收集必要的统计数据会需要过多的时间和资源。如果在某个生产环境中不使用数据库，则可能采用花时间通过常规方法改变索引的手段。然而如果在生产中使用数据库，则由于在速度，资源和总体性能方面非常消极的影响，花费完成图 1 所示的改变所需的时间和精力是非常不可行的。例如，在使用图 1 的方法的情况下，如果一个工具或应用程序依赖一个现有索引并且用户在步骤 120 改变或放弃上述索引，则数据库可以停机并且整个系统会死锁。

因而，对于预览索引改变对优化方案的影响的常规方法，用户经常被迫尽量避免调整索引的尝试。于是，当 SQL 语句在数据库中执行时这经常导致会费用大量时间和精力的数据库不能实现最优索引分布或优化方案，尤其是对于 较大的数据库管理系统。因而需要使用一种更快速并且更加有效的方式来改变数据库表索引设计并且创建这些索引的优化方案。

发明内容

本发明允许用户在新索引被加到数据库表中，从表中丢弃现有索引或修改表的现有索引时能够看到数据库查询的优化方案如何发生改变。

一个方法和装置为用户提供了一种框架，这种框架允许用户在不必花费常规方法所需的时间和资源的情况下试验表的索引分布并且预览各种索引分布构造对数据库查询优化方案的影响。

根据本发明的各个方面，创建一个模拟要测试的数据库表或数据库中的"原始表"的结构的虚拟表。通常通过复制原始表来创建虚拟表，其中排除原始表中的任何数据。因而如果数据被存储在原始表的行中，则数据行不被复制到虚拟表中。然而复制任何与原始表或"原始索引"相关的现有索引以定义一个与虚拟表相关的虚拟索引。

通过在复制原始表定义虚拟表时排除数据，可以方便快速地修改相关虚拟索引并且保持原始表的总体结构。可以非常快速地增加新索引和放弃现有索引。并且如果没有原始索引，则可以方便地创建新虚拟索引。

在查询中，针对原始表的引用被针对虚拟表的引用替换。数据库管

理系统则为查询确定一个新的优化方案。由于使用虚拟表和虚拟索引确定新优化方案，所以方案的检索比使用原始表和任何相关原始索引产生的方案更快速。这是由于当复制原始表定义虚拟表时排除了原始表中的实际数据。因而在改变索引设计之后可以快速识别对优化方案的任何改变。

在向用户显示新优化方案之前，用原始表和原始索引的名称替换新优化方案中任何对虚拟表和任何虚拟索引的引用。通过这种方式，用户可以比较新优化方案和初始优化方案并且在不关心或甚至不需要知道虚拟对象在产生新优化方案过程中的使用的情况下分析变化。

附图说明

图 1 是关于常规方法 100 的流程图，其中上述常规方法 100 观察数据库表的索引变化对 SQL 语句优化方案的影响;

图 2 是图解一个示例性硬件环境的模块图，其中可以实现本发明的示例性实施例;

图 3 是图解基于本发明一个示例性实施例的系统的模块图，上述系统观察数据库表的索引变化对数据库查询优化方案的影响；和

图 4 是关于基于本发明一个示例性实施例的常规方法的流程图，上述常规方法观察数据库表的索引变化对数据库查询优化方案的影响。

具体实施方式

图 2 是图解一个示例性硬件环境的模块图，其中可以实现本发明的示例性实施例。在图 2 的硬件环境中，一个客户端计算机 200 被连接到一个服务器计算机 202。客户端计算机 200 和服务器计算机 202 均可以包含处理器，随机访问存储器(RAM)，只读存储器(ROM)，键盘，显示器，固定和/或可移动数据存储设备，和数据通信设备。

在图 2 中，服务器计算机 202 与一个数据库 212 通信，数据库 212 最好是一个 Oracle 数据库。本领域的技术人员应当理解，可以使用诸如微软公司 SQL 服务器数据库， Sybase SQL 服务器数据库和 IBM DB2 数据库的其它数据库。

本领域的技术人员可以认识到，客户端计算机 200 和服务器计算机

202 可以使用参照图 2 描述的上述部件的任何组合或任意数量的不同部件，外设和其它设备。本领域的技术人员还可以认识到，可以在一个单独计算机而不是多个共同联网的计算机上实现本发明的示例性实施例。

通常使用数据库管理系统软件实现本发明的示例性实施例，例如 PLATINUM Technology 制作并销售的 SQL-Station 软件，虽然可以通过任意的数据库管理系统软件来实现，例如 Oracle 销售的 Developer / 2000 开发包或 IBM 销售的 DB2 产品。此外，可以实现本发明示例性实施例的 PLATINUM SQL-Station 软件可以和其它软件结合使用，例如 Oracle Developer/2000 软件和 IBM 销售的 DB2 产品。

在图 2 中，软件包含客户端计算机 200 执行的 SQL-Station 客户端程序 204 和 Relational Extender 客户端程序 206。软件还包含服务器计算机 202 执行的 Oracle 服务器程序 208 和 Relational Extender 程序 210。在其相应计算机 200 或 202 的操作系统的控制下执行这些程序，上述操作系统包括 Windows 95, Windows NT, OS/2, AIX, MVS, Unix 等等。本领域的技术人员会认识到，可以使用上述程序的任意组合或任意数量的不同程序来实现本发明的示例性实施例。

SQL-Station 客户端程序 204 和 Relational Extender 客户端程序 206 针对 Oracle 服务器程序 208 和 Relational Extender 服务器程序 210 管理的数据库 212 产生执行各种搜寻和检索函数，即查询的命令。在最优实施例中，这些查询符合 SQL 标准要求，虽然在不偏离本发明范围的前提下也可以使用其它类型的查询。查询调用 Oracle 服务器程序 208 和 Relational Extender 服务器程序 210 执行的函数，例如对用户和系统数据进行定义，访问控制，解释，编译，数据库检索和更新的函数。

通常情况下，DBMS 软件，SQL 查询和其中导出的指令均可实际包含在诸如一或多个数据存储设备和/或数据通信设备的计算机可读介质中，或者可以从中读取出来。此外，RDBMS 软件，SQL 查询，和其中导出的指令均由在被客户端计算机 200 和/或服务器计算机 202 读取和执行时导致客户端计算机 200 和/或服务器计算机 202 执行实现和/或使用本发明的实施例所必需的步骤的指令组成。

图 3 是图解基于本发明一个示例性实施例的系统的模块图，上述系统确定数据库表的索引变化对数据库查询优化方案的影响。图中显示两个用户或"模式"，即： TUTOR: 302 和" PAFO-HR" 304，均访问一个数据库 306，数据库 306 最好是一个诸如 Oracle 8 数据库的 Oracle 数据库。

在图 3 中，各个用户 302， 304 控制其拥有并且存储在其帐户内的表中的对象和信息。例如，用户 PAFO-HR 304 控制原始表 308 和一个相关索引 310，以及虚拟表 312 和一个与虚拟表 312 相关的虚拟索引 314。表 308， 312 和对应的相关索引 310， 314 被存储在数据库上。

各个用户 302， 304 通常控制该用户帐户内存储的信息以便排斥数据库 306 的其它用户。然而用户可以被授予各种涉及其它用户帐户的特权。通常，只可以从一个用户向另一个用户授予涉及一个用户的帐户的特权。例如在图 3 中，只有 PAFO-HR 304 具有向其它用户授予涉及 PAFO-HR 304 帐户的特权的能力。

一个这样的特权是" READ"，即允许访问另一个用户的帐户中的信息。例如在图 3 中，用户 PAFO-HR 304 允许用户 TUTOR 302 或授予 TUTOR 302 READ 特权来访问 PAFO-HR 304 帐户内的表中存储的信息和对象。因而 TUTOR 302 能够查询 PAFO-HR 的帐户内的表并且访问 PAFO-HR 304 帐户中的这种信息。另一方面，一个第三用户 " SCOTT " (未示出)没有被授予这种涉及 PAFO-HR 帐户的 READ 特权，所以 SCOTT 不能访问 PAFO-HR 304 控制的表。

图 3 的系统中具有的另一个特权是 " CREATE_TABLE "特权或特权集合，上述特权也可以从一个用户授予另一个用户。在得到一个用户的授权的情况下， CREATE - TABLE 特权允许其它用户在某个用户的帐户内创建表并且在表中存储对象。

由于为其它用户提供更强的单方面选择修改或删除另一个用户帐户内存储的信息的能力，用户经常不希望向数据库系统中其它用户授予 CREATE_TABLE 特权。因而在图 3 的数据库环境中， TUTOR 302 没有被授予涉及 PAFO-HR 帐户的 CREATE-TABLE 特权。

在图 3 中，根据本发明的示例性实施例，在数据库 306 上存储一个名称为" Oracle 方案分析器"(" PAFO") 316 的软件包。这个软件包可以被数据库系统的用户 302，304 访问。与 TUTOR 302 不同的是，软件包 PAFO 316 具有授予 PAFO 316 的 PAFO-HR 304 CREATE_TABLE 特权和数据库管理员 (DBA)授予的若干其它特权。因而一个登录到 TUTOR 302 帐户上的用户可以使用 PAFO 软件包 316 实现访问和试验 PAFO-HR 304 帐户中的表和索引的任务，如果没有登录则不具有完成上述任务的特权。

在图 3 中， TUTOR 302 调用软件包 PAFO 316 执行本发明的方法。当调用 PAFO 316 时， PAFO 316 为 PAFO-HR 304 帐户创建并动态配置一个过程 318。接着在 PAFO-HR 304 帐户中执行配置的过程 318。配置的过程 318 可以向 TUTOR 302 帐户的用户授予访问 PAFO-HR 304 帐户的必要特权。配置的过程接着向 TUTOR，即用户的帐户授予对虚拟表的 READ (非写)特权。通过这种方式， TUTOR 可以在没有 CREATE_TABLE 特权的情况下访问原始表 308 和虚拟表 312。此后参照图 3 和 4 描述配置的过程 318 的功能。

在图 4 的步骤 402 中，过程 316 识别 SQL 语句 320 中引用的原始表 308 并且复制原始表 308 以定义虚拟表 312。在步骤 402 中，这个复制包含向新表中复制原始表统计数据，其中包含列统计，柱状图和分段存储。通过这种方式，表面上虚拟表包含的数据行和原始表一样多。然而最好从定义虚拟表 312 所复制的信息中排除原始表 308 中的任何实际数据。因而当数据被存储在原始表 308 的行中时，数据行不被复制到虚拟表 312 中。

在步骤 402 中，通过一个原始表复本进行创建和操作有许多好处。用户对原始表 308 的访问不被中断并且变化不降低性能。在生产环境中，执行的应用不受影响。当使用 Oracle 数据库时，不需要针对原始表 308 修改 Oracle 目录中的内容。此外由于虚拟表不包含任何数据行，因而索引建立非常快速。

在步骤 402 中，如果在 PAFO-HR 304 帐户中有其它原始表，则也

可以复制这些表以便定义对应的虚拟表。并且由于如下所述的原因，最好维护一个列表(未示出)，在该列表中虚拟表的名称与其建立所依据的原始表相关。

在步骤 404 中定义与虚拟表 312 相关的虚拟索引 314。在这个步骤中，如果存在任何与原始表 308 相关的原始索引 310，则过程 316 复制原始索引 310 以定义虚拟索引 314。例如如果针对图 3 的原始表 308 中示出的两个列定义了原始索引，则通过复制原始索引针对虚拟表 312 中示出的两个列创建对应的虚拟索引。

与原始索引 310 相关的统计数据也被复制到虚拟索引 314 中，并且可以根据用户的指示来设置。通过这种方式，虚拟索引具有与原始索引相同的数据结构，其中包含相同的约束和定义。因而对于所涉及的任何优化方案，虚拟索引的结构均与原始索引相同。

在步骤 404 中，如果没有与原始表 308 相关的原始索引 310，则用户可以通过过程 316 创建和定义虚拟索引 314。并且在一个用户希望进行没有索引的试验的情况下，用户可以使用配置的过程 316 通过删除原始索引 310 中出现的任何索引来进行简单选择以定义虚拟索引 314。这里，在 PAFO-HR 304 上创建过程 318 并且执行过程 318 以便向 TUTOR 302 授予 READ 特权。

在步骤 406，当定义虚拟索引 314 之后，PAFO 316 用一个针对虚拟表 312 的引用替换 SQL 语句 320 中任何针对原始表 308 的引用。另外，用虚拟索引 314 的名称替换 SQL 语句 320 中任何针对原始索引 310 的引用。

被替换名称修改的 SQL 语句 322 接着被发送到数据库服务器。帐户 TUTOR 302 的用户会相信其正在访问原始表 308，但实际上正在访问虚拟表 314。此外，通过创建原始表的一个复本并且简单改变 SQL 语句中引用的对象的名称，不必进行在原始表上费时又费资源的创建新索引的操作。

由于 PAFO 程序 316 已经改变 SQL 语句 320 并且定义了经过修改的 SQL 语句 322，数据库在步骤 408 中解释经过修改的 SQL 语句 322

以确定虚拟表的新优化方案 324。

在步骤 410 中， PAFO 过程 316 接着修改数据库服务器返回的新优化方案 324 中的信息以便定义一个经过修改的优化方案 326。具体地，分别参照原始索引 310 和原始表 308 替换新优化方案 324 中针对虚拟索引 314 和虚拟表 312 的任何引用。

接着在步骤 412 中向用户显示用替换名称修改的优化方案 326。因而用户的感觉是仅仅在操作初始对象 308 和 310。通过使用上述方法，改变索引和检索新优化方案所需的时间经常从若干小时减少到若干秒。因而用户在不需要关心或察觉虚拟表和虚拟索引的使用的情况下就能够得到改进新优化方案检索的速度与系统性能的好处。

用户可以将经过修改的优化方案 326 与任何针对原始表 308 和原始索引 310 创建的初始优化方案相互比较。如果新优化方案更好，例如似乎会改进数据库中执行 SQL 语句 320 的速度和效率，则用户可以选择在原始表 308 上实际建立虚拟索引 314。反之，如果不会改进性能，则用户可以试验不同的虚拟索引 314 或在不浪费时间和资源的情况下继续使用原始索引 310，其中如果不继续使用原始索引 310 则需要构造虚拟索引 314。

可以由多个用户在相同对象上同时执行如图 3 和 4 所述的各个功能。这些功能不干扰对象的其它用户，也不对性能产生显著的影响，这允许在一个生产系统中执行这些功能。

例子

在一个例子中，一个用户登录到图 3 的 TUTOR 302 帐户上。 在这个例子中，帐户 PAFO-HR 304 中的原始表 308 是一个存储某公司或企业的雇员名称的表，此后被称作 "EMPLOYEES" 表。TUTOR 302 被授予涉及 PAFO-HR 帐户的 READ 特权，所以 TUTOR 302 能够向 EMPLOYEES 表发送查询。

已经针对 EMPLOYEES 表创建了虚拟表 312，其中虚拟表名称是

"T_########1 "，初始 SQL 语句 320 如下所示:

```
SELECT *
FROM hr. employees e
WHERE hiredate > :H_Date
```

下面的 SQL 语句作为修改后的 SQL 语句 322 被发送到服务器，其中原始表名称" EMPLOYEES "被虚拟表名称" T_########1"替换:

```
SELECT *
FROM hr."T_########l" e
WHERE hiredate > :H_Date
```

如果 SQL 语句 320 包含优化线索，则 PAFO 316 检查包含原始表名称" EMPLOYEES"的线索或一个与其相关的、作为线索的参数的索引。例如，假定用户正在操作下面 SQL 语句 320:

```
SELECT /*+ INDEX (e, I_EMP_HIREDATE) */*
FROM EMPLOYEES e
WHERE hiredate > :H_Date
```

当用户请求观察 SQL 语句 320 的优化方案时，下面消息被发送到数据库服务器以作为修改的 SQL 语句 322:

```
SELECT /*+ INDEX (e, I_########5) */*
FROM hr.T_########1 e
WHERE hiredate > :H_Date
```

为了避免系统中的性能下降，各个用户最好只被允许创建

EMPLOYEES 的一个单独复本以定义" T_########1 "虚拟表。在已经为该用户创建一个复本的情况下最好限制用户创建新的复本。

对于下述过程，最好指定原始表的名称而不是虚拟表名称。所以如果用户在 EMPLOYEES 表上创建一个第一虚拟索引，用户可以在 EMPLOYEES 表的虚拟复本上创建一个第二虚拟索引或放弃一个索引，但是表名称参数应当是原始表的名称。在后一种情况下，即用户重新使用之前创建的一个虚拟表的情况下， PAFO 软件包保证原始表中仍然存在检索的索引名称。例如，可以使用下面的 SQL 查询:

```
SELECT index_name FROM expl_indexes
WHERE owner = :Index_Owner
AND index_name = :Index_Name
AND table_name = :Table_Name
```

最终，由于通常只返回原始表上存在的索引的名称，应当通过如下所述的语句查询虚拟索引:

```
SELECT owner, index_name FROM expl_indexes
WHERE table_owner = :Virtual_table_Owner
AND table_name =:Virtual_Table_Name
AND (owner, index _name)NOT IN
(('owner 1','index1'), (`owner2','index2'),...),
```

其中('owner1 '， ' index1 ')对是返回索引的拥有者和名称。

在这个例子中，下面步骤被用于观察数据库表的索引变化对数据库查询优化方案的影响。

步骤 1: 创建虚拟表

可以执行下面过程以创建虚拟表，假定一个虚拟表尚未创建:

```
begin
pafo.explain_virtual.create_virtual_table
(:table_owner, :table_name);
end;
```

如果虚拟表已经存在，则返回一个差错报告。参数的定义如下所述:

约束变量	模式	数据类型	描述
table_owner	Input	VARCHAR2(30)	原始表拥有者的名称。将为其创建虚拟表。
table_name	Input	VARCHAR2(30)	原始表的名称。

表 1

在表 1 中， VARCHAR2(30)表示一个多达 30 个字符的字符串。并且，模式= INPUT 指示数据被输入到过程中。反之，如以下表中所使用的，模式= OUTPUT 表示从过程中检索出的数据。

步骤 2: 定义虚拟表和索引名称

在创建虚拟表之后，可以执行下面过程返回定义虚拟索引所复制的虚拟表名称和索引名称。

```
begin
  pafo.explain_virtual.get_virtual_table_aliases
(:table_owner, :table_name, :virtual_table_name, :virtual_index_count,
:original_index_owners, :original_index_names, :virtual_index_names);
```

end;

约束变量	模式	数据类型	描述
table_owner table_name virtual_table_name	Input Input output	VARCHAR2(30) VARCHAR2(30) VARCHAR2(30)	原始表拥有者 原始表名称 被用来创建虚拟表的名称。
virtual_index_count	Output	INTEGER	原始表上存在的索引的数量。在虚拟表上重新创建所有这些索引但具有不同的名称。
original_index_owners	Output	Array of VARCHAR2(30)	索引拥有者数组。这些名称对于初始和虚拟索引均是相同的。(数组大小=:virtual_index_count)
original_index_names	Output	Array of VARCHAR2(30)	索引名称数组。这些名称是原始表上使用的名称。(数组大小=:virtual_index_count)
virtual_index_names	Output	Array of VARCHAR2(30)	索引名称数组。这些名称是被用来创建"original_index_names"中列出的索引的复本的名称。"original_index_names"和"virtual_index_names"数组中的名称之间最好有 1-1 对应关系。(数组大小=:virtual_index_count)

表 2

步骤 2 的调用通常假定使用 CREATE_VIRTUAL_TABLE 过程创

建虚拟表。这意味着如果虚拟表存在于一个集簇中，则该集簇与原始表所存在的集簇相同。即虚拟表不是将一个非集簇表变成集簇表的请求的产物。否则返回一个差错报告。

步骤 3: 缺省统计数据

新虚拟索引最好具有现实统计数据以便保证创建一个相当于具有在原始表上创建的索引的优化方案。EXPLAIN_VIRTUAL 提供过程 DEFAULT_NEW_INDEX_STATS 协助用户设置适当的统计数据。

如果原始表具有已存在的索引，则那些索引的存储信息和统计数据会被用来产生虚拟索引的缺省存储信息和统计数据。如果没有分析出已存在的索引，则会使用有限制的缺省值。

在执行 DEFAULT_NEW_INDEX_STATS 之后，向 PAFO 返回缺省存储信息和统计数据以便显示和修改。允许用户改变这些统计数据以便较好地标识索引性质。为了帮助用户判定如何改变统计数据， PAFO 允许用户显示关于当前在表上创建的其它索引的统计数据。

可以执行下面过程以获得缺省统计数据:

```
begin
  pafo.explain_virtual.default_new_index_stats(:table_owner, :table
_name,:ind_name,:ind_col_names, :num_ind_columns, :default_tablespa
ce, :init_trans, :max_trans, :pct_free, :btree_levels, :leaf_blocks, :avg_lea
f_blocks, :avg_data_blocks, :cluster_factor, :blocks_allocated, :extents_al
located, :distinct_values);
end;
```

参数定义如下所述:

参数	模式	数据类型	用途
table_owner	input	VARCHAR2(30)	拥有会在其上创建索引的表的 ORACLE 帐户。
table_name	Input	VARCHAR2(30	表名称。
ind_name	Input	VARCHAR2(30)	建议创建的索引的名称。
ind_col_names * 1	Input	Array of VARCHAR2(30)	将创建索引的列。数组中列的顺序应当匹配用户请求的顺序。
num_ind_columns	Input	NUMBER	列的数量；即数组 IND_COL_NAMES 中的元素数量。
default_tablespace	Output	VARCHAR2(30)	将在其上创建索引的表空间名称。
init_trans	Output	NUMBER	创建索引的 INIT_TRANS 参数。
max_trans	Output	NUMBER	创建索引的 MAX_RANS 参数。
pct_free	Output	NUMBER	创建索引的 PCT_FREE 参数。
btree_levels	Output	NUMBER	将通过索引存储的统计数据。
leaf_blocks	Output	NUMBER	"
avg_leaf_blocks	Output	NUMBER	"
avg_data_blocks	Output	NUMBER	"
cluster_factor	Output	NUMBER	"
blocks_allocated	Output	NUMBER	"
extents_allocated	Output	NUMBER	"
distinct_values	Output	NUMBER	"

表 3

*1 - 保证各个数组元素的数值为 null 结束并且在适当的 OCI 约束参数中指定数组的各个元素的实际长度。

当执行过程 DEFAULT_NEW_INDEX_STATS 时可以产生下面的差错代码。第二列包含有关差错的描述，并且在括号中包含有关用户如何能够排除差错的建议。

-20001	用户既不是表的拥有者；也不具有表的 INDEX 特权；也不具有 CREATE_ANY_INDEX 特权。 (应当获得适当的特权。)
-20002	表不存在。(检查表名称和拥有者。)
-20003	指定列已经有索引。(表已经具有一个针对所请求的列的索引。)
-20004	列___未找到。(索引列不存在。)
-20005	非法的表拥有者。(输入的表拥有者 Oracle 帐户不存在。)
-20006	索引名称不唯一。(为虚拟索引输入一个不同的索引名称。)
-20010	在表上先执行 CLEAN_UP 。(表已经具有一个由用户创建的虚拟索引。)

表 4

输出参数会被输入到下一个要执行的过程 EXPLAIN_VIRTUAL.CREATE_VIRTUAL_INDEX 。 并且注意，用户可能会试图在用户没有访问的表空间上创建索引。

步骤 4: 创建虚拟索引

最好复制原始表的任何现有索引以定义虚拟索引。还复制索引统计数据和存储信息。因此可以执行下面过程:

```
begin
  pafo.explain_virtual.create_virtual_index
(:table_owner, :table_name, :ind_name, :ind_col_names, :num_ind_colu
mns, :default_tablespace, :init_trans, :max_trans, :pct_free, :btree_levels
, :leaf_blocks, :avg_leaf_blocks, :avg_data_blocks, :cluster_factor, :block
s_allocated, :extents_allocated, :distinct_values);
end;
```

前面在步骤 3 中定义了这些参数。

步骤 5: 最终步骤

这里，PAFO 如上所述创建一个显示初始 SQL 文本的新 WHAT IF 框架(frame)。允许用户修改文本和增加线索。复制 SQL 框架的"标签"并且使用相同的序号。例如，如果 SQL 框架标记是 SQL : HISTORY ; 1，则 WHAT IF 框架应当被标记成 WHAT IF : HISTORY;1。

当用户请求一个优化方案时， PAFO 应当改变所解释的 SQL 文本，但不是在 SQL 框架上。但是 SQL 应当看上去类似初始 SQL 语句。因而过程期望得到 SQL 文本的一个复本并且用表复本的名称(:temp_table_name)替换原始表名称(:table_name)的所有出现。

如果 SQL 语句使用一个同义词(局部或公共)指向原始表，则修改的 SQL 文本应当包含虚拟表拥有者和虚拟表名称。在线索引用表或表上一个索引的情况下也可能需要改变线索参数。接着创建方案并且将方案检索到存储器中。

对于 OPERATION 列从' TABLE '开始的各个 OBJECT_OWNER 和 OBJECT_NAME 组合，应当将临时表名称改变成初始名称(:table_name)。

对于前面步骤 2 中返回的各个虚拟索引，应当在优化方案中搜寻这些虚拟索引名称并且用原始索引名称替换。在数组(:original_index_owners, :original_index_names)中标识虚拟索引。在数组 (:virtual_index_names)中存储相关的替换索引名称。所以每当 OPERATION 列以文本" INDEX "为开始时 PAFO 便进行搜寻。如果 OBJECT_OWNER 和 OBJECT_NAME 匹配 ORIGINAL_INDEX_OWNER (0)和 VIRTUAL_INDEX_NAME (0)数值，则索引名称应当被改变成 ORIGINAL_INDEX_NAME (0)。最好针对各个数组元素重复这个处理。

接着显示修改的方案并且在创建的虚拟索引对性能有影响时通知用户。

当显示" OBJECT"标签中的信息时， PAFO 最好如步骤 4&5 中所述进行相同的改变；即 PAFO 显示 T_########1 而不是 EMPLOYEES 的索引。但在向用户显示信息时 PAFO 应当使用名称 EMPLOYEES 代替 T_########1 。并且当显示索引时，应当显示相关的 ORIGINAL_INDEX_NAME 数值。

当用户点击一个包含虚拟表名称或其某个索引的方案步骤时，应当显示相关虚拟索引的统计数据。因而，用户点击 EMPLOYEES，用户看见名称 EMPLOYEES，但 PAFO 列出 T_########1 的统计数据。对于 EMPLOYEES 的索引也是如此。

放弃一个虚拟索引

这个过程最好只能放弃那些属于一个虚拟表的索引。所以如果一个虚拟表不存在，则应当创建这个虚拟表。通过过程 CREATE_VIRTUAL_TABLE 创建虚拟表。接着用户可以指示其希望放弃一个索引的表。接着可以使用下面的步骤:

步骤 1. 列出所有用户要访问的表

```
SELECT owner, table_name
FROM exp1_tables
```

步骤 2. 确定选择的表是否虚拟表

```
SELECT table_id, new_table_name
FROM my_explain_virtual_tables
WHERE orig_table_name = :Table_name
```

如果查询返回一个结果行，则虚拟复本存在。否则创建虚拟表(参见上面创建虚拟索引中的步骤 1)。

步骤 3. 列出虚拟表上的虚拟索引

列出虚拟表上的虚拟索引(参见上面创建虚拟索引中的步骤 2)。

步骤 4. 放弃虚拟索引

为了放弃一个虚拟索引，执行下面的过程:

```
begin
pafo.explain_virtual.drop_virtual_index
(:table_owner, :table_name, :virtual_index_name)
end;
```

参数	模式	数据类型	用途
table_owner	Input	VARCHAR2(30)	拥有原始表的 ORACLE 帐户。
table_name	Input	VARCHAR2(30)	原始表的名称。
index_name	Input	VARCHAR2(30)	虚拟索引的名称; (EXPLAIN_VIRTUAL_INDEXES.NEW_INDEX_NAME)

表 5

清除

为了放弃一个具体虚拟表并且在 PAFO 注册表中清除相关信息，可以执行下面的过程:

```
begin
  pafo.explain_virtual.clean_up_table
```

(:virtual_table_creator, :original_table_owner, :original_table_name);

end;

参数	模式	数据类型	用途
virtual_table_owner	Input	VARCIEIAR2(30)	创建虚拟表的ORACLE帐户。
original_table_owner	Input	VARCHAR2(30)	拥有原始表的ORACLE帐户。
original_table_name	Input	VARCHARZ(30)	原始表的名称。

表 6

为了放弃所有虚拟表，执行下面过程:

begin

pafo.explain_virtual.clean_up_all;

end;

应当理解，上述具体实施例只是为了图解本发明的原理，本领域的技术人员在不偏离本发明的范围和宗旨的前提下可以进行各种修改。因而本发明的范围仅受后面的权利要求书的范围的限制。

说 明 书 附 图

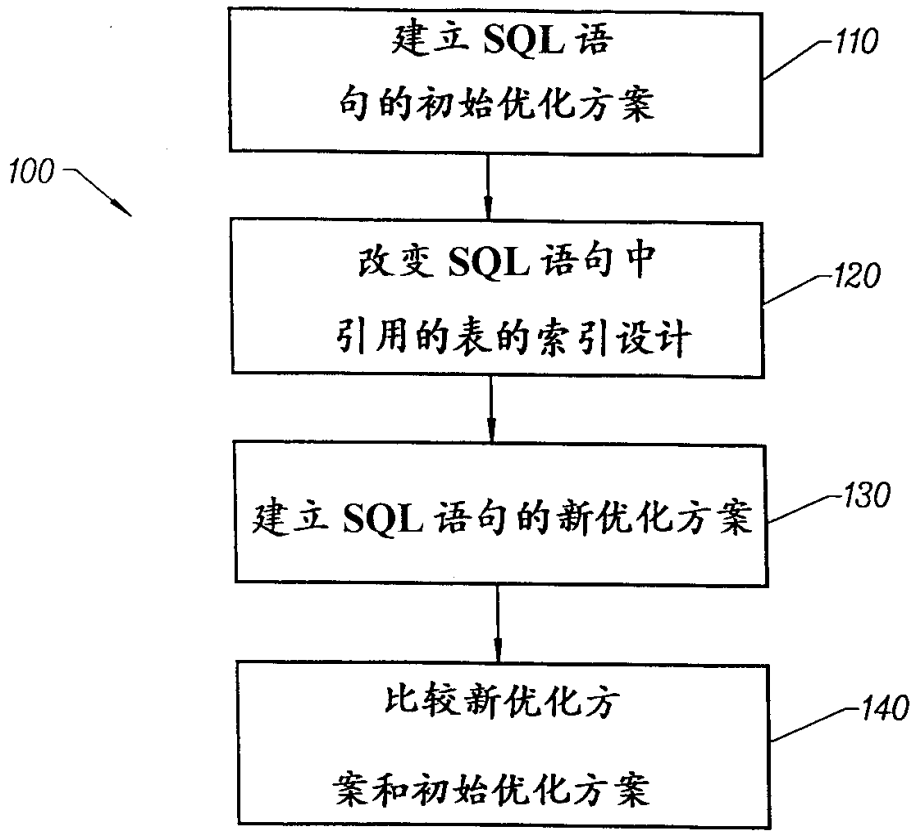


图1

现有技术

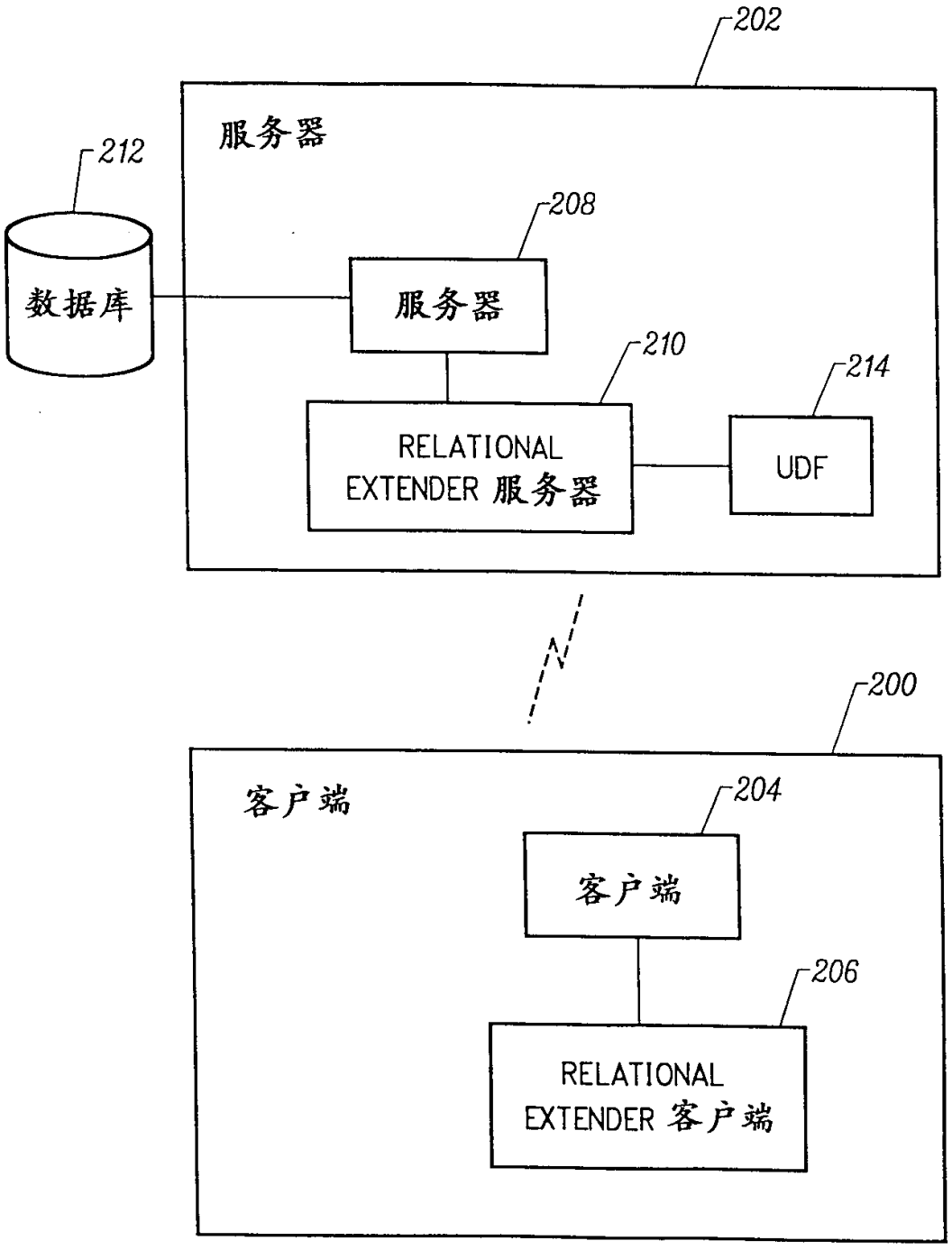
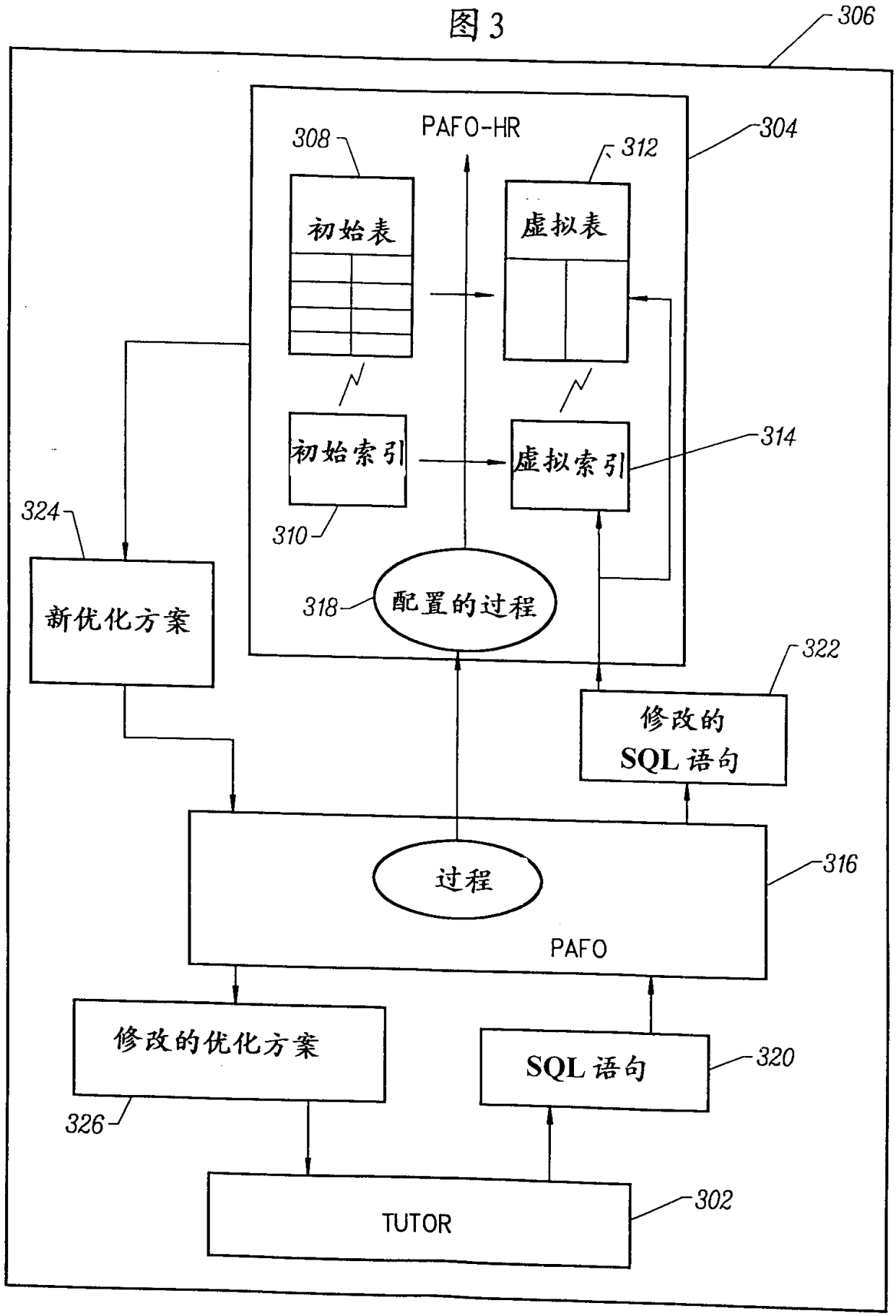

202
服务器
212
208
数据库
服务器
210
214
RELATIONAL
EXTENDER 服务器
UDF
200
客户端
204
客户端
206
RELATIONAL
EXTENDER 客户端

图2

图 3


306
PAFO-HR
308
312
304
初始表
虚拟表
初始索引
虚拟索引
314
310
324
新优化方案
318
配置的过程
322
修改的
SQL 语句
过程
316
PAFO
修改的优化方案
SQL 语句
320
326
TUTOR
302

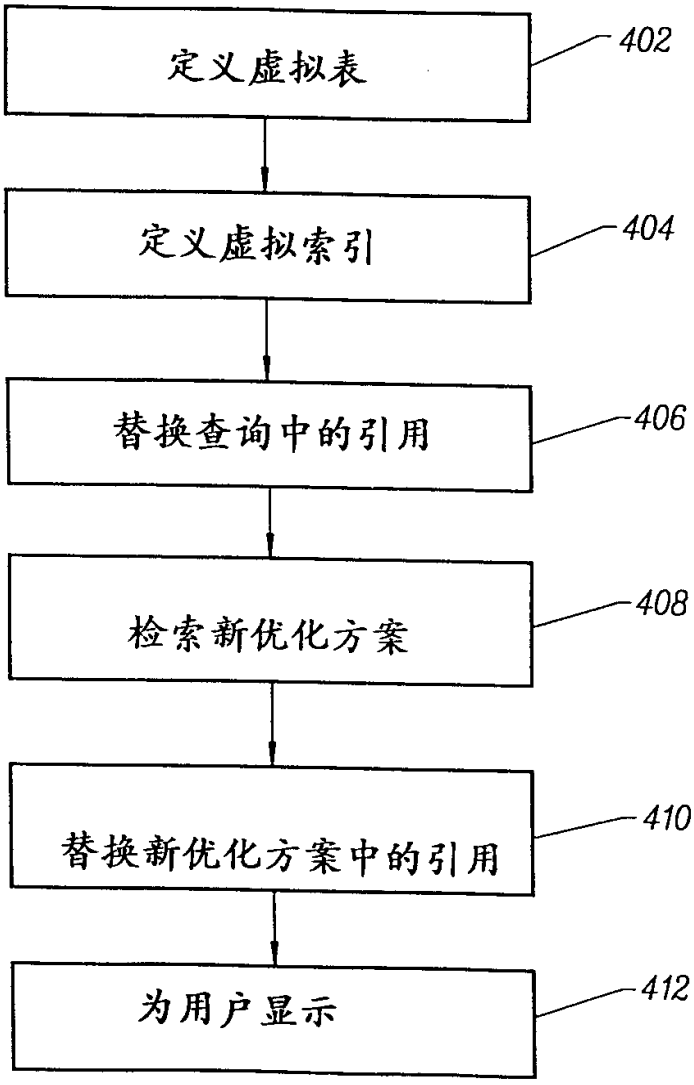

定义虚拟表
402
定义虚拟索引
404
替换查询中的引用
406
检索新优化方案
408
替换新优化方案中的引用
410
为用户显示
412

图4