(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 102687124 A
(43)申请公布日 2012.09.19

(21)申请号 201080059390.X
(22)申请日 2010.12.13
(30)优先权数据
12/646,716 2009.12.23 US
(85)PCT申请进入国家阶段日
2012.06.25
(86)PCT申请的申请数据
PCT/US2010/060154 2010.12.13
(87)PCT申请的公布数据
WO2011/078986 EN 2011.06.30
(71)申请人 伊姆西公司
地址 美国马萨诸塞州
(72)发明人 布莱恩·哈根布赫
西瓦拉姆克里斯南·纳拉亚南
威廉·C·惠普奇 弗洛里安·瓦斯
(74)专利代理机构 北京金信立方知识产权代理有限公司 11225
代理人 黄威 王智
(51)Int.Cl.
*G06F 11/30*(2006.01)

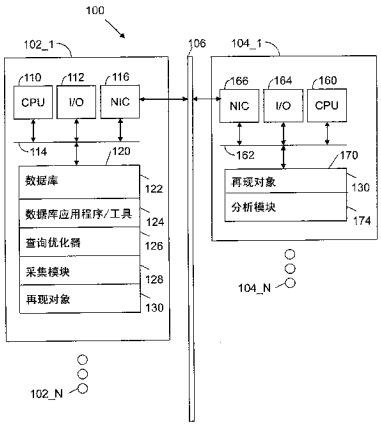
权利要求书 2 页 说明书 12 页 附图 2 页

(54)发明名称

分析查询优化器性能的设备和方法

(57)摘要

一种分析查询优化器性能的方法包括识别事件触发。构造表征事件触发时用户计算机的运行参数的再现对象。将该再现对象从用户计算机传送到测试计算机。在测试计算机上分析该再现对象来表征查询优化器的性能。




CN 102687124 A

1. 一种分析查询优化器性能的方法，包括：

识别事件触发；

在用户计算机上构造表征所述事件触发时所述用户计算机的运行参数的再现对象；

将所述再现对象从所述用户计算机传送到测试计算机；以及

在所述测试计算机上分析所述再现对象以表征查询优化器的性能。

2. 根据权利要求 1 所述的方法，其中所述识别包括识别从查询优化器崩溃、特定查询的提交和对再现对象的请求当中选择的事件触发。

3. 根据权利要求 1 所述的方法，其中所述运行参数包括输入查询、查询方案对象、数据库模式信息、数据库统计和物理系统数据。

4. 根据权利要求 1 所述的方法，其中所述分析包括分析查询优化器崩溃。

5. 根据权利要求 1 所述的方法，其中所述分析包括分析次优查询方案。

6. 根据权利要求 1 所述的方法，其中所述分析包括分析不正确的查询方案。

7. 根据权利要求 1 所述的方法，其中所述分析包括分析工作流。

8. 根据权利要求 1 所述的方法，其中所述分析包括执行假设情景分析。

9. 一种用户计算机，包括：

处理器；及

连接到所述处理器的存储器，所述存储器存储采集模块，所述采集模块包括指令，所述指令用于使所述处理器：

识别事件触发；

构造表征所述事件触发时所述用户计算机的运行参数的再现对象；以及

将所述再现对象从所述用户计算机传送到测试计算机。

10. 根据权利要求 9 所述的用户计算机，其中用于识别事件触发的可执行指令包括用于识别从查询优化器崩溃、特定查询的提交和对再现对象的请求当中选择的事件触发的可执行指令。

11. 根据权利要求 9 所述的用户计算机，其中所述运行参数包括输入查询、查询方案对象、数据库模式信息、数据库统计和物理系统数据。

12. 一种测试计算机，包括：

处理器；

连接到所述处理器的存储器，所述存储器存储分析模块，所述分析模块包括指令，所述指令使所述处理器：

分析表征事件触发时用户计算机的运行参数的再现对象，以表征所述用户计算机的查询优化器的性能。

13. 根据权利要求 12 所述的测试计算机，其中所述运行参数包括输入查询、查询方案对象、数据库模式信息、数据库统计和物理系统数据。

14. 根据权利要求 12 所述的测试计算机，其中用于分析的可执行指令包括用于分析查询优化器崩溃的可执行指令。

15. 根据权利要求 12 所述的测试计算机，其中用于分析的可执行指令包括用于分析次优查询方案的可执行指令。

16. 根据权利要求 12 所述的测试计算机，其中用于分析的可执行指令包括用于分析不

正确的查询方案的可执行指令。

17. 根据权利要求 12 所述的测试计算机，其中用于分析的可执行指令包括用于分析工作流的可执行指令。

18. 根据权利要求 12 所述的测试计算机，其中用于分析的可执行指令包括用于分析假设情景的可执行指令。

# 分析查询优化器性能的设备和方法

## 技术领域

[0001] 本发明涉及数字数据处理。更具体地，本发明涉及分析查询优化器的性能。

## 背景技术

[0002] 查询优化包括将数据库查询转化为在存储于数据库中的数据上执行的高效的程序或者查询方案。数据库查询通常以查询语言表述，如结构化查询语言 (SQL)、通用查询语言 (CQL) 或者多维表达式 (MDX) 等等，数据库查询被转化为一种或多种可行的查询方案。查询方案列明了一组步骤，用来访问或者修改与查询有关的数据。诸如如何访问给定的数据关系、以何种顺序连接数据关系、排序顺序等详情都有可能形成查询方案的一部分。

[0003] 对于给定的查询，查询方案的各个不同的组成部分，如访问路径、连接方法和排序顺序，可以产生很多查询方案。典型的数据仓库查询可以形成数亿种可行的方案。可以根据不同的参数，包括例如访问磁盘的次数和取回数据所需的响应时间，来建立查询方案的成本模型。查询优化器可估计已给定查询的所有可行的查询方案的成本，并确定最佳方案，也就是执行该查询的最有效的方案。

[0004] 一般来说，用户不能直接访问查询优化器。因此，用户看不到优化过程。然而，用户能识别任意多个的情形中的次优性能，包括优化器进程崩溃、优化器产生的查询方案产生错误的结果、低效率的查询方案或者相较于系统改造前的系统操作的次等方案。不能访问查询优化器限制了用户对在不同情况下方案在何处产生的假设分析能力。

[0005] 评估查询优化器的性能包括对多个参数的理解。通常，查询优化器的用户不愿意让查询优化器的供应商看见自己的系统，尤其是用户的数据。即使能够访问数据，获取正确的数据也很难。此外，用户通常也不愿允许任何形式的评估干扰正在进行的业务流程。所以，尽管需要了解所使用的系统中查询优化器的性能，但是这么做的机会很有限。因此，就需要提供技术来评估所部署的查询优化器的性能。

## 发明内容

[0006] 分析查询优化器的性能的方法包括识别事件触发。构造表征事件触发时用户计算机的运行参数的再现对象。将该再现对象从用户计算机传输到测试计算机。在测试计算机中分析该再现对象以表征查询优化器的性能。

[0007] 用户计算机具有连接到处理器的存储器。该存储器存储采集模块，该采集模块包括指令，该指令使处理器识别事件触发，构造表征事件触发时用户计算机的运行参数的再现对象，并将再现对象从用户计算机传输到测试计算机。

[0008] 测试计算机具有连接到处理器的存储器。该存储器存储包括指令的分析模块，以使处理器分析表征事件触发时用户计算机的运行参数的再现对象，以表征查询优化器的性能。

## 附图说明

[0009] 通过以下结合附图的详细描述能够更充分地理解本发明，其中：

[0010] 图 1 示出根据本发明一实施例配置的计算机系统。

[0011] 图 2 示出根据本发明一实施例配置的再现对象。

[0012] 图 3 示出与本发明一实施例相关的运行过程。

[0013] 在所有图中相同的参考标号表示对应的部分。

## 具体实施方式

[0014] 图 1 示出根据本发明一实施例配置的系统 100。系统 100 包括用通信信道 106 连接起来的用户计算机 102-1 和测试计算机 104-1，该通信信道可以是有线或者无线信道。用户计算机 102-1 包括标准构件，例如通过总线 114 连接的中央处理单元 110 和输入 / 输出设备 112。所述输入 / 输出设备可以包括键盘、鼠标、显示器和打印机等等。总线 114 还连接到网络接口卡 116，网络接口卡 116 提供与其他计算机如计算机 104-1 的连接。存储器 120 也连接到总线 114。该存储器存储数据和可执行模块，以执行本发明的操作。特别地，存储器 120 存储数据库 122 和一组相关的数据库应用程序或者工具 124。数据库应用程序或者工具 124 用于构建数据库查询。每个数据库查询被施加到查询优化器 126，所述查询优化器生成查询方案，该查询方案被施加到数据库 122 以产生数据结果。这些构件是现有技术中已知的。

[0015] 本发明集中在采集模块 128 和再现对象 130 上。采集模块 128 包括可执行指令，用于在预定情形下采集查询、数据库和系统信息。所述预定情形可以包括优化过程崩溃、被跟踪的优化过程和假设分析等等。采集模块 128 将该查询、数据库和系统信息加载到再现对象 130 中。然后再现对象 130 会被传送到测试计算机 104-1。所以，采集模块 128 可以包括用来加载和传送再现对象 130 的可执行的指令。

[0016] 图 1 示出用户计算机 102-1 至 102-N。在典型的实施例中，存储器 120 中的数据库 122 和其他模块分布在多个机器中。类似地，测试计算机 104-1 可以是单个机器或者它的功能可以分布在多个机器 104-1 至 104-N 中。

[0017] 测试计算机 104-1 包括标准构件，该标准构件包括通过总线 162 连接到输入 / 输出设备 164 和中央处理单元 160 的网络接口卡 166。存储器 170 也连接到总线 162。存储器 170 存储从用户计算机 102-1 接收到的再现对象 130。此外，存储器 170 还存储分析模块 174，该分析模块包括用来分析再现对象 130 的可执行指令。可以将任意数目的分析编码为分析模块 174，下面提供其例子。

[0018] 因此，本发明提供一种分析查询优化器性能的技术。优势在于，所述分析的执行是与用户计算机分开的，因此不会中断所使用的用户计算机的运行。此外，配置再现对象来提供用户愿意分享的系统信息。另有优点是，再现对象不需要包含实际用户数据。再现对象的分析与所使用的用户系统分开执行。分析的结果可以和推荐的软件修复一起发送给用户。特别地，分析模块 174 可将查询优化器报告反馈给用户。在某些情况下，分析模块 174 提供软件补丁和 / 或推荐的设置以提高系统性能。

[0019] 图 2 示出再现对象 130 的一个实施例。再现对象 130 包括输入查询 202，其是查询优化处理的主体。输入查询用于从数据源中检索数据的正式表达。

[0020] 再现对象 130 还包括查询方案对象 204，所述查询方案对象是应用于数据库 122 以

执行指定的输入查询 202 的查询方案。也就是说，该查询方案是针对数据库的输入查询的最佳执行的一组操作。

[0021] 再现对象 130 还包括数据库模式信息 206。数据库模式至少包括关系数据库的各个表的描述。

[0022] 再现对象 130 还包括数据库统计 208。数据库统计是表征数据库内的条目的指标。最后，再现对象 130 包括系统数据 210。系统数据 210 包括表征处理查询方案对象的物理机器和物理机器的性能的指标。所述数据可以包括机器的数目、可用的存储器和散列集合等等。

[0023] 图 3 示出与本发明一实施例相关的运行过程。在 300，监视事件触发。特别地，采集模块 128 监视事件触发。事件触发可能是指定的查询、假设情景分析、系统崩溃、对再现对象的请求或任何其他预定事件的提交。在 302，响应于该事件触发，构造再现对象。然后在 304，传送该再现对象。例如，将再现对象 130 从用户计算机 102-1 传送到测试计算机 104-1。然后在 306，分析再现对象。例如，测试计算机 104-1 的分析模块 174 可以被用于分析再现对象 130。

[0024] 在一个实施例中，再现对象 130 是根据称为数据交换语言 (DXL) 的语法定义的以可扩展标记语言 (XML) 编码的对象。在附录中提供了此种再现对象的标有注释的例子。根据本发明的实施例也可以使用其他编码。

[0025] 除了图 2 中所示的构件以外，再现对象还可包括上下文信息，比如配置参数、与内部优化步骤的顺序和结果有关的追踪信息（如定时信息）和相关信息。

[0026] 优选地，针对可扩展性优化该再现对象以发展多个版本。非常可取的格式是详细的、人可读的，并且支持标准查询语言，如 Xpath 或者 Xquery，以便于统计分析。

[0027] 再现对象格式的优选性能包括：版本控制，以明确识别软件和协议的版本；压缩，以通过邮件或其他介质传送；加密；人可读；适于利用标准 XML 编辑器分析；适于通过标准 XML 查询工具查询；以及易于在支持存储 XML 的任何数据库中归档。

[0028] 一旦触发，采集模块 128 将询问优化器的所有软件构件，并请求将所有相关数据序列化。例如，查询优化器 126 可包括优化上下文信息、元数据缓存、语法分析程序和搜索模块，询问每个构件均的信息。在一个实施例中，所有组件使用标准抽象 API 连接在一起；每个构件独立地判定什么信息是相关的。理想地，任何新加的构件都提供回调函数。优选地，采集模块 128 不干扰实际查询的执行，也就是说，好像采集模块没有运行一样执行或中断查询。

[0029] 在一个实施例中，通过例如作为数据访问协议的一部分的 SQL 或者其他通用配置机制来显示用于生成再现对象的句法。例如，作为一种开放源码的对象关系数据库管理系统的 PostgreSQL(Postges) 支持通过如下句法改变配置：

[0030] SET EXPLAIN_DXL = on

[0031] 这中开启所有后续语句的再现对象，直到配置改变为：

[0032] SET EXPLAIN_DXL = off

[0033] 该配置改变独立于工作负载，也就是说，实际工作负载的句法不需要被改变。

[0034] 由内部事件（例如崩溃）触发的再现对象的产生可以被下列语句调用：

[0035] SET EXPLAIN_DXL = on

[0036] <offending statement>

[0037] SET EXPLAIN_DXL = off.

[0038] 该再现对象格式便于将最后得到的结构存储 / 记录在标准记录设施中，例如数据中心监控基础设施。使用这些外部设施、额外的后处理如将再现对象自动发送到测试计算机 104-1（如供应商的支持部门）变得可能。

[0039] 分析模块 174 可以作为匹配或相似软件版本的独立优化过程来实现。载入机构读取再现对象并从再现对象获取的地方开始重演优化。通过向语法分析程序发送原始查询表达来开始该重演。可选地，可以利用来自再现对象的信息，例如优化上下文和元数据缓存信息，来准备优化器中的构件。在优化重演期间，不同的优化器构件可使用标准回调机制来请求额外的信息，标准回调机制能够从其它数据库处理导入该信息，如优化上下文信息或元数据。载入器截取这些请求并以从再现对象提取的信息来响应。

[0040] 该截取机制将正在重演的优化与其它数据库处理断开。这就使得不需要访问数据库系统就可以孤立地重演优化，包括在不同硬件平台上重演（不同硬件配置、不同的操作系统等等）。

[0041] 分析模块 174 可包括用于支持许多再现对象分析的可执行指令。例如，分析模块 174 可以包括分析查询优化器崩溃的可执行指令。在这种情况下，优化器在 SQL 语句优化期间由于软件缺陷而崩溃。内部错误处理机制开始运行并自动激活再现对象的采集和导出。再现对象被发送至测试计算机 104_1。最终得到的再现对象可被用于在相同软件版本的优化器的单独副本上再现该事件。该问题能够被解决而不需要工程师访问生产系统，也不需要停机时间。工程部门开发出修复，并且在最终传送给客户之前检验该修复。在一个实施例中，再现对象提供一种完整的并且完全自主的退化测试，该退化测试能够被编入工程部门的日常退化测试中。

[0042] 分析模块 174 还可以被编码用来评定次优的查询方案。在这种情形下，用户注意意外的查询行为，例如，意外地执行缓慢，这可能因为是次优方案。理解这种行为变化是非琐细的并且通常不表明是软件缺陷，而仅是由于用户数据中的重大变化。用户使用句法扩展生成再现对象。该再现对象被传送至测试计算机 104-1 中的分析模块 174。分析模块 174 能够评估感知到的缺陷。

[0043] 分析模块 174 也能够用可执行指令编码来评估不正确的方案。在这种情形下，用户在执行特定查询时候注意到错误结果。用户使用句法扩展来明确地请求再现对象。该再现对象被传送至测试计算机 104-1 中的分析模块 174。分析模块 174 研究原因并辨别是优化器存在缺陷还是由于正确的查询方案的执行错误导致错误结果。在优化器存在缺陷的情况下，可以调用上面讨论的优化器崩溃工作流。在执行错误的情况下，可以应用传统的故障排除技术。

[0044] 分析模块 174 也可被执行用于支持工作流监视和记录。典型的工作流由每隔一段时间运行的几百个查询组成。通常认为“重要”查询的一个小子集是性能关键。在调整该系统之后，用户为该关键查询生成再现对象并将其归档（例如，将它们存储到数据库中）。在数据库软件升级之后发生方案退化（查询的性能降低）的情况下，用户可以将一新的再现对象和归档的再现对象进行比较，并确定性能降低是否归因于软件退化。

[0045] 分析模块 174 也可被执行来支持假设分析。为了调整的目的，理解哪个数据库对

象以何种方式和频率使用是非常有帮助的。原始的查询文本包含对数据库对象如表格的引用，但是优化器根据附加的对象引用如索引来做决定。为了正确调整系统，需要理解对这些对象的准确应用。用户可以记录工作流并将最终得到的再现对象归档到数据库中。之后，数据可被查询来确定对象的某些特征。实例包括：根据索引的有用性将索引分级，确定等同性谓词中那些列被频繁使用以产生它们的索引，以及基于在记录的工作流上的查询识别调整机会。也可通过改变环境设置，比如添加存储器、添加机器等，来执行假设情景分析。

[0046] 本领域技术人员能够得知本发明的很多相关优点。再现对象允许“重演”问题并对其分析。例如，当识别出软件缺陷时，再现对象能够被用来开发工作区作为临时措施或者被用来帮助开发优化器代码的修复并且在发布给客户之前验证该修复。一旦开发出修复，再现对象就获取关键测试案例用于退化测试。再现对象是自主的，并且在没有人工干预的情况下生成，因而消除了管理上的错误。如果优化器为系统请求额外的元数据，那么在优化期间再现对象可能增大。尽管再现对象一般是响应于触发的事件而产生的，但是也可以综合生成再现对象与产生测试案例。再现对象可被存储和分析以用于统计目的，例如，用于检测查询或模式特性与崩溃或次优结果之间的相互关系。

[0047] 错误报告提供当系统故障或者崩溃时的瞬态信息。这种报告可以满足简单处理的需要，例如在单个机器上运行的单个应用程序。另一方面，这种工具对复杂的应用程序或者在多个机器上运行的应用程序没有太大帮助。在这种情境下，需要随时跟踪各种系统参数。如下所示的示例性的再现对象，不仅仅是在系统崩溃时，而是随时跟踪系统参数。因此，该再现对象不仅用于故障检测，也适用于优化运行（也就是不包含系统崩溃，而是结果似乎是次优的）。

[0048] 本发明的一实施例涉及具有计算机可读存储介质的计算机存储产品，该介质上具有用于进行各种计算机执行的操作的计算机代码。该媒介和计算机代码可以是特别为本发明的目的设计和构造的，或者可以是在计算机软件领域的技术人员广泛知晓和可用的。例如计算机可读介质的例子包括但不限于：磁性介质，如硬盘、软盘和磁带；光学介质，如CD-ROM、DVD 和全息装置；磁光介质；专门配置用来存储和执行程序代码的硬件装置，如专用集成电路（“ASIC”）、可编程逻辑器件（“PLD”）、ROM 和 RAM 装置。计算机代码的例子包括机器代码，如由编译器产生的，以及由计算机使用解释器来执行的包含高级代码的文件。例如，本发明一实施例可用 JAVA®、C++、其他面向对象编程语言和开发工具来实现。本发明的另一实施例可以使用硬连线电路代替或者结合机器可执行的软件指令来实现。

[0049] 为了解释的目的，前面的描述使用了特定的术语以提供对本发明的全面理解。然而，对于本领域技术人员来讲很明显，不需要具体的细节就能够实现本发明。因此，前面对本发明的具体实施例的描述是为了说明和描述的目的而给出的。它们不意图是详尽的或者将本发明限制于所公开的精确形式；显然，鉴于上述教导，很多修改和变化都是可能的。这些实施例被选定和描述是为了最好地解释本发明的原理及其实际应用，因此它们使本领域技术人员能够最好地应用本发明以及具有各种修改的各种实施例，以适合于特定的预期应用。所附权利要求及其等价定义意图限定出本发明的范围。

[0050] 附录

[0051]

<r:optimizer-repro version="1.0" database-software-version="3.4.0" xmlns:r="http://greenplum.com/dxl/2010/03/">

<!--**********

Input SQL object; the repro object captures the original input query, as shown below

**********-->

<r:sql>

select length(c2.name), sum(sales.quantity)

from customer as c2 inner join sales on sales.customer_id = c2.id

where c2.name like 'Bob%' and

sales.saledate > '2008-03-04' and

sales.saledate < '2009-01-01'

</r:sql>

<!--**********

Query object; the repro object captures the query plan selected by the query optimizer, as shown below

**********-->

<r:query>

<r:project relation-name="result">

<r:projection-columns>

<r:colref target="gb.name_length"/>

<r:colref target="gb.quantity"/>

</r:projection-columns>

<r:group-by relation-name="gb">

[0052]

```
<r:group-by-columns>
  <r:function-call attribute-name="name_length" name="length" output-
type="int4">
    <r:colref target="customer.name"/>
  </r:function-call>
</r:group-by-columns>
<r:aggregate-columns>
  <r:sum attribute-name="quantity">
    <r:colref target="sales.quantity"/>
  </r:sum>
</r:aggregate-columns>
<r:inner-join relation-name="ij">
  <r:left>
    <r:subquery relation-name="c2" alias="c2">
      <r:get table-name="customer" relation-name="customer">
        <r:columns>
          <r:column name="name" attribute-name="name"/>
          <r:column name="id" attribute-name="id"/>
          <r:column name="birthdate" attribute-name="birthdate"/>
        </r:columns>
        <r:filter>
          <r:like>
            <r:colref target="customer.name"/>
            <r:constant type="varchar" value="Bob%"/>
          </r:like>
        </r:filter>
      </r:get>
    </r:subquery>
  </r:left>
  <r:right>
```

[0053]

```
<r:get table-name="sales" relation-name="sales" table-alias="sales">
  <r:columns>
    <r:column name="customer_id" attribute-name="customer_id"/>
    <r:column name="quantity" attribute-name="quantity"/>
    <r:column name="saledate" attribute-name="saledate"/>
  </r:columns>
  <r:filter>
    <r:and>
      <r:greater-than>
        <r:colref target="sales.saledate"/>
        <r:constant type="date" value="2008-03-04"/>
      </r:greater-than>
      <r:less-than>
        <r:colref target="sales.saledate"/>
        <r:constant type="date" value="2009-01-01"/>
      </r:less-than>
    </r:and>
  </r:filter>
</r:get>
</r:right>
<r:join-expression>
  <r:equals>
    <r:colref target="sales.customer_id"/>
    <r:colref target="c2.id"/>
  </r:equals>
</r:join-expression>
</r:inner-join>
</r:group-by>
</r:project>
</r:query>
```

[0054]

```
<!--**********
```

Schema data; the repro object collects information on the database schema, including, in this example, table names, column names and column attributes

```
**********-->

<r:schema>
  <r:table name="customer">
    <r:columns>
      <r:column name="id" type="int4" is-nullable="false" />
      <r:column name="name" type="varchar(100)" is-nullable="false" />
      <r:column name="birthdate" type="date" is-nullable="true" />
      <r:column name="address_street_1" type="varchar(200)" is-nullable="true" />
      <r:column name="address_street_2" type="varchar(200)" is-nullable="true" />
      <r:column name="address_city" type="varchar(100)" is-nullable="true" />
      <r:column name="address_zip" type="varchar(10)" is-nullable="true" />
    </r:columns>
  </r:table>
  <r:table table-name="sales">
    <r:columns>
      <r:column name="customer_id" type="int4" is-nullable="false" />
      <r:column name="quantity" type="int2" is-nullable="false" />
      <r:column name="saledate" type="date" is-nullable="false" />
    </r:columns>
  </r:table>
</r:schema>
```

[0055]

<!--**********

Statistics data; the repro object characterizes database statistics or metrics characterizing entries within a relational database, including, in this example, such metrics as table name, number of rows, average row width, number of values, etc.; observe that statistics are gathered over periods of time (histogram-buckets) and therefore can be used to evaluate a set of circumstances preceding a trigger event

**********-->

<r:statistics>

<r:table-statistics name="customer" num-rows="444444" average-row-width="450"/>

<r:table-statistics name="sales" num-rows="12345678" average-row-width="40"/>

<r:column-statistics name="customer" column="name" num-distinct-values="123456">

<r:histogram>

<r:histogram-bucket start="Aardvark" end="Frank" estimated-num-values="98592"/>

<r:histogram-bucket start="Frank" end="Moe" estimated-num-values="105134"/>

<r:histogram-bucket start="Moe" end="Roger" estimated-num-values="99287"/>

<r:histogram-bucket start="Roger" end="Zydrunas" estimated-num-values="99783"/>

</r:histogram>

</r:column-statistics>

<r:column-statistics name="sales" column="saledate" num-distinct-values="2190">

<r:histogram>

[0056]

<r:histogram-bucket start="2000-01-01" end="2000-07-01" estimated-num-values="1010101"/>

<r:histogram-bucket start="2000-07-01" end="2001-01-01" estimated-num-values="1143948"/>

<r:histogram-bucket start="2001-01-01" end="2001-07-01" estimated-num-values="991235"/>

<r:histogram-bucket start="2001-07-01" end="2002-01-01" estimated-num-values="1000000"/>

<r:histogram-bucket start="2002-01-01" end="2002-07-01" estimated-num-values="991235"/>

<r:histogram-bucket start="2002-07-01" end="2003-01-01" estimated-num-values="1008348"/>

<r:histogram-bucket start="2003-01-01" end="2003-07-01" estimated-num-values="1093589"/>

<r:histogram-bucket start="2003-07-01" end="2004-01-01" estimated-num-values="990213"/>

<r:histogram-bucket start="2004-01-01" end="2004-07-01" estimated-num-values="998965"/>

<r:histogram-bucket start="2004-07-01" end="2005-01-01" estimated-num-values="996352"/>

<r:histogram-bucket start="2005-01-01" end="2005-07-01" estimated-num-values="1103847"/>

<r:histogram-bucket start="2005-07-01" end="2006-01-01" estimated-num-values="993544"/>

</r:histogram>

</r:column-statistics>

</r:statistics>

<!--**********

[0057]

Environment settings; the repro object collects system data or metrics characterizing the physical machines and the properties of the physical machines that process a query plan object; in this example, the repro object characterizes the number of machines, the available memory and Boolean variables such as hash aggregation, index scan and nested loop join

**********-->

```
<r:environment>
    <r:setting name="number-machines" value="15"/>
    <r:setting name="work-memory-available" value="32MB"/>
    <r:setting name="enable-hash-aggregation" value="true"/>
    <r:setting name="enable-index-scan" value="false"/>
    <r:setting name="enable-nested-loop-join" value="true"/>
  </r:environment>

</r:optimizer-repro>
```

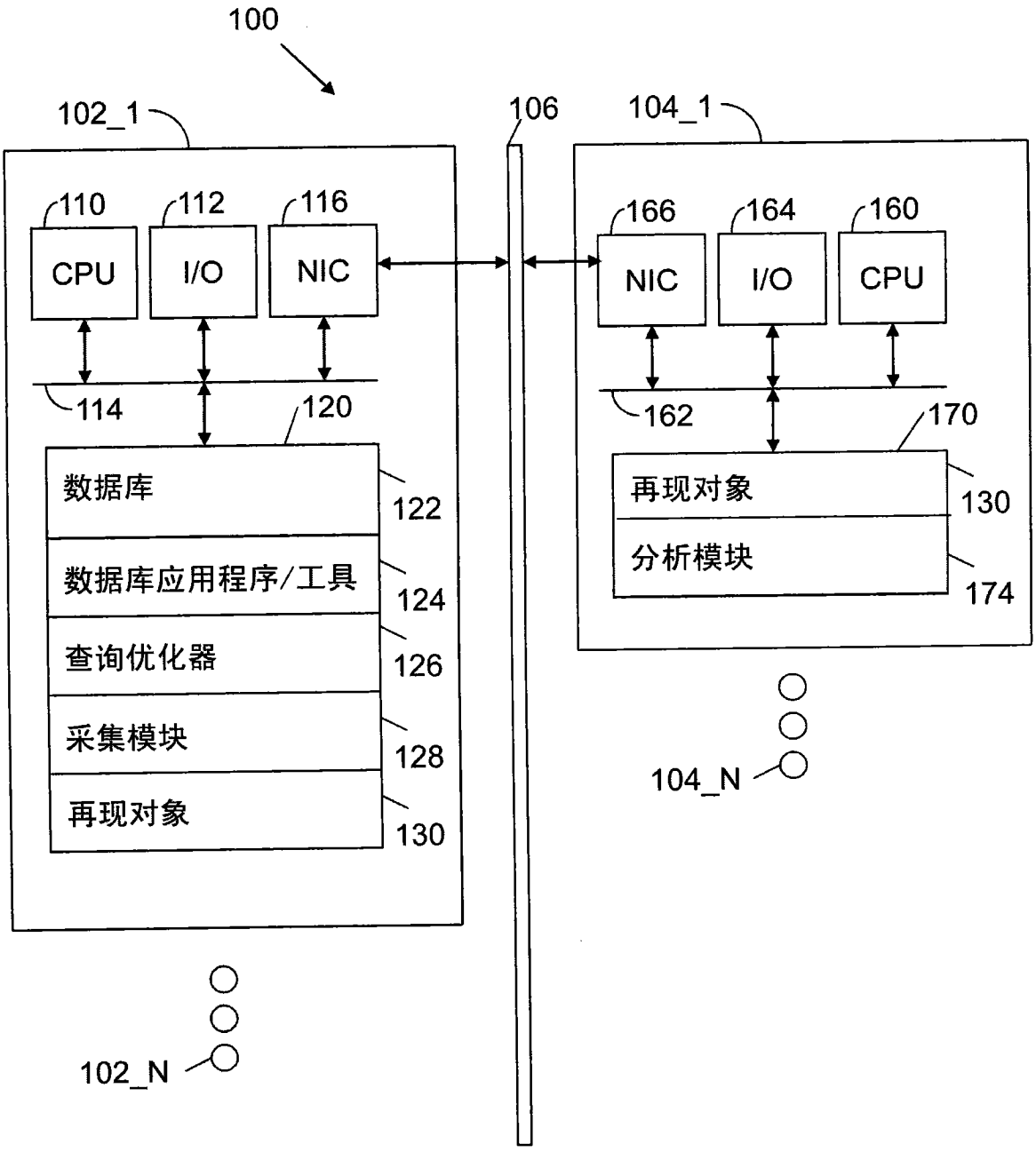

100
102_1
106
104_1
110
112
116
CPU
I/O
NIC
166
164
160
NIC
I/O
CPU
114
120
162
170
数据库
122
数据库应用程序/工具
124
查询优化器
126
采集模块
128
再现对象
130
再现对象
130
分析模块
174
104_N
102_N

图 1

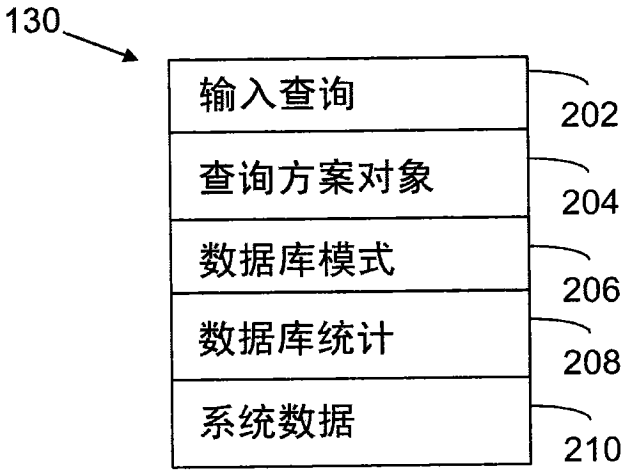

130
输入查询
202
查询方案对象
204
数据库模式
206
数据库统计
208
系统数据
210

图 2

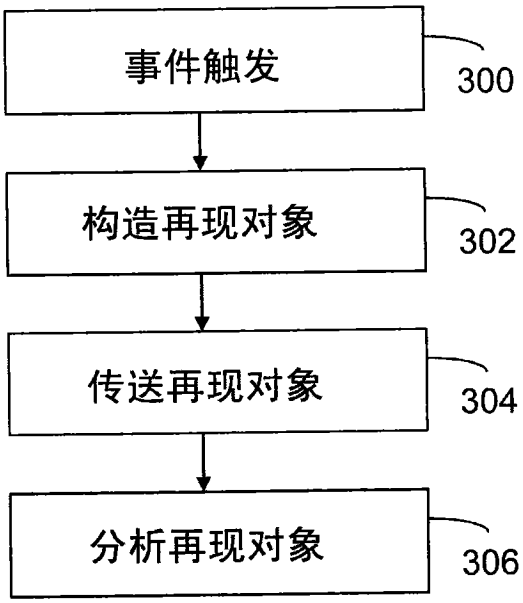

事件触发
300
构造再现对象
302
传送再现对象
304
分析再现对象
306

图 3