(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 105989194 B
(45)授权公告日 2019.12.24

(21)申请号 201610160512.3

(22)申请日 2016.03.21

(65)同一申请的已公布的文献号
申请公布号 CN 105989194 A

(43)申请公布日 2016.10.05

(30)优先权数据
14/663,819 2015.03.20 US

(73)专利权人 国际商业机器公司
地址 美国纽约阿芒克

(72)发明人 S.布尔诺奈斯 Y.O.劳 李潇 闵红 J.威伦加 周祥为

(74)专利代理机构 北京市柳沈律师事务所 11105
代理人 张晓明

(51)Int.Cl.
G06F 16/27(2019.01)

(56)对比文件
CN 102354292 A,2012.02.15,
US 5802528 A,1998.09.01,
CN 103391311 A,2013.11.13,
CN 103995854 A,2014.08.20,
CN 103379139 A,2013.10.30,
CN 102354292 A,2012.02.15,

审查员 解欣

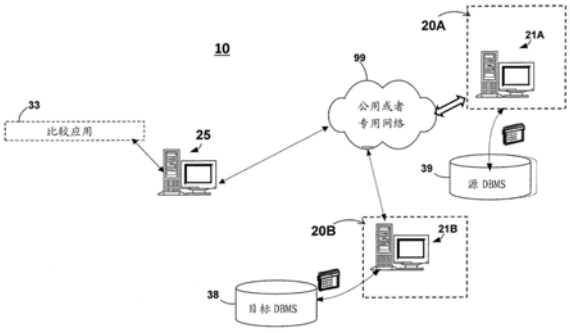
权利要求书4页 说明书23页 附图15页

(54)发明名称

表数据比较的方法和系统

(57)摘要

公开了一种用于表数据比较的系统和方法，该系统和方法使得表具有高准确性(高置信度)，其中一个表是另一个表的拷贝，通过复制同步保持该拷贝。该方法利用具有抽样的、基于统计结果的或者基于物化查询表的方法进行数据库比较。该方法首先识别包括源数据库表的数据的行的子集的块和目标数据库表中的相应块，并且获得与每个块关联的统计结果值。然后，比较相应源块和目标块的统计结果值，并且基于比较结果，确定源数据库与目标数据库的一致性评估。又一种方法使得能够在正在比较识别的块的时候，以考虑到对初始源数据库和目标数据库的实时数据修改的方式，将数据确定为持久性的或者不是持久性的。



CN 105989194 B

1.一种表数据比较的方法,包括:

识别包括源数据库表的数据的行的子集的块和包括目标数据库表的数据的行的子集的相应块;

获得与包含在识别的源数据库表的数据的行的子集的块中的数据关联的统计结果值,并且获得包含在目标数据库表的数据的行的子集的相应块中的数据的又一个统计结果值;

比较统计结果值,以确定匹配结果;以及

基于所述比较的结果,确定每个源数据库表和目标数据库表的块是否一致,其中,

在确定了不匹配比较结果时,提取每个所述识别的源数据库表的数据的行的子集的块和目标数据库表的数据的行的子集的相应块,并且还对所述识别的源数据库表的数据的行的子集的块和目标数据库表的数据的行的子集的相应块中的行的逐行比较累计数据进行比较,以进行一致性确定;或者

在确定匹配比较结果时,避免为了对所述识别的源数据库表的数据的行的子集的块和目标数据库表的数据的行的子集的相应块执行逐行比较从所述块提取所述数据;

其中所述源数据库表和所述目标数据库表包括多个块,并且在确定匹配比较结果时,

从所述源数据库表和目标数据库表识别又一个块,并且对所述又一个块重复所述获得、比较和一致性确定,以及

对多个块重复所述识别、获得、比较和一致性确定。

2.根据权利要求1所述的方法,其中所述识别块包括识别物化查询表图,所述物化查询表图定义形成在先应用用户数据库查询操作的结果的行的子集。

3.根据权利要求2所述的方法,其中所述识别块包括识别通过抽样操作选择的所述源数据库表的行。

4.根据权利要求3所述的方法,其中所述抽样操作采用随机选择函数。

5.根据权利要求1所述的方法,其中所述获得统计结果值包括如下中的一个或者多个:

采用处理收集统计结果,统计结果包括:所述块或者表的行计数值、所述块或者表的列计数值、所述块或者表的平均行长度值、统计结果的行计数和直方图、键行计数值、对所述块或者表的一个或者多个列产生的规定累计值、对所述块或者表的一个或者多个列产生的规定平均值、对所述块或者表的一个或者多个列产生的规定标准偏差值;以及

采用用户定义函数处理所述块中的所述数据,以计算统计结果值。

6.根据权利要求1所述的方法,在所述目标数据库表包含由所述源数据库表复制的数据的数据库环境下运行,所述方法还包括:

在第一时间点,确定所述源数据库表中的所述识别块的行与所述目标数据库表中的所述相应块的所述行之间的第一组差异;

在所述第一时间点后至少复制等待时间间隔之后的第二时间点,确定所述源数据库表中的所述块的行与目标数据库表中的所述块的所述行之间的第二组差异,

基于不同时间点的所述第一组差异和第二组差异,将所述源数据库表的所述识别块与所述目标数据库表的相应块之间的差异确定为是持久性类型、非持久性类型、暂停类型或者未知类型中的一个。

7.根据权利要求6所述的方法,还包括:

确定进入操作的未提交读出UR隔离级别模式,使得查询数据避免与一个或者多个用户

应用争用锁,以及

如果确定进入所述UR模式,并且当前时间在复制等待时间间隔之后,则

再比较所述源数据库表和目标数据库表的相应块中的相应行的数据的所述累计数据值,并且如果确定所述行的所述再比较的累计数据值匹配,则所述差异类型不是持久性的,否则,

如果确定所述再比较累计数据值差异为不匹配,则

比较当前确定差异结果和从在先等待时间间隔发生的所述行的差异确定结果,并且如果确定所述当前差异结果与在先差异结果之间的所述比较差异匹配,则

确定是否在使共享锁布置于提取行上并且使另一个处理将共享锁布置于同一个行上的操作的游标稳定性CS隔离级别模式下评估了所述行的所述差异结果,并且如果未在操作的所述CS级别模式下操作,则确定所述差异类型是未知的;

并且如果确定所述当前差异结果与在先差异结果之间的所述比较差异匹配,则确定所述差异类型是持久性的。

8.根据权利要求7所述的方法,其中如果确定所述行的所述比较的当前确定差异结果和所述在先差异确定结果不匹配,则:

在所述UR模式下,将对所述源数据库表和目标数据库表中的块的相应行重复所述再比较所述累计数据值规定的迭代次数,直到检测到匹配状况;并且如果在同一个迭代中未检测到匹配状况,则:

对所述行重复所述比较所述当前确定差异结果和所述在先差异确定结果;以及

如果在达到所述迭代次数之后,保持所述不匹配状况差异结果,则确定所述持久性类型是暂停的。

9.一种用于表数据比较的系统,包括:

存储器,配置所述存储器,以存储从源数据库表和目标数据库表收到的数据;

处理器,所述处理器与所述存储器通信,配置所述处理器以执行方法,从而:

识别包括源数据库表的数据的行的子集的块和包括目标数据库表的数据的行的子集的相应块;

获得与包含在识别的源数据库表的数据的行的子集的块中的数据关联的统计结果值,并且获得包含在目标数据库表的数据的行的子集的相应块中的数据的又一个统计结果值;

比较统计结果值,以确定匹配结果;以及

基于所述比较的结果,确定每个源数据库表和目标数据库表的所述块是否一致,其中在确定了不匹配比较结果时,配置所述处理器设备,以提取每个所述识别的源数据库表的数据的行的子集的块和目标数据库表的数据的行的子集的相应块,并且还对所述识别的源数据库表的数据的行的子集的块和目标数据库表的数据的行的子集的相应块中的行的逐行比较累计数据进行比较,以进行一致性确定;或者

在确定匹配比较结果时,避免为了对所述识别的源数据库表的数据的行的子集的块和目标数据库表的数据的行的子集的相应块执行逐行比较,从所述块提取所述数据;

其中所述源数据库表和所述目标数据库表包括多个块,并且在确定匹配比较结果时,进一步配置所述处理器设备,以从所述源数据库表和目标数据库表识别又一个块,并且对所述又一个块重复所述获得、比较和一致性确定,以及

对多个块重复所述识别、获得、比较和一致性确定。

10.根据权利要求9所述的系统，其中块包括物化查询表图，所述物化查询表图定义形成在先应用用户数据库查询操作的结果的行的子集。

11.根据权利要求9所述的系统，其中进一步配置所述处理器设备，以：执行抽样操作，从而识别所述块的行。

12.根据权利要求11所述的系统，其中所述抽样操作采用随机选择函数。

13.根据权利要求9所述的系统，其中为了获得统计结果值，进一步配置所述处理器设备，以执行如下中的一个或者多个：

采用处理收集统计结果，统计结果包括：所述块或者表的行计数值、所述块或者表的列计数值、所述块或者表的平均行长度值、统计结果的行计数和直方图、键行计数值、对所述块或者表的一个或者多个列产生的规定累计值、对所述块或者表的一个或者多个列产生的规定平均值、对所述块或者表的一个或者多个列产生的规定标准偏差值；以及

采用用户定义函数处理所述块中的所述数据，以计算统计结果值。

14.根据权利要求9所述的系统，其中所述目标数据库表包含由所述源数据库表复制的数据，进一步配置所述处理器设备，以：

在第一时间点，确定所述源数据库表中的所述识别块的行与所述目标数据库表中的所述相应块的所述行之间的第一组差异；

在所述第一时间点后至少复制等待时间间隔之后的第二时间点，确定所述源数据库表中的所述块的行与目标数据库表中的所述块的所述行之间的第二组差异，

基于不同时间点的所述第一组差异和第二组差异，将所述源数据库表的所述识别块与所述目标数据库表的相应块之间的差异确定为是持久性类型、非持久性类型、暂停类型或者未知类型中的一个。

15.根据权利要求14所述的系统，其中进一步配置所述处理器设备，以：

确定进入操作的未提交读出UR隔离级别模式，使得查询数据避免与一个或者多个用户应用争用锁，以及

如果进入所述UR模式，并且当前时间在复制等待时间间隔之后，则

再比较所述源数据库表和目标数据库表的相应块中的相应行的数据的所述累计数据值，并且如果确定所述行的所述再比较累计数据值匹配，则所述差异类型不是持久性的，否则，

如果确定所述再比较累计数据值差异为不匹配，则

比较当前确定差异结果和从在先等待时间间隔发生的所述行的差异确定结果，并且如果确定所述当前差异结果与在先差异结果之间的所述比较差异匹配，则

确定是否在使共享锁布置于提取行上并且使另一个处理将共享锁布置于同一个行上的操作的游标稳定性CS隔离级别模式下评估了所述行的所述差异结果，并且如果未在操作的所述CS级别模式下操作，则确定所述差异类型是未知的；

并且如果确定所述当前差异结果与在先差异结果之间的所述比较差异匹配，则确定所述差异类型是持久性的。

16.根据权利要求15所述的系统，其中如果确定所述行的所述比较的当前确定差异结果和所述在先差异确定结果不匹配，则进一步配置所述处理器设备，以：

在所述UR模式下，将对所述源数据库表和目标数据库表中的块的相应行重复所述再比较所述累计数据值规定的迭代次数，直到检测到匹配状况；并且如果在同一个迭代中未检测到匹配状况，则：

对所述行重复所述比较所述当前确定差异结果和所述在先差异确定结果；以及

如果在达到所述迭代次数之后，保持所述不匹配状况差异结果，则确定所述持久性类型是暂停的。

表数据比较的方法和系统

技术领域

[0001] 本发明涉及数据复制(replication),并且尤其涉及在数据库管理系统中采用的方法和系统,用于将源数据库结构中含有的数据与在目标数据库中复制的对应于源数据库表的数据进行比较,并且识别任何差异。

背景技术

[0002] 数据复制是企业通过数据冗余保证连续数据复制的惯常做法。从技术的角度出发,有基于磁盘复制方法和基于中间层软件复制方法。根据复制协议,有同步复制和异步复制。

[0003] 在异步复制中,做了导致数据变化的事务后,复制数据;因此,不影响源站点事务性能。为了证实数据是否100%准确复制,特别是当应用异步复制时,通常使用数据比较应用。对于数据库,进行比较以在每个关键值和每个记录列的记录数量方面保证源中的和(复制)目标的数据项一致(匹配)。

[0004] 当数据库表变得非常大,并且两个数据库在物理上相距远(例如,某个距离)时,这种比较非常昂贵,因为从表提取数据,并且将一个数据库中的行发送到另一个进行比较的成本。为了减少传递的数据量,可以传递一个或者多个记录的校验和,代替记录本身。仅当校验和比较不匹配时,才采用行比较。

[0005] 此外,为了改善性能,能够采用并行比较,并且为了确认有效而比较数据块的校验和。然而,并行比较仅改善比较的历时,而不减少工作量,并且因此,I/O和网络仍紧张并且占用CPU。

[0006] 对于许多客户,非常希望降低进行表差异比较的成本。这样既包含比较成本又包含比较花费的时间。此外,数据量可以极高。在这种情况下,比较必须使得当进行该比较时资源不崩溃。

发明内容

[0007] 本技术领域存在以某个准确性级别,而非基于逐行检查方式(例如,利用行数据或者校验和)提供轻量比较的需要。

[0008] 本技术领域还存在提供一种轻量比较,该轻量比较包含即使在正在比较数据时,当在源或者目标更新数据时,仍对活环境的跟踪机制。

[0009] 在该公开中,提供了一种能够以高准确性(高置信度)进行轻量表比较的系统和方法以及计算机程序产品。所公开的轻量表比较的特征从如下方面解决了该问题:1)基于统计结果的;2)基于抽样的;以及3)基于物化查询表(MQT)的,该MQT是其定义基于查询结果的表。

[0010] 根据一个方面,提供了一种表数据比较的方法。该方法包括:识别包括源数据库表的数据的行的子集的块和包括第二数据库表的数据的行的子集的相应块;获得与包含在源表的识别块中的数据关联的统计结果值,并且获得包含在目标表块的相应块中的数据的又

一个统计结果值;比较统计结果值,以确定匹配结果;以及基于比较的结果,确定每个源数据库表和目标数据库表的块是否一致,其中编程处理器设备执行识别、获得、比较以及确定操作。

[0011] 根据又一个方面,提供了一种用于表数据比较的系统。该系统包括:存储器,配置该存储器,以存储从源数据库表和目标数据库表收到的数据;处理器,该处理器与存储器通信,配置该处理器以执行方法,从而:识别包括源数据库表的数据的行的子集的块和包括第二数据库表的数据的行的子集的相应块;获得与包含在源表的识别块中的数据关联的统计结果值,并且获得包含在目标表块的相应块中的数据的又一个统计结果值;比较统计结果值,以确定匹配结果;以及基于比较的结果,确定每个源数据库表和目标数据库表的块是否一致。

[0012] 提供了一种执行操作的计算机程序产品。该计算机程序产品包含处理电路可读并且存储该处理电路运行的指令从而运行(各)方法的储存介质。(各)方法与上面所列的相同。

附图说明

[0013] 根据下面对其说明性实施例所做的详细描述,本公开的特征和优点显而易见,结合附图阅读该详细描述。附图中:

[0014] 图1是示出根据实施例的通用数据比较架构的原理框图;

[0015] 图2A是示出根据本公开提供的一个实施例的应用的数据流图,配置该应用,以确定数据库环境下的差异;

[0016] 图2B是示出根据本公开提供的一个实施例的表比较应用的应用架构的示意图;

[0017] 图2C是示出根据本公开提供的一个实施例的表比较应用的应用架构的示意图;

[0018] 图2D是示出根据本公开提供的一个实施例的飞行比较应用的应用架构的示意图;

[0019] 图3示出执行在此描述的方法的示例性硬件配置;

[0020] 图4A－4B示出一个实施例中的轻量表比较的通用处理;

[0021] 图5示出一个实施例中为了轻量表比较而从源表数据库和目标表数据库中提取相应块的统计数据的详图;

[0022] 图6示出在一个实施例中为了执行轻量表比较从MQT表提取数据的方法流程的一个实施例;

[0023] 图7示出在一个实施例中基于抽样的轻量表比较方法的一个方案;

[0024] 图8是示出在一个实施例中RETRY_COMPARE_WAIT时间间隔如何处理复制;

[0025] 图9示出在一个实施例中在活更新环境下确定特定差异的持久性类型的方法;

[0026] 图10示出如在此所述用于对持久性差异进行确定的系统;

[0027] 图11示出用于对采用使用仅专门用于对单独差异执行再检验的专用处理线程的持久性差异进行确定的系统;

[0028] 图12示出因为进度监视功能指出获得的完成统计结果的典型输出消息;

[0029] 图13是示出执行在此描述的详细过程的结果的典型差异检测结果表;以及

[0030] 图14示出典型报告,配置该典型报告,以在差异检测中提供全部统计结果。

具体实施方式

[0031] 现在将参考下面的讨论和本申请的附图更详细描述本公开。为了说明的目的提供本申请的图，本说明书的下面将更详细参考本申请的图。

[0032] 现在描述提供轻量表比较的系统、方法和计算机程序产品。在此描述的轻量表比较的系统和方法的特征从1)统计结果；2)抽样；和/或者3)物化查询表(MQT)方面确定源表和目标表中的差异，该MQT是其定义基于查询结果的表。

[0033] 根据进一步特征，现在描述的用于轻量表比较的系统、方法和计算机程序产品还运行，以在“活”更新环境下，检测对相应源和目标数据库表行项目中的差异，在“活”更新环境下，在执行比较操作的同时，更新初始源和/或者目标数据库表，以识别检测到差异的特定行的持久性类型。这可以包含在已经确定为不同的源和目标数据库表行中对基于数据行的并列数据块执行“再检验”的特征。

[0034] 因此，在第一方面中，为了比较数据，执行“小”块的比较。当比较“活”数据时，使用小数据块具有显著优势，因为仅当比较一对块时，既从源又从目标读出数据，包含在这些块中的行发生变化的概率小。实用程序捕获比较应用访问其之前对块的任何变化，而不考虑对该块完成比较之后发生的任何变化。

[0035] 因此，在第一方面中，提供了一种轻量表比较的系统和方法，以从1)统计结果；2)抽样；和/或者3)物化查询表(MQT)方面确定源(src)数据表和目标(trgt)数据表中的差异，该MQT是其定义基于查询结果的表。

[0036] 在一个方面中，利用“合并”过程，用于轻量表比较的系统和方法实现用于数据表比较的系统架构和操作环境，在该“合并”过程中，合并过程根据上述系统和共有的共同未决的标题为“Database Table Comparison”的美国专利公布No.2012/0317134和标题为“Difference Determination in a Database Environment”的美国专利公布No.US 2014/0372374描述的方法执行数据表比较，如同在此做全面陈述一样，通过引用合并这两个美国专利公布中每个的全部公开和内容。

[0037] 在又一个方面中，用于轻量表比较的系统和方法实现修改的系统架构和操作环境，以执行差异再检验技术，关于持久性，该差异再检验技术保证最新确定差异及其状况。

[0038] 图1是示出实现根据一个实施例的系统和方法的数据比较架构10的原理框图。该架构包含第一源数据库(DMBS)20A，该第一源数据库(DMBS)20A实现第一计算设备，例如，服务器21A并且存储包括源数据库表39的原始数据。配置源数据库系统20A，以提供数据，并且计算用于轻量数据库表比较用途的校验和等。同样，架构10包含第二目标数据库(DBMS)20B，该第二目标数据库(DBMS)20B实现第二计算设备，例如，服务器21B，并且存储由目标数据库表38中的源数据库复制的数据。可以配置目标数据库系统20A，以对比较应用作出响应，从而提供相应数据，并且计算目标数据库的相应校验和，以用于轻量数据库表比较用途。提供比较应用33，该比较应用33在远离头两个数据块38、39但是提供公共或者专用通信网络99通信的又一个计算设备25中实现为存储过程。在操作中，配置比较应用33，以起动在源数据库39中本地提取数据并且起动在目标数据库39中本地提取相应数据；接收源数据和相应目标数据；将包含在源数据库39中的块中的数据与包含在目标数据库39中的相应块中的数据进行比较；以及报告该结果。在一个方面中，由比较应用执行轻量数据库表比较。应当明白，在替换实施例中，比较应用不一定远离源数据库或者目标数据库，但是可以在本地

在源计算设备20A或者目标计算设备20B运行。此外，源数据库和目标数据库二者可以常驻并且共享单个设备。

[0039] 图2A是示出应用的数据流图70，根据轻量表比较技术配置该应用，以确定数据库环境下的差异。

[0040] 在图2A中，数据流图70示出根据本公开提供的轻量表比较的一个实施例的应用72，配置应用72，以起动数据库环境下的差异。如图所示，应用72在不同时点将源表74内的数据与目标表76内的数据进行比较，该不同时点分隔开诸如复制等待间隔的预定间隔。根据比较，应用72确定第一时点的差异78和第一时点之后的第二时点的差异80。根据差异78、80。应用72根据一组预定规则86确定持久性差异82和/或者瞬间差异84。然后，应用72输出至少一个持久性差异或者瞬间差异。在一些实施例中，通过基于包含差异的非关键值和关键值的校验和过滤第二时间点的差异80，应用72确定一组瞬间差异，并且接着，确定该组瞬间差异中的至少一个差异是持久性差异或者瞬间差异。在一些实施例中，利用源表或者块与目标表或者块之间的几个不同差异确定区别开源表和目标表的持久性差异和瞬间差异。例如，在一个实施例中，基于第二时点的差异80并且还基于第二时点之后的第三时点的差异，确定至少一个差异是持久性差异或者瞬间差异。

[0041] 在一个实施例中，在表比较应用的架构中可以实现在此公开的技术，该表比较应用允许在比较时对源表或者目标表进行更新，并且配置该表比较应用，以区分持久性差异和瞬时性差异，但是尽管如此还要配置该表比较应用，以有效比较WAN分离的表或者块并且与表或者块的大小无关。表或者块比较应用还称为比较应用是以包含预处理阶段、区分阶段和清洁阶段的三个顺序阶段比较表或者块的并行实用程序。实用程序在内部对这两个表分区，并且并行地比较这些分区。并行地对每个分区对进行行检索。然后，将在每个分区对中发现的差异组合，以获得总结果。

[0042] 图2B是示出根据一个实施例的比较应用的架构90的示意图。区分阶段的比较应用包含协同线程池，该协同线程池包含：主线程94、分区线程96、合并线程98、子输入和工作线程，如下所做的更详细讨论。在区分阶段，主线程94创建分区线程95，该分区线程95基于分块列(block-by column)划分表比较操作，并且将表比较操作划分为较小的但大小相同的子操作或者任务，该子操作或者任务中的每个对应于源表和目标表的子集或者分区。分块列是索引列，该索引列能够是分区键、主键或者任何列集。在这方面，分区线程95从源表74中选择每个分区的边界分块值，包含最小边界分块值和最大边界分块值，其中源表74存储于源数据库79中。在一个实施例中，将分区边界确定为源表74的块数和源表74的总行数的函数。在另一个实施例中，将边界确定为特定块中的预期行数和源表的行大小的函数。分区线程95利用边界分块值产生查询语句，以从源表和目标表中提取特定分区或者块。然后，分区线程95通过任务队列97将查询语句传递到合并线程98。根据一个实施例，块表记录任务队列97处理的特定块语句。

[0043] 图2C是示出根据本公开提供的一个实施例的比较应用的架构100的示意图。如上所述，在运行时并且基于预定工作负荷平衡策略或者拉出模型，分区线程95通过任务查询将查询语句和关联分区比较任务分配到合并线程98。合并线程98识别与相应分区比较任务的区别。为此，每个合并线程98创建两种工作线程102_1，……，102_n，这两种工作线程102_1，……，102_n包含：仅与源数据库交互的工作线程和仅与目标数据库交互的工作线程。对

于每个分区，合并线程98通过任务容器对每个工作线程102分配相应描述相应分区的查询语句。然后，工作线程102调用在源数据库和目标数据库中的每个上的存储过程，或者直接从源数据库和目标数据库提取数据。

[0044] 在一个实施例中，预定并且配置存储过程，以通过查询语句识别的分区返回聚合校验和。在此还将聚合校验和称为合成校验和。存储过程接收查询语句作为输入参数，并且对该数据块执行多行或者块提取，以抽取识别分区中的所有行或者块。存储过程计算每个行或者块的相应行校验和。将键值和校验和插入与调用工作线程102关联的不记入日志的全局临时表(GTT)实例。此外，存储过程调用分区校验和功能是将识别的分区中的所有基于行的校验和聚合为单个校验和值。在一些实施例中，分区校验和比行校验和长。例如，在特定实施例中，每个行校验和的长度是4位，而每个分区校验和的长度是8位。

[0045] 在一个实施例中，如果两个分区校验和匹配，则合并线程98将当前分区看作在源表和目标表中一致，并且从任务队列中请求下一个分区或者块。相反，对于允许从全局临时表提取，合并线程98与其他合并线程竞争。获得允许后，合并线程98将和并请求发送到工作线程102，以起动合并比较子阶段。在该合并比较子阶段，运行于分区上的两个工作线程102从由键顺序分类的全局临时表提取键和相应基于行的校验和，并且通过校验和项目队列，将该键和相应基于行的校验和送到合并线程98。然后，合并线程98对键值执行合并连接，以逐行地或者逐块地发现差异、通过差异队列106对差异报告线程104报告识别到的差异。

[0046] 在另一个实施例中，运行于分区上的两个工作线程102提取由键顺序分类的源表和目标表，并且计算相应基于行的校验和，并且通过校验和项目队列，将它们送到合并线程98，而不必调用存储过程。然后，合并线程98对键值执行合并连接，以逐行地或者逐块地发现差异，通过差异队列106对差异报告线程104报告识别的差异。

[0047] 在一个实施例中，配置在此还称为报告线程的差异报告线程104，以从差异队列106读出发现的差异，并且将差异项插入差异表中，该差异项含有识别差异和诸如插入、更新或者删除的相应动作的键值，以校正差异。在清除阶段，如果操作，则比较应用从数据块除去全局临时表的定义，并且输出识别的差异和每个线程的完成统计结果。

[0048] 为了示例性和非限制性目的，通过扩展表比较应用的架构，可以实现在此公开的用于轻量表比较的技术。扩展架构可以用作用于的基础。在此，这种应用还称为飞行比较应用程序。

[0049] 在一个实施例中，对于每对源表和目标表，应用产生或者生成差异表。差异表中的每个记录是差异项并且表示源表与目标表之间的特定行差异，其中利用其键值可识别每行。每个差异项包含一组键值、差异标志以及持久标志。该组键值指在源表和目标表中都作为键值的一组属性值。利用该键值，能够提取源表和目标表中的相应行。差异标志指明在源表与目标表之间特定键行如何差异。从包含更新、插入和删除的至少三种差异标志中选择差异标志－例如是否为了校正差异而要求执行更新、插入或者删除操作。持久标志指明差异项是代表持久项还是瞬时项，并且从包含持久、非持久(瞬时)、未知和怀疑的至少四种持久标志中选择持久标志。对于用户，头3种标志是外部标志。对于差异初始化，怀疑是内部类型的。

[0050] 在一个实施例中，两行之间的每个比较使用两行之间的最后比较的结果，以确定两行之间的任何差异的持久性。如果对于复制保持的表发生更新可疑，则应当在从该最后

比较操作开始经历了复制等待窗口之后执行当前比较操作。可以将比较结果划分为如下三种。第一，如果在两组结果中都存在差异并且相应行仍相同，则认为该差异是持久性的。第二，如果在当前比较结果中不存在差异，则认为该差异是瞬时的。根据该实施例，差异已经飞行、已经回滚或者已经修复。为了释放存储空间，可以在第二次比较之后去除这些差异。第三，如果在当前比较结果或者在两个结果中存在差异，而且其相应行不同，则认为差异是未知类型的，并且可以在后续比较操作中确定其。

[0051] 在一个实施例中，当根据这里的技术配置时，应用呈现一组性质，包含：持久识别、较高并行性、较低开销，并且改善可用性。如上所述，利用差异表中反映的先前识别的差异，飞行比较应用确定差异的持久性。在一些实施例中，为了更有效捕获关于键值和非键值的差异变化，应用保持基于比较的校验和(CCRC)。在一个实施例中，每个CCRC值分别聚合来自源表和目标表的两个基于行的校验和。因此，还可以将基于列比较的校验和称为聚合校验和或者合成校验和。关于CCRC在这些行校验和之外呈现的性质，当将其CCRC值进行比较时，可以简化比较两个连续不同结果的过程。在一些实施例中，对于CCRC值的校验和冲突，应用还可以配置有预定处理程序。在一些实施例中，在差异表中，还将CCRC记录为新列。

[0052] 图2D是示出根据本公开提供的一个实施例的飞行比较应用的架构120的示意图。在一些实施例中，可以将合并线程看作代理122_1，……，122_n，配置该代理122_1，……，122_n，以确定源表与目标表的相应块之间的差异。类似地，可以将差异报告线程看作差异报告代理，并且与在此公开的其他线程类型也如此。在差异报告线程处理了差异项并且因此将结果记录在差异表中之后，将确定的差异作为差异项插入差异队列中。在一些实施例中，在假定源表和目标表具有小于阈值量的差异，例如，小于0.01％的差异的情况下，比较应用运行。然而，这种假定在活动时/活动环境下可能不正确，其中瞬时性差异的数量可能大。因此，架构以如下方式中的一个或者多个扩展。例如，不使差异队列仅占用单个存储块，而提供具有至少两个旋转存储块124的差异队列，并且还将其称为队列差。当差异报告线程104正在读出并且处理块的内容时，合并线程不应当对块进行内容变更，直到差异报告线程完成处理整个块。

[0053] 作为另一个例子，不立即对差异报告线程报告差异并且然后分别插入差异，而可以基于线程协调和差异插入的预定编组策略，批处理差异。在一个实施例中，在确定满足如下条件中的一个之后，合并线程将阵列插入的给定存储块准备好处理通知差异报告线程：(i)差异队列中的存储块124中的一个是满的；(ii)在特定时间间隔内，例如在对应于识别到差异时的时间间隔内，存储块124中的一个含有特定数量的差异；以及(iii)完成全部比较。在一个实施例中，差异报告线程能够在多行插入能够使用的相应描述符区域内直接使用每个块。描述符区域指动态执行插入语句要求的变量的集合。描述符区域的一个例子是SQL描述符区域(SQLDA)，该SQL描述符区域(SQLDA)是执行SQL INSERT语句要求的变量的集合。每个变量描述代表含有目标表的列的一个或者多个值的缓冲器的主变量阵列。

[0054] 作为又一个例子，不仅仅将键值和关联动作保持为差异表的每个差异项的一部分，扩增差异项，以对每个包含一个或者多个如下属性：持久性类型、基于比较的校验和以及创建时间戳。在一个实施例中，差异报告线程确定持久性类型，而由合并线程确定基于行的校验和。

[0055] 在一个实施例中，为了改善比较性能并且为了确定报告差异的持久性类型，CCRC

表示特定行的比较结果。假定其键值能够识别源表或者目标表的块中的每行，则可以合成键值，以产生特定行校验和值。每个CCRC由两个相应特定行校验和值聚合。至少在某些情况下，CCRC可以简化比较，例如，仅比较两次不同调用飞行比较应用产生的相同键值的CCRC。如果校验和匹配，则认为差异是持久的，否则认为差异是瞬时的。

[0056] 在一个实施例中，利用各种预定技术的任何一种，应用将两个基于行的校验和聚合为CCRC。例如，按位“异或”(XOR)可以用作聚合函数。为了计算CCRC，源特定行校验和和目标特定行校验和不应当匹配。如果该校验和相同，则意味着各行互相不同。此外，为了达到均匀分布特定行校验和值，应当类似地分布XOR产生的校验和值。另外，尽管两个不同差异可能具有CCRC冲突，但是配置应用，以支持处理该冲突。

[0057] 本说明书的下面将更详细讨论在数据库表比较操作时发生的活更新条件下用于识别差异持久性类型的持久性差异识别。

[0058] 图4A－4B示出一个实施例中的轻量表比较的通用处理。

[0059] 特别是，图4A－4B示出在一个实施例中在图1的比较应用33中的区分阶段运行的合并线程为了执行“轻量”表比较而执行的方法200。如图4A所示，在此所示的用于轻量表比较的方法200从使用统计结果、表抽样、MQT方面或者这些方面的组合出发进行比较。

[0060] 在合并线程执行方法200时，假定例如用户已经通过界面(例如，图形的、命令行参数或者其他方式)设定了用于轻量数据库表比较模式的“COMPARE_MODE”切换变量，指明了轻量表比较模式。例如，在一个实施例中，用户可以从COMPARE_MODE变量已经被设定为指出“METRICS”情况下的统计结果方面出发选择表比较模式。另外，用户可以从COMPARE_MODE变量已经设定为指出“SAMPLE”的情况下的抽样方面出发交替地选择表比较模式。最后，用户可以从COMPARE_MODE变量已经设定为指出MQT轻量数据库表处理的情况下的MQT方面出发交替地选择表比较模式。在替换实施例中，默认设定可以指出特定轻量数据库表比较模式。

[0061] 因此，例如，在图4A中的201，合并线程获得源数据库的尚未比较块TBi，并且在203，关于用户例如利用统计结果或者抽样是将切换变量“COMPARE_MODE”设定为指出执行全量表比较的FULL MODE还是指出轻量表比较模式，进行第一确定。如果COMPARE_MODE变量指出FULL MODE表比较，则该处理进入步骤225，在步骤225，在本技术领域内众所周知，执行全模式数据块表逐行比较。另外，在步骤203，如果COMPARE_MODE变量未指出FULL编辑模式，则该处理进入步骤206，以对是否已经选择利用统计结果的轻量比较模式进行确定。因此，在步骤206，对于用户是否将切换变量“COMPARE_MODE”预设为指出利用计算的数据库统计结果值或者比较块的其他度量值进行轻量表比较的METRICS模式进行判定。如果COMPARE_MODE变量确实指出METRICS，则该处理进入步骤210，在步骤210，调用相应工作线程，以提取从源表和目标表选择的块的统计结果，以根据统计结果比较模式进行轻量表比较。将参考图5更详细描述关于分区线程提取统计结果的进一步详情。

[0062] 在210从源数据库表和目标数据库表的相应块提取了相应统计结果后，该处理进入步骤215，在步骤215，合并进程执行从源数据库表和目标数据库表提取的相应块的统计结果值的实际比较。另外，返回206，如果确定切换变量“COMPARE_MODE”未指出METRICS，则该处理在213继续，以获得从源数据库表和目标数据库表选择的相应MQT块，从而根据MQT数据库表比较模式继续轻量表比较，此后，该处理进入步骤215，在步骤215，合并进程执行源MQT表和目标MQT表的实际比较。应当明白，在一个实施例中，用户预设的“COMPARE_MODE”切

换指出“MQT”轻量表比较模式，在这种情况下，执行MQT轻量数据库表比较。将参考图6更详细描述关于工作线程提取MQT数据进行比较的进一步详情。

[0063] 尽管图4A未示出，但是在其他实施例中，提供能够与基于统计结果的轻量表比较解决方案或者MQT集成的基于SAMPLING的解决方案。在此描述的基于SAMPLING轻量表比较解决方案还能够与在此合并的共有的共同未决美国专利公布No.2012/0317134和No.US 2014/0372374描述的两种方法中的任何一种集成。将参考图7更详细描述关于工作线程从表抽样行从而形成块来进行比较，或者形成块和相应统计结果来进行统计结果比较的进一步详情。

[0064] 在图4A中的215，无论是否正在将相应提取的源MQT表和目标MQT表进行比较，也无论是否正在将对应于源块和目标块的统计结果进行比较，也无论是否正在将相应源表和目标表的抽样行的块进行比较，该处理都进入步骤218，以确定从相应源数据库和目标数据库提取的统计结果、提取MQT表或者提取的抽样数据是否匹配。

[0065] 作为例子，在实施例中，对相应源数据库表和目标数据库表中的一个或者多个相应块执行基于统计结果的轻量表比较。即，正在比较与每个块关联的统计结果。例如，数据库提供统计结果值，或者作为一种选择，利用与每个源表和目标表中的相应块关联的标准函数或者用户定义函数，计算统计结果值(不是表中的实际数据)，并且使每个统计结果值返回比较应用，以在218进行比较。

[0066] 如果在218确定相应比较值匹配，则该处理继续，并且不需要从块提取数据，因为基于比较数据(例如，统计结果)认为源块和目标块中的所有行匹配。因为为了进行轻量表比较可以比较源表和目标表中的另外块，所以该处理进入图4B中的步骤248，在步骤248，对于合并线程是否将利用轻量表比较处理更多块进行确定。如果表中还存在块，则该处理返回图4A中的201，以重复处理下一个块。另外，在图4B中的248，如果不再有块要处理，则比较应用结束或者执行其他处理。

[0067] 另外，如果218确定从源数据库表和目标数据库表获得的提取值(统计结果、MQT或者抽样)不匹配，则实用程序可以通过显示器对用户报告源表和目标表中不匹配的特定块或者块的边界或者形成块的行的范围指出表差异，而不报告任何特定行的知识。在又一个实施例中，在图4A中的218确定不匹配后，该处理进入步骤221，以确定是否调用存储过程(USE_SP)，在该存储过程(USE_SP)中，为了特定块的比较，计算基于行的校验和。

[0068] 即，在221，如果确定未调用存储过程，则该处理进入226，以进一步执行基于行的处理，从而确定并且报告源表与目标表之间识别到的差异。，这需要在226使工作线程提取源表和目标表，并且计算基于行的CRC，对于基于行的每个CRC，合并线程可以比较并且报告任何识别差异。此后，该处理进入图4B中的248，以确定是否还有块要处理，在这种情况下，如果确定表中没有要处理的另外块，则该处理将返回图4A中的步骤201。然后，比较应用结束或者执行前天处理。另外，在图4B的248，如果没有要处理的块，则比较应用结束，或者执行前提处理。

[0069] 另外，返回221，如果要调用处理过程，则该处理进入230，在230，关于源表/目标表的当前块的资源使用(例如，调用存储过程消耗的全局临时表空间大小)是否可以超过先前确定的最大容量阈值“GTT_MAXSZ”，进行确定。正如基于抽样的实施例中为了产生块使用的资源消耗可以大也可以小。然而，如果太大，则限制并且避免对非常大的块应用存储过程。

因此，在230，如果确定块大小大于“GTT_MAXSZ”阈值，则该处理进行，以执行根据226的处理的处理。另外，如果块大小不超过GTT_MAXSZ最大阈值，则在235，调用存储过程，在235，工作线程利用Row_CRC_Only模式调用存储过程，并且在240，在存储的不记入日志的全局临时表(GTT)中提取基于行的校验和结果集，对于该基于行的校验和结果集，合并线程可以比较键值和基于行的CRC并且报告任何识别的差异。此后，该处理进入图4B中的248，以确定是否还存在要处理的块，在这种情况下，如果确定表中还存在要处理的块，则该处理将返回图4A的步骤201。然后，比较应用结束或者执行其他处理。另外，在图4B中的248，如果不存在要处理的块，则比较应用结束或者执行其他处理。

[0070] 在一个实施例中，在步骤226和240报告识别的差异包含确定与确定的差异关联的持久状况。在另一个实施例中，在步骤226和240第一时间报告识别的差异触发确定与确定的差异关联的持久状况。为了确定差异的状况，可以要求多次执行方法200。在一个实施例中，在差异再检验阶段，将每个差异看作单独块。正如在本说明书的下面所做的进一步详细描述，在确定检测差异的持久状况时，提供差异再检验机制。

[0071] 现在参考图5，图5示出为了轻量表比较而从源表数据库和目标表数据库提取相应块的统计结果数据的图4A的方法步骤210的详情。

[0072] 图5示出一个实施例中通过使对应于为了进行“轻”量表比较从相应源数据库表和目标数据库表中提取相应块的统计结果的合并线程的工作线程运行执行的方法250。每个合并线程具有两个工作线程，以并行提取统计结果。一个工作线程用于源表，另一个用于目标表。

[0073] 如图5所示，方法250从统计结果的方面出发完成轻量表比较。在该实施例中，逐块地执行基于统计结果的表比较。在第一步骤255，工作线程获得块的说明(specification)，以利用统计结果执行表比较。在一个实施例中，可以将该块规定为表的子集或者整个表本身。根据本公开的实施例，提取的块可以包括抽样行和通过执行在本说明书的下面更详细讨论的抽样技术获得的一个或者多个列的块。

[0074] 在一个实施例中，用户可以通过界面规定块大小。可以在表分区时产生块。例如，根据下面描述的抽样技术，用户可以选择特定块大小，并且然后实用程序能够基于每行的储存字节的数量确定每块的表行的平均数量。基于选择的行数量确定该块的行的范围的分块边界值，并且在产生的查询语句中提供该分块边界值，以获得该块的统计结果值。即，可以对从每个表中选择行抽样，从每个表中选择行也可以基于范围。使选择行(列)聚合，以由数据库表形成一个“块”。该“块”可以包含整个表，但是通常是整个表的选择行。

[0075] 然后，获得块后，在图5的258获得要比较的表的块的统计结果类型的说明。在一个实施例中，基于规定的统计结果类型，确定是否能够利用内置函数获得规定的统计结果。因此，在260，关于是否为了计算块的统计结果值而调用预定的或者内置的数据库例程进行判定。例如，在263，利用数据库中内置的标准函数，包含但并不局限于：表行计数(基数(cardinality))，列计数、平均行长度以及基于行的或者基于列的校验和函数，可以计算块的统计结果值。作为一种选择，或者此外，如265所指出的，可以利用用户定义函数计算统计结果值。无论是在263通过内置数据库例程获得还是在265从用户定义函数获得，合并线程产生了源块和目标块的统计结果值后，该过程都返回图4A中的步骤215，在步骤215，比较应用中的合并线程最后将获得的每个源表和目标表中的相应块的统计结果值(例如，表行计

数(基数)、列计数、平均行长度)进行比较。

[0076] 在一个实施例中,作为基于统计结果的表比较方法的一部分,在263,工作进程收集某些数据库程序已经收集的统计结果,例如,表行计数(基数)、列计数、平均行长度等等。在该实施例中,代替从源表和目标表提取行,比较应用的工作线程对数据库发出SQL语句。该值在两个数据库上的数据库的目录表中直接可用,或者该值能够基于实际表数据或者索引数据产生/计算。然后,合并线程比较两个表的统计结果。在一个实施例中,面向SQL语句获得一个或者多个上述统计结果(表行计数(基数)、列计数、平均行长度等等)。

[0077] 此外,由于在263,一些数据库程序已经收集了特定记录统计结果,诸如基数和直方图统计结果列、键基数等等,所以比较应用工作线程对两个数据库上的数据库目录表发出SQL语句,并且合并线程比较源表和目标表二者中的相应块的这些基数和直方图统计结果列、键基数统计结果。在一个实施例中,可以面向SQL语句获得一个或者多个上述统计结果。

[0078] 在又一个实施例中,在图5中的263或者265,比较应用工作线程可以调用内置统计结果或者(各)度量函数(例如,MAX、MIN、AVG、SUM、VARIANCE和STDDEV),可以预定义内置函数,内置函数也可以是用户定义函数,例如,用户能够规定对一个或者多个列产生的累计值。例如,这些另外统计结果可以包含但并不局限于:对所述块到表的一个或者多个列产生的规定平均值、对所述块到表的一个或者多个列产生的规定标准偏差、等等。能够单独地或者组合地使用这些内置统计结果或者用户定义函数。然后,由数据库对象减去这些统计结果或者累计值。例如,可以对特定数据库、“块”的特定集,例如,表、或者列的特定子集、或者行的特定范围,获得统计结果值或者累计值。在一个实施例中,基于自动确定列值范围,将每个表分割为多个块。每个块具有一个或者多个累计值/统计结果值。在另一个实施例中,利用抽样方法可以产生统计结果值/累计值,正如在本说明书的下面所做的更详细描述。

[0079] 此外,在又一个实施例中,在图5中的263,比较应用工作线程可以调用(各)内置函数,以获得并且计算运行时间统计结果或者实时统计结果。即,可以编制比较应用,以记录并且接收更新次数、在每个数据库对象的之前间隔中插入和删除。从源表和目标表二者之间的一致最初状态开始,比较实时统计结果有助于检测例如10秒的预定时间间隔内的数据变化导致的不一致性。由于异步数据复制中存在延迟,所以当为了比较识别时间窗口时,需要外加该延迟。工作线程实现差异再检验机制(例如,诸如专用表中的时间令牌),以解决该问题,正如本说明书的下面所做的更详细讨论。

[0080] 比较与该表的每个块关联的差异统计结果或者累计值提供不同认识层面或者不同置信度的数据一致性。

[0081] 在又一个实施例中,提供了用于轻量表比较的基于MQT(物化查询表)的方法。

[0082] 由于在计算时知道物化图(MV或者物化查询表MQT),所以形成含有例如用户查询的查询的中间结果或者最终结果的数据库对象。例如,其可以是位于远程的数据的本地拷贝,其也可以是表或者连结结果的行和/或者列的子集,还可以是基于表数据的聚合的累计。可以将MV看作映射选择表的结果的函数。因此,在一个实施例中,根据如下考虑MV图(例如,表1):

[0083] MV(Table1)=func(Table_1);

[0084] 或者作为一种选择,根据如下考虑:

[0085] MV(Table1,Table2..Table_n)＝func(Table_1,Table_2..Table_n),

[0086] 其中函数func能够是诸如聚合函数,诸如sum()、average()、max()、min()、count()、join(例如,多个表Table_1、Table_2等等)。例如,MV或者MQT可以是两个或者多个表的合并或者连结。MV或者MQT是分别存储的表。

[0087] 一些数据库保持物化查询表,以将预计算的查询结果或者子查询结果存储于独立表中,以加快查询处理的速度。非常常见的是,MQT表存储基表中的数据聚合的结果。通常,数据库中存在当更新数据时,或者当周期性地刷新MQT数据时递增更新MQT的机制。

[0088] 当更新其初始源表时,更新/刷新MV。在大多数DBMS(数据库管理系统)中,这由数据库自动触发和管理。例如,作为对初始表中执行的任何DELETE、INSERT或者UPDATE的响应,数据库自动刷新和更新MQT。因此,在一个实施例中,对于每个要比较的表或者每个表块,基于MQT的轻量表比较要求MQT。

[0089] 在轻量表比较中,存在采用MQT的两个独立方面。

[0090] 1)第一方面是仅将MQT定义为初始表的子集。“子集”可以是原始表仅在表列/表属性的子集上的投影。该子集还能够是行或者分区的子集。与全表比较相比,这种MQT具有比初始表小的大小,并且在诸如I/O、存储器脚印和计算的计算成本方面,源MQT和目标MQT的比较不昂贵。

[0091] 下面是为了对整个源表和目标表执行全面比较而创建的物化图查询的这种定义的一个非限制性例子。在这种典型用户查询中,假定源数据库提供基于行的校验和函数CHECKSUM(*)。对于典型的源数据库表MQT_1_s创建:

[0092] Create MQT_1_s as(SELECT KEY_COL1,KEY_COL2,CHECKSUM(*)as ROW_CRC FROM SCH1.TAB1_SOURCE);

[0093] 其中KEY_COL1、KEY_COL2是第一表(TAB1_SOURCE)中的列选择。对于典型目标数据库表MQT_1_t创建:

[0094] Create MQT_1_t as SELECT KEY_COL1,KEY_COL2,CHECKSUM(*)as ROW_CRC FROM SCH2.TAB1_TARGET)。

[0095] 其中KEY_COL1、KEY_COL2是第一表(TAB1_TARGET)中的列选择。通过退出RDBMS机制,实时处理MQT刷新。

[0096] 2)第二方面是例如仅为了比较的目的,通过提取预定义MQT的结果,修改基于统计结果的比较。这样的优点是利用MQT,能够定义适合特定统计结果的、甚或比RDBMS统计结果目录提供的更负责/更综合的统计结果函数。

[0097] 用户定义MQT表的定义的一个非限制性例子,以在对特定块或者整个表进行基于统计结果的比较时使用,对于源数据库表,物化查询如下:

[0098] Create MQT_2_s as(SELECT COUNT(*)as ROW_COUNT,SUM(COL1)as SUM_COL1 FROM SCH1.TAB1_SOURCE);

[0099] 其中实现决定行(COUNT(*))的数量的行累计统计结果函数SELECT COUNT(*),并且对于目标数据库表:

[0100] Create MQT_2_t as SELECT COUNT(*)as ROW_COUNT,SUM(COL1)as SUM_COL1 FROM SCH2.TAB1_TARGET。

[0101] 引入了MQT后,表比较应用能够在两个相应MQT表之间进行逐行比较,代替提取表

和计算基于行的校验和。如果MQT的内容不匹配，则比较应用能够对特定行或者特定块或者整个表采用正规比较方案，而无需MQT。

[0102] 通过退出RDBMS机制，能够实时刷新MQT和RDBMS统计结果。因此，基于MQT的解决方案的不同之处在于，MQT表能够存储查询的结果集，并且当发生任何变化时，能够立即刷新该查询的结果集。对于物化查询表，可以使对初始表所做的改变，例如，作为DELETE、INSERT或者UPDATE表操作的一部分级联。对于表比较，要求每个要比较的表或者每个表块的MQT。

[0103] 图6示出在一个实施例中为了利用MQT(物化查询表)执行表比较图4A的数据提取步骤213的方法流程270的一个实施例。在一个实施例中，在基于抽样的解决方案或者基于统计结果的解决方案中或者基于原始全模式比较解决方案中，能够使用基于MQT的解决方案。该方法利用数据比较工具定义并且创建MQT。由于用户可能已经具有已经对应用/查询目的定义的类似MQT，所以这里的方法可应用于并且不排除使用用户定义的MQT代替定义MQT的比较工具。

[0104] 在一个实施例中，逐表执行基于MQT的表比较。因此，在一个实施例中，在图6的275确定是否存在与源MQT关联的特定统计结果，并且对于轻量表比较，可以比较与目标MQT关联的统计结果。如果在275确定没有基于统计结果的比较要执行，则该处理进入278，在278，创建或者检索定义为初始对应于全源数据库和目标DB的子集的MQT表。例如，该步骤可以规定或者获得上面描述的MQT_1_s表和MQT_1_t表。然后，该处理返回图4A的步骤215，合并线程实际执行源MQT表和目标MQT表的基于轻量表的比较。

[0105] 返回图6的步骤275，如果有基于特定统计结果的比较的说明，则该处理进入步骤280，以确定是否已经对基于统计结果的表比较定义了现有的或者用户定义的MQT表。如果没有定义在前用户定义的MQT表，则该处理进入步骤282，在步骤282，创建或者获得定义的MQT表，该定义的MQT表具有适合对应于源表和目标表中的特定块的或者整个源数据库和目标数据库的统计结果函数。然后，可以获得源MQT表/目标MQT表中的统计结果用于比较。另外，在图6的284，根据具有适合相应源数据库/目标数据库的统计结果函数的用户定义MQT源表/目标表的选择行/列，检索统计结果。例如，对整个表的基于统计结果的比较包含上述物化查询MQT_2_s和MQT_2_t。在另一个例子中，MQT表存储源数据库和目标数据库上的每个键值的行的列的总和和计数。比较应用的合并线程比较MQT的内容。

[0106] 在两种情况下，对源数据库MQT表和目标数据库MQT表获得统计结果后，该处理返回图4A的步骤215，以执行基于统计结果的表比较。这可以调用基于统计结果的表比较处理，诸如在此参考图4A所述。

[0107] 结合抽样，MQT甚至能够位于表中的行的SAMPLING的顶上，因此，其能够用作原始表的预计算的累计表。在一个实施例中，每行能够对应于键列值的特定范围。

[0108] 在又一个实施例中，提供了一种用于轻量表比较的基于抽样的方法。在一个实施例中，提供了一种基于抽样的表的分区解决方案。对于轻量表比较，执行抽样，以确定哪个表和表的哪个子集需要比较。基于抽样的方法哪个是基于行的、基于列的、基于块的、基于页的、基于分区的、基于表的、基于表空间的或者基于数据库中的一个。

[0109] 在在此合并的US 20140372374和US20120317134的公知框架中，并且如上所述，分区器是扫描表并且产生合并器的块的线程。将产生的块插入称为taskQueue队列。合并线程仅比较taskQueue中可用的块。在一个实施例中，当采用抽样时，分区器仅随机拾取少量块，

并且将这些块插入taskQueue,代替产生/插入所有块。在另一个实施例中,当采用抽样时,分区器将产生由随机选择的行构成的块。

[0110] 因此,当产生特定块SELECT语句时,该方法还可以随机选择比较哪列。因此,不需要比较所有块的所有列。

[0111] 当每块含有一行并且仅含有一行时,其是基于行的抽样。

[0112] 当每块对应于整个表分区时,其是基于分区的抽样。或者当选择分区时,能够对该选择分区产生多个较小的块。

[0113] 在一个实施例中,选择规则能够是:1)完全随机,并且在一个实施例中,基于条件公式:

[0114] rand()>sample_rate

[0115] 其中sample_rate是用户规定的或者机器预定的预定义值;2)对先前未比较的块完全随机,即,在该实施例中,需要将已经比较了哪个块记录在先前比较历史中;3)通过预计算行数并且然后每隔第n行选择等同子集,也可以实现抽样;4)利用分布(直方图)、优选序列或者键值、基于时间的(更近更新的表段)、基于循环的等等,抽样选择判据能够是随机的;或者5)在一个实施例中,一些DBMS引擎包含在查询树的底部仅规定可能行的抽样的能力。例如,DB2 LUW包含TABLESAMPLE函数,这样允许人或者利用行级Bernoulli抽样或者利用系统页级抽样规定抽样概率。

[0116] 在基于抽样的轻量表比较方法中,代替比较表中的或者表的所有列中的所有记录,使用表数据的抽样。抽样在如下两个方面显著降低成本:1)其仅存取表记录的子集,并且减少后续数据处理量;2)特定抽样选项发出SQL语句,该SQL语句的执行比全比较SQL语句的执行廉价。例如,在CPU时间方面,“Select count(*)”语句(例如,仅用于计数的提取记录)能够比“select*”(选择全部)语句廉价,因为在运行时栈(接口)的较高级别,较少的数据需要通过,并且因此具有较短的指令通路长度。

[0117] 因此,在一个方面,对于相应列,表分区不执行分类和从表提取所有行,相反,在本说明书的实施例中,该方法执行分类并提取抽样行,并且利用抽样行确定每个块的边界。

[0118] 图7示出在一个实施例中基于抽样的轻量表比较方法285的一个方案。如图7所示,使用抽样获得块需要获得抽样集,以得到边界值、抽样率和随机函数,从而获得要求大小的块。获得边界值后,产生SQL语句,以从源数据库和目标数据库提取块。

[0119] 如图7所示,为了轻量表比较形成块的基于抽样的方法285包含第一步骤288,该第一步骤288示为确定表分区线程是否已经收到表示用户对每个块规定的抽样行数量的SAMPLE_PER_BLOCK参数值。如果收到的SAMPLE_PER_BLOCK参数被用户定义,则该处理进入步骤290。另外,如果收到的SAMPLE_PER_BLOCK参数值未被用户定义,则在289,系统获得默然SAMPLE_PER_BLOCK参数值,并且然后,进入步骤290。在一个实施例中,默然SAMPLE_PER_BLOCK块值是50。

[0120] 在图7的步骤290,根据如下,由SAMPLE_PER_BLOCK参数值求得抽样阈值SAMPLE_THRESHOLD:

[0121] SAMPLE_THRESHOLD=max(min(max_sample_threshold,SAMPLE_PER_BLOCK/num_rows_per_block),min_sample_threshold)

[0122] 其中num_rows_per_block,max_sample_threshold和min_sample_threshold是由

程序确定/调节并且收到的或者由用户规定的值。

[0123] 然后，在图7的步骤295，产生为了形成块从表提取抽样使用的SQL查询。在一个典型实施例中，为分类和提取行产生的SQL查询可以是：

[0124] SELECT BLOCK_BY_COL1,BLOCK_BY_COL2 FROM SOURCE_TAB WHERE RAND()< SAMPLE_THRESHOLD ORDER BY 1,2

[0125] 其中RAND()函数返回介于0与1之间的随机浮点值。假定非零值的SAMPLE_PER_BLOCK，则SAMPLE_THRESHOLD是介于0.00001与1之间的浮动值。例如，DBMS引擎提供内置随机值，生成函数(例如，DB2 z/OS)提供RAND()，这样返回介于0与1之间的随机浮点值。例如，利用随机选择，下面的查询将返回整个表的约1％行：

[0126] SELECT BLOCK_BY_COL1,BLOCK_BY_COL2 FROM TABLE1 WHERE RAND()<0.01 ORDER BY 1

[0127] 然后，在图7的398，根据抽样阈值，确定哪些行将用于块的边界行。第(n-1)边界切割和第n边界切割能够用于形成第n块。整个查询结果集中的第n边界切割的行数等于：

[0128] n*SAMPLE_THRESHOLD*num_rows_per_block

[0129] 因此，对于块生成，特别是当行数巨大并且分块列的总长度不小时，提取所有分块值非常昂贵。为了缩短CPU时间和历时，在键值检索中采用抽样。代替提取所有分块列值，表分区线程仅提取称为抽样的行的子集。利用隐藏参数SAMPLE_PER_BLOCK，用户能够控制每块的预期抽样数。基于抽样获得块。

[0130] 在又一个实施例中，利用分布(直方图)、优选序列或者键值、基于时间的(更近更新的表段)、基于循环的等等，抽样选择判据能够是随机的。在一个实施例中，一些DBMS引擎包含在查询树的底部仅规定可能行的抽样的能力。例如，DB2 LUW包含TABLESAMPLE函数，这样允许用户或者利用行级Bernoulli抽样或者利用系统页级抽样规定抽样概率。

[0131] 在另一个实施例中，通过预计算行数并且每隔n行选择等同子集，通过线上分析处理(OLAP)函数也可以实现抽样。

[0132] 应当明白，在一个方面，所描述的三种不同轻量表比较技术能够单独地使用也可以组合，例如，将基于统计结果的轻量表比较和为了大表分区而抽样的数据组合。

[0133] 当轻量比较报告不一致性时，能够将全模糊比较(full-bloom comparison)用于不规则检测。在一个实施例中，当抽样的数据库对象的累计值/统计结果值不匹配时，开始比较特定数据库对象的原始数据，以确定哪个行/哪个列不同。

[0134] 抽样再检验

[0135] 如上所述，合并形成是负责发现源表与目标表之间的差异的实际代理。将发现的所有这些差异插入存储器内队列QueueDiff。diffReporter线程处理该差异，并且然后，将结果记录在DIFF表中。

[0136] 在一个方面，轻量表比较采用活比较应用，该活比较应用将在此合并的未决专利申请的IBM Infosphere® 比较(ASNTDIFF)实用程序扩展到对正活动地更新的两个表进行比较，并且然后，将在块中检测到的任何差异进行再比较，以在正在比较块时，核算复制等待时间和可能对数据产生的在线变化。

[0137] 由于仅当正在比较一对块时既从源又从目标读出数据，所以降低了对包含在这些块中的行产生变化的概率。在该比较中包含比较应用访问这对块之前发生的任何变化，不

考虑对一组块的比较结束之后发生的任何变化。

[0138] 当复制延迟之后再检验时，如果数据仍不同，则考虑差异是持久性的，并且当第一次比较数据时，差异最初不由未提交事务导致。

[0139] 因此，配置用于比较广域网分离的非常大表的高度可伸缩性表比较应用，以假定在比较时对源表和目标表进行更新。因此，实用程序能够检测与异步复制处理等待时间导致的瞬间差异不同的持久性差异。在并行模式下，该方法包含将每个表划分为行的块，该行或者1)由在此描述的每个抽样技术产生，或者2)由分块列的值范围划界，或者3)块范围的源表和目标表中的行，与其他块的行中的进行独立比较。因此，并行执行多个块比较。在一个实施例中，通过考虑到行的大小，该方法确定块中的最佳行数。此外，在一个实施例中，表大小确定快的数量。对于该块的范围中的第一行和最后一行或者连续块的边界行，分块列值区分块。如本说明书的上面所述，安排块对，以对合并线程池服务的队列进行比较。每个合并器一次比较一对块。

[0140] 在一个实施例中，该方法将在块中检测到的任何差异进行再比较，以在正在比较该块时，核算复制等待时间和对数据产生的在线变化。不再访问当比较块时相同的数据。同样，对于整个表，在已经确定两行相同后，不再做比较。

[0141] 当比较活数据时，比较小块具有显著优势。因为仅当比较一对块时，既从源又从目标读出数据，所以包含在这些块中的行发生变化的概率降低。比较中包含比较应用访问该对之前的任何变化，而不考虑对该块完成比较之后发生的任何变化。

[0142] 正如在此要查阅的，提供下面的定义：

[0143] DOUBTFUL(D)：差异可以是持久性的，也可以非持久性的。当第一次比较后，差异相同时，其是最初持久性类型的。

[0144] UNKNOWN(U)：差异可以是持久性的，也可以是非持久性的。在活模式比较中，如果差异不是持久性的，则可能的原因只是禁止长时间提交事务或者利用游标稳定性(CS)(或者较高管理)提取的数据。

[0145] TIMEOUT(T)：差异可以是持久性的，也可以是非持久性的。达到再比较极限，并且行仍不同，因为相同行(即，具有相同键值的行)保持正在更新而非正在提交。比较极限由RETRY_COMPARE参数控制。

[0146] PERSISTENT(P)：差异是持久性的。当复制延迟之后再检验时，该数据仍存在差异，并且该差异是当第一次比较该数据时最初由未提交事务导致。

[0147] RETRY_COMPARE_WAIT＝minimum_wait_in_seconds：用于规定比估计的最大端到端等待时间和含有在该环境下正在比较的表的最长运行事务的时间之和大的值的外部参数。比较应用利用该值确定何时再比较差异，以为要复制的变化留够时间。即，能够在特定时间窗口内估计复制等待时间(END2END_LATENCY)，并且该等待时间能够用于确定RETRY_COMPARE_WAIT的值(请参见图8)。

[0148] 如果在其复制到目标之前实用程序再比较差异，则可能报告错误差异。在一个方面，实用程序可以返回，以仅比较先前比较中的差异。在一个例子中，RETRY_COMPARE_WAIT的默认值是2，并且最小值是1。

[0149] RETRY_COMPARE＝num_retries：用于规定实用程序将在放弃之前保持变化的行比较多少次的外部参数。如果达到比较极限并且行因为相同行保持更新仍不同，则将该差异

报告为DIFF表中的T(暂停比较)。在一个例子中,默认值是2,并且最小值是0。

[0150] 在一个实施例中,比较应用将未提交读出(UR)隔离级别用于查询数据,以避免与用户应用争用锁。在这种模式下,返回未提及数据,可以回滚该数据,因此,比较应用不包含UR模式下检测到的差异。利用游标稳定性(CS)或者较高隔离再检验差异,以保证回滚数据和飞行事务不包括在最终比较中。游标稳定性模式是允许共享锁布置于提取的行上,使得当提取另一行时或者当游标关闭时释放共享锁的隔离级别。允许另一个处理,以将共享锁布置于相同行上,但是不允许处理获取互斥锁,以修改行中的数据。

[0151] 如果未授权比较应用在游标稳定(CS)模式或者较高隔离级别下查询数据,则利用"U"报告其中一些可能是由回滚事务或者飞行事务导致的所有差异,表示对DIFF输出表的DIFF_PERSISTENCE列未知。利用参数NUMTHREADS能够调节同时CS读出的数量,该参数NUMTHREADS规定允许比较应用创建的线程的数量。每次CS读出仅调用一个键值。在一个实施例中,同时CS读出的最大数量等于NUMTHREADS/3-1。在一个实施例中,能够设定参数CS_READ=Y|N,以在允许共享锁布置于提取行上的CS隔离级别,防止比较应用发出任何SELECT语句,使得当提取另一行或者关闭游标时,释放共享锁。默认CS值是Y("是"),这意味着,允许利用CS隔离级别提取。在一个实施例中,通过在比较应用调用后执行CS模式查询,手动检验报告为未知的差异。

[0152] 为了有效再比较检测到的差异,通过记录检测到的每个差异的64位校验和(CRC),并且该值与新CCRC(基于比较的校验和)列相加,比较应用对检测到的每个差异保持CRC。确定了该差异的持久性类型后,将该差异和CCRC值记录到DIFF输出表中。每个CCRC值是对于相同键值位于源的行的32位校验和(CRC)和位于目标的行的32位校验和的并置。CCRC唯一地识别差异。

[0153] 图8是示出在用于复制时使用RETRY_COMPARE_WAIT时间令牌的时序图375的一个实施例。在图375中,源数据库表的最初数据变化示于378。异步复制不开始捕获或者重放数据变化,直到在诸如时间380指出的源表提交事务或者变化,从最初变化开始的时间表示"脏"UR读出间隔。对于UR数据中的数据提取,异步复制导致的变坏从诸如380的发出提供之前源表处的最初数据变化开始,并且在诸如387的在目标表发出提交之前的相应数据变化结束。因此,复制延迟,例如,复制端到端延迟还应当包含源表处的提交时间与目标表处的相应提交之前之间的相关事务的整个时间窗口。RETRY_COMPARE_WAIT参数表示在图8的382的第一UR提取与图8的388的第二UR提取之间示出的时长。应当将RETRY_COMPARE_WAIT的值设定为大于复制延迟的值。

[0154] 图9示出用于确定持久性类型为"怀疑"的特定差异的实际持久性类型的方法300。在一个实施例中,能够在活更新环境下进行该确定。在所示的实施例中,参数RETRY_COMPARE_WAIT=n秒,例如,利用1秒的默认值设定n,该值设定比较应用在再次对于差异比较行之前等待的时间。

[0155] 因此,在305,合并线程发现差异(i=1),例如,其中i是指出对于该差异已经执行的差异比较的次数的递增索引。将该差异的持久性类型标记为"怀疑"。

[0156] 经历了时间间隔RETRY_COMPARE_WAIT后,下一个步骤308对相应行执行再比较。因此,延迟之后,诸如在图8的时间388,比较应用再次在UR模式下提取行、再比较CRC。该再计算可以调用函数,以该(这些)行确定是否基于行的CRC@src==基于行的CRC@tgt。该方法

还执行使索引i递增1,以指出已经执行了又一个再比较。

[0157] 在312,如果匹配,即,确定CCRC相同,则仍存在相同差异。该差异不是持久性差异,如354所示。这可以由源处的长时间运行的未提交事务获得,或者因为有效复制延迟大于RETRY_COMPARE_WAIT秒。

[0158] 然后,在315,该方法对所执行的在先比较(UR模式下)与最后比较之间的差异进行比较。该再比较可以调用函数,以确定新计算的CCRC是否与存储器中的CCRC匹配,即,对于该行,是否基于计算的校验和CCRC@C＝＝CCRC@C(i-1)。

[0159] 在318,如果不匹配,即,确定CCRC差异在当前比较和在先比较之间存在匹配,则该处理进入321,在321,对于再试次数,关于是否当前比较计数i-1<＝RETRY_COMPARE极限进行确定。如果在321,确定未达到再试极限的次数,即,i-1<＝RETRY_COMPARE评估“是”,则该处理返回步骤308,在步骤308,执行进一步再比较,并且重复308之后的处理步骤。另外,如果已经达到再试极限,即,确定当前索引i-1<＝RETRY_COMPARE评估“否”,则确定persistence_type已经暂停,即,是值T。即,参数RETRY_COMPARE＝n规定如果已经达到比较极限并且因为相同行保持更新行仍不同,则在放弃之前,比较应用将对保持变化的行再比较多少次,在DIFF表中,对于暂停比较,将差异报告为“T”。

[0160] 返回步骤318,如果确定匹配,即,CCRC的差异在当前比较与在先比较之间不存在匹配,则该处理进入图9的步骤325,在步骤325,该方法确定是否已经将CS_READ参数设定为“是”(Y),即,指出CS隔离级别(游标稳定性)模式。如果在325,确定CS_READ参数已经设定为“是”,则在328,再次将CS或者较高隔离级别用于再比较,提取行。另外,在325,如果在325确定CS_READ参数未设定为“是”,则确定persistence_type未知,即,是值U。

[0161] 因此,利用在328正在再比较的行,在330,对于该(这些)行确定是否基于行的CRC@src＝＝基于行的CRC@tgt。如果匹配,则在354,确定该差异不是永久性的差异,并且对该差异赋予NOT永久性的指示符。因此,如果将行与CS比较并且不存在差异,则这意味着该差异由现在提交的或者复制的飞行事务或者该事务是回滚而导致。无论哪种方式,差异都消失。

[0162] 另外,在330,如果确定该行的基于行的CRC@src＝＝基于行的CRC@tgt不匹配,则该处理进入步骤335,在步骤335,通过比较在先比较(UR模式下)和最后比较之间的差异,再次执行基于比较的校验和比较,即,是否基于比较的校验和CCRC@C＝＝CCRC@C(i-1)。

[0163] 然后,在338,确定在比较在先比较(CS)与在先(i-1)比较之间的差异中是否匹配。如果确定不匹配,即,差异的相关行从最初比较发生变化,则该处理进入步骤321,在步骤321,通过对RETRY_COMPARE参数评估当前值i,关于当前再试比较级别,再次进行确定。根据321的该评估,该处理可以进入步骤308,在UR模式下进一步再比较,或者在已经达到RETRY_COMPARE极限的情况下,进入步骤351,在步骤351,以T(暂停)指出persistence_type。

[0164] 如果在338,基于比较的CCRC@C＝＝CCRC@C(i-1)导致匹配,则该处理进入步骤340,在步骤340,比较应用在间隔RETRY-COMPARE_WAIT之后再一次在UR模式下提取行、再比较CRC、以及将新CCRC与DIFF表中的CCRC比较。该再比较可以调用函数以对该(这些)行确定是否基于行的CRC@src＝＝基于行的CRC@tgt。

[0165] 如果在343确定匹配,则在345,确定该差异不是永久性差异,并且对该差异赋予NOT永久性的指示符。然而,如果在343确定不匹配,则该处理进入步骤347,在步骤347,再次对该比较与最后比较之间的差异进行比较,即,基于比较的校验和CCRC@C＝＝CCRC@C(i-1)

是否导致匹配。

[0166] 进入350,如果确定基于比较的校验和CCRC@C＝＝CCRC@(i-1)导致不匹配,则该处理返回步骤321,在步骤321,通过对RETRY_COMPARE参数评估当前值i,关于当前再试比较级别,再次进行确定。根据321的该评估,该处理可以进入步骤308,在UR下进一步再比较,或者在已经达到RETRY_COMPARE极限的情况下,进入步骤351,在步骤351,以T(暂停)指出persistence_type。

[0167] 另外,如果在350确定基于比较的校验和CCRC@C＝＝CCRC@(i-1)确实导致匹配,则该处理进入353,在353,将该差异的持久性类型指示为持久(P)。

[0168] 因此,正如在步骤328和340所指出的,当确定与CS和与CCRC的比较相同时,意味着最终提交源事务,而且尚未复制源事务。因此,RETRY_COMPARE_WAIT秒后直到RETRY_COMPARE极限,在UR模式下再比较该行。

[0169] 此外,如步骤318和338指出的,如果当CS比较并且与计算的CCRC比较时存在差异(不匹配),则行存在差异,但是从最后时间基于不同的值,指出例如已经再次在源更新了行,并且尚未复制到目标。在340,在UR模式下,经历了复制延迟后,再次比较数据。

[0170] 尽管未示出,但是DIFF表中的每个差异具有DIFF_TIME时间戳,该DIFF_TIME时间戳是为了最后比较提取数据的时间。用户能够使DIFF_TIME与MONITOR_TIME相关。如果此时END2END_LATENCY过大,即,比总再试时间长,则该差异因为该行仍没有被复制。能够再次运行比较应用,或者手动选择行,以比较它们。

[0171] 在一个实施例中,又一个参数RECHECK_DIFFS＝diff-schema-name.diff-table-name可以用于仅再比较在前面的DIFF输出表中报告为差异的行。

[0172] 图10示出如在此所述用于对持久性差异进行确定的系统400。在一个实施例中,如果当在复制延迟之后再检验时数据仍不同并且当第一次比较数据时最初由未提交事务导致差异,则该差异是持久性的。在diffReporter线程104确定persistence-type值。在第一差异卸载阶段,在图10的图中,QueueDiff 106是将差异从合并器合并线程发送到diffReporter的同步队列。合并线程98计算CCRC并且确定诸如在此参考图9描述的持久性类型。所有识别差异的最初持久性类型是“D”(怀疑)。当合并线程录制检测到的差异时,立即从QueueDiff 106读出“D”差异(持久性类型)105A。这是基于行的通知。diffReporter 104将差异存储于本地专用队列中,即,Insert_Queue 108中。在一个实施例中,当最大允许存储空间不够大,不能在Insert_Queue 108中保持所有差异时,创建溢出文件109。

[0173] 图10的图中的第二持久性差异识别阶段在从Insert_Queue 108中指出的差异的时间戳DIFF_TIME开始的RETRY_COMPARE_WAIT秒之后开始。diffReporter 104将每个差异105B插入同步队列,即,QueueDiffRecheck 116中。合并线程98的池消费该队列116。对于处理的每个差异105B,合并线程通过QueueDiff 106将再检验结果105C(持久性类型)送回diffReporter。

[0174] 当diffReporter线程104报告合并线程98检测到的差异时,,图10的图中的第三持久性差异报告阶段开始。diffReporter线程104保持读出QueueDiff,并且将能够确定其持久性类型(即,其差异类型是“U”(未知)、“T”(暂停)或者“P”(持久性的))的差异插入DIFF表126中。在图9中解释了详细确定方法。将具有持久性类型“D”(怀疑)的差异插入Insert_Queue 108中,由合并器做进一步比较;忽略具有类型“N”(非持久性的)的差异。对于用户仅

外部化类型“P”、“T”和“U”。为了确定实际持久性，DIFF输出表中的类型“U”要求利用CS做附加比较，并且DIFF输出表中的类型“T”要求使用UR的更多再比较。

[0175] 在一个实施例中，实用程序采用进度监视函数。在该实施例中，可以将JOBPROGRESSPERCT＝Y|N的参数值设定(为“否”)，并且用于消除对行计数的步骤。因此，输出消息，即，图12中所示的典型消息150不显示在153指出的完成百分比和在155指出的开始百分比。相反，百分比值仅由完成的行数和正在比较的/已经比较的行数代替，用于报告(例如，“完成10000行，开始15000行”)。当其基于块的校验和匹配时，不活模式和活模式都能够报告完成基于块的比较。如果不匹配，则在一次通过基于行的比较后，不活模式能够获取该结果，但是活模式要求以至少一个RETRY_COMPARE_WAIT间隔多次通过基于行的比较。因此，为了使活模式下比较进度不表现失效甚或使系统挂起，还在155报告基于块的比较开始的行计数的百分比。比较了最初基于块的统计结果/数据提取后，将基于块的比较看作“开始”。图12所示的典型消息150将基于块的比较消息显示为具有基于块的比较开始155的百分比或者行计数。最初基于块的统计结果/数据提取比较之后，将基于块的比较看作“开始”。

[0176] 完成块之后，改变特定块比较状况。当并且仅当已经确定了所有这些行的持久性类型时，完成特定块的比较。还可以同时比较多个块。在跟踪特定块的整个比较处理的一种实现中，可以使用散列表。当最初UR比较之后将具有持久性类型“D”的第一差异发送到diffReporter时，将该块插入散列表中。当确定所有这些怀疑差异的持久性类型时，从散列表中移除该块。

[0177] 此外，在基于散列的块状况表中的块状况发生变化之前，插入并且提交所有关联差异。此外，在一个实施例中，为了避免频繁提交或者大量提交，当一个块完成时，提交插入的差异，或者差异数是某个数，例如，50的倍数。

[0178] 在一个实施例中，在图10的图中，在主线程94执行了去除最初预处理阶段中的计数行后，总体性能得到改善。然而，当JOBPROGRESSPERCT＝Y时，仍要求源的计数行。该任务分配到diffReporter线程104，即，在活比较的最初阶段，对diffReporter线程引入潜在瓶颈。如果计数未完成，则diffReporter 104不能开始处理合并线程报告的差异。当类型“D”差异的数量大于diffQueue 106的大小时，阻塞合并线程98，直到diffQueue不满或者不限制存储的储存器的diffQueue变得可用。在一个例子中，当前硬编码值是10,000。

[0179] 现在参考实施例描述关于确定利用游标稳定性(CS)提取数据的进一步详情。当绑定存储过程并且关联通用封装时，变量UR(未提交读出)用作数据存取的隔离级别。隔离级别越高，则数据库(例如，DB2)必须获取的锁越多，其中可能影响到使其他应用变慢或者导致死锁状况。为了避免“脏”读出导致的差异，通过作为SQL语句的一部分清楚地陈述子句“WITH CS”，利用隔离级别CS(游标稳定性)提取数据。用于数据提取的CS游标正在利用“FOR READ ONLY”使用多行提取。收到负SQLCODE后，比较应用不再试查询语句。当错误代码指出1)当前工作单位因为死锁或者暂停已经回滚，代表应用进程发出ROLLBACK语句，撤销当前工作单位时的所有更新；或者2)死锁或者暂停导致未成功执行，未发出ROLLBACK语句，但是对于死锁，请求应用进程本身或者发出ROLLBACK语句或者终止中的一个时，通过提交并且然后报告具有持久性类型“U”的差异，实用程序释放锁。

[0180] 其他DBMS变量控制在暂停之前，在图9中的328，比较应用将等待CS读出的特定锁

的时间量。例如,在IBM DB2 z/OS上,比较在暂停之前等待锁的最大时间长度是IRLMRWT DSNZPARM值加DEADLOCK IRLM PROC值。

[0181] 运行于服务器平台,例如,IBM's z/OS上的典型数据库系统(IBM DB2)具有PREPARE子句USE CURRENTLY COMMITTED,以存取最后提交的数据版本(即,在"阻塞"工作单位已经改变该行但是尚未提交该改变之前现有的数据版本),而不等待写入器释放锁。这样能够改善诸如表比较应用的只读应用的性能。当数据处于正在更新或者删除的过程中时,实用程序能够获得提交数据(跳过处于正在插入处理中的行)。不幸的是,由于不是DB2 z/OS上的所有数据库都支持UPDATE,因此,对于任何阻塞UPDATES,CS读出事务必须等待COMMIT或者ROLLBACK操作。当正在进行UPDATE操作时,DB2 LUW支持同时对最后提交的先前版本的数据进行存取。然而,在一个实施例中,在如下条件下:当封装运行于上面的表空间不是通用表空间时,USE CURRENTLY COMMITTED仅适用于通用表空间;对于表、分区或者表空间锁;对于使用LOCK TABLE IN EXCLUSIVE MODE时;当锁保持器正在执行批量删除时;以及如果锁保持器已经升级,使用WAIT FOR OUTCOME行为。

[0182] 在实现DB2数据库的典型实施例中,利用表空间中定义的参数控制锁定行为。不正确的参数值能够导致资源频繁不可用,并且降低并发性。LOCKSIZE、LOCKMAX和MAXROWS是这样三个参数:LOCKSIZE PAGE参数是对性能和并发性的一种选择。为了在不增加数据共享开销的情况下实现行级锁定,可以利用MAXROWS 1迫使每行到其自己的页。MAXROWS 1增加了存储数据要求的页数并且消除了压缩的所有好处。

[0183] 为了实现高可用性,没有理由允许发生锁升级。锁升级的好处是,在减少保持于IRLM中的大量锁时,由DB2代表不释放其锁的不良设计的应用提供保护。因此,在一个实施例中,可以设定LOCKMAX=0值,以避免锁升级。

[0184] 另一个NUMLKTS子系统参数(LOCK PER TABLESPACE字段)规定CREATE TABLESPACE和ALTER TABLESPACE语句的LOCKMAX子句的默认值(处于子系统级别)。LOCKMAX的值应当小于NUMLKUS子系统参数(LOCK PER USER字段)。超过任何一个极限都能够导致资源不可用状况。当比较应用导致资源不可用状况时,应当首先将CS_READ参数的值设定为"N",并且然后,调查根本原因。

[0185] 在典型实现中,合并器因此具有两个职责:1)比较每个块并且将任何可疑差异候选对象送到diffReporter;以及2)逐个并且逐阶段再检验可疑差异(由diffReporter发送)。在活模式比较中,基于块的最初比较可以发现许多基于行的可疑差异。少量连续块中的可疑差异可以是许多,并且造成性能问题。为了解决该性能问题,如图11的系统图410所示,提供了称为"dMergers"398仅用于再检验单独差异的专用合并器,如创建dMerger线程块399所指出的。由于这些dMergers 398不比较表块,所以这些dMergers 398不增加使用,但是相反却改善性能。

[0186] 因此,作为例子,在图11中,当用户输入"LIVE=Y,NUMTHREADS=21,另一个PARALLELCMPS=2"时,比较应用将创建2个合并器((2*3),每个合并器将创建2个工作线程,1个diffReporter线程104、1个分区器线程95、1个主线程94。在该例中,假定关于线程数量的极限(例如,21个),仍可以创建21-9=12个线程。使用所有这些线程,可以对用户透明地创建4个以上diff-recheck-only合并线程398。

[0187] 在又一个示例性实施例中,参考图13,图13示出执行差异检测的详细过程的结果。

例如，用户能够从差异表获得整个过程。在一个实施例中，当且仅当差异是对差异块分配的具有相同键值的行时，详细过程和结果报告复写差异。

[0188] 因此，提供图13所示的差异检测输出表160，该差异检测输出表160具有：数据字段列：例如，指出块数(在Block_Num列字段166中指出)的数据字段列163；以及用于识别相关行(例如，该表具有一列键ID字段167)的键列；相关行的并且具有三种类型中的一种的相应差异类型(Diff_Type)：U(更新)、D(删除)和I(插入)；以及指出块的持久性状态(DIFF_IS_PERSISTENT)值的列165，例如，P(持久性的)、U(未知的)、T(暂停的)和D(怀疑的)或者非持久性的(N)；字段162值的计算的CCRC值及其相应时间戳值161。如图13的典型输出160所示，第一次比较后，全部头5行是候选对象差异。块ID是键列。特别是，一行168不指出检测到的差异，并且不是差异，而是从合并器到diffReporter的块1完成的信号。键值将是NULL。此外，在典型输出160中，利用指示为“C”(提交)的持久性字段值，行169示出提交读出之后，例如，第三次比较之后的差异。利用指示为“P”的持久性字段值，该输出的最后5行170示出持久性的差异。

[0189] 当不必需详细差异检测时，比较应用不报告差异检测过程。相反，仅报告每个差异的最终结果。这意味着，仅报告持久性类型P(持久性的)、U(未知的)和T(暂停的)的差异，并且将该差异插入输出DIFF表。

[0190] 在又一个实施例中，完成了所有比较后，可以配置diffReporter线程，以报告差异检测中的统计结果，如图14所示，该统计结果提供总值175，该总值175包含：从源表提取的总行数；差异总数(包含全部DIFF_IS_PERSISTENT值T、U、P)；复写差异的总数；具有DIFF_TYPE＝U(具有不同非键列)的差异的总数；具有DIFF_TYPE＝I(插入)类型(只有源)的总行数；具有DIFF_TYPE＝D(删除)类型(只有目标)的总行数；CS读出的总数；其DIFF_PERSISTENCE是值T(暂停)的总行数；再比较之后其差异不是持久性差异的总行数；其差异DIFF_PERSISTENCE是值P(持久性的)的总行数；其差异DIFF_PERSISTENCE是值U(未知的)的总行数；以及再比较的总数(其值不计算最初比较)。

[0191] 图3示出计算系统500的示例性硬件配置的一个实施例，对该计算系统500编程，以执行图4A－7和9所示的实现轻量表比较的方法步骤和在此参考图4A－4B、5－7和9描述的持久性差异识别。硬件配置优选地具有至少一个处理器或者中央处理单元(CPU)511。CPU 511通过系统总线512与：随机存取存储器(RAM)514、只读存储器(ROM)516、输入/输出(I/O)适配器518(用于将诸如磁盘单元521和磁带驱动器540的外围设备连接到总线512)、用户接口适配器522(用于将键盘524、鼠标526、扬声器528、麦克风532和/或者其他用户接口设备连接到总线512)、用于将系统500连接到数据处理网络、因特网、内联网、局域网(LAN)等等的通信适配器534、以及用于将总线512连接到显示设备538和/或者打印机539(例如，数字打印机等等)的显示器适配器536互连。

[0192] 本发明可以是系统、方法和/或计算机程序产品。计算机程序产品可以包括计算机可读存储介质，其上载有用于使处理器实现本发明的各个方面的计算机可读程序指令。

[0193] 计算机可读存储介质可以是可以保持和存储由指令执行设备使用的指令的有形设备。计算机可读存储介质例如可以是－－但不限于－－电存储设备、磁存储设备、光存储设备、电磁存储设备、半导体存储设备或者上述的任意合适的组合。计算机可读存储介质的更具体的例子(非穷举的列表)包括：便携式计算机盘、硬盘、随机存取存储器(RAM)、只读存

储器(ROM)、可擦式可编程只读存储器(EPROM或闪存)、静态随机存取存储器(SRAM)、便携式压缩盘只读存储器(CD-ROM)、数字多功能盘(DVD)、记忆棒、软盘、机械编码设备、例如其上存储有指令的打孔卡或凹槽内凸起结构、以及上述的任意合适的组合。这里所使用的计算机可读存储介质不被解释为瞬时信号本身,诸如无线电波或者其他自由传播的电磁波、通过波导或其他传输媒介传播的电磁波(例如,通过光纤电缆的光脉冲)、或者通过电线传输的电信号。

[0194] 这里所描述的计算机可读程序指令可以从计算机可读存储介质下载到各个计算/处理设备,或者通过网络、例如因特网、局域网、广域网和/或无线网下载到外部计算机或外部存储设备。网络可以包括铜传输电缆、光纤传输、无线传输、路由器、防火墙、交换机、网关计算机和/或边缘服务器。每个计算/处理设备中的网络适配卡或者网络接口从网络接收计算机可读程序指令,并转发该计算机可读程序指令,以供存储在各个计算/处理设备中的计算机可读存储介质中。

[0195] 用于执行本发明操作的计算机程序指令可以是汇编指令、指令集架构(ISA)指令、机器指令、机器相关指令、微代码、固件指令、状态设置数据、或者以一种或多种编程语言的任意组合编写的源代码或目标代码,所述编程语言包括面向对象的编程语言—诸如Smalltalk、C++等,以及常规的过程式编程语言—诸如“C”语言或类似的编程语言。计算机可读程序指令可以完全地在用户计算机上执行、部分地在用户计算机上执行、作为一个独立的软件包执行、部分在用户计算机上部分在远程计算机上执行、或者完全在远程计算机或服务器上执行。在涉及远程计算机的情形中,远程计算机可以通过任意种类的网络—包括局域网(LAN)或广域网(WAN)—连接到用户计算机,或者,可以连接到外部计算机(例如利用因特网服务提供商来通过因特网连接)。在一些实施例中,通过利用计算机可读程序指令的状态信息来个性化定制电子电路,例如可编程逻辑电路、现场可编程门阵列(FPGA)或可编程逻辑阵列(PLA),该电子电路可以执行计算机可读程序指令,从而实现本发明的各个方面。

[0196] 这里参照根据本发明实施例的方法、装置(系统)和计算机程序产品的流程图和/或框图描述了本发明的各个方面。应当理解,流程图和/或框图的每个方框以及流程图和/或框图中各方框的组合,都可以由计算机可读程序指令实现。

[0197] 这些计算机可读程序指令可以提供给通用计算机、专用计算机或其它可编程数据处理装置的处理器,从而生产出一种机器,使得这些指令在通过计算机或其它可编程数据处理装置的处理器执行时,产生了实现流程图和/或框图中的一个或多个方框中规定的功能/动作的装置。也可以把这些计算机可读程序指令存储在计算机可读存储介质中,这些指令使得计算机、可编程数据处理装置和/或其他设备以特定方式工作,从而,存储有指令的计算机可读介质则包括一个制造品,其包括实现流程图和/或框图中的一个或多个方框中规定的功能/动作的各个方面的指令。

[0198] 也可以把计算机可读程序指令加载到计算机、其它可编程数据处理装置、或其它设备上,使得在计算机、其它可编程数据处理装置或其它设备上执行一系列操作步骤,以产生计算机实现的过程,从而使得在计算机、其它可编程数据处理装置、或其它设备上执行的指令实现流程图和/或框图中的一个或多个方框中规定的功能/动作。

[0199] 附图中的流程图和框图显示了根据本发明的多个实施例的系统、方法和计算机程

序产品的可能实现的体系架构、功能和操作。在这点上，流程图或框图中的每个方框可以代表一个模块、程序段或指令的一部分，所述模块、程序段或指令的一部分包含一个或多个用于实现规定的逻辑功能的可执行指令。在有些作为替换的实现中，方框中所标注的功能也可以以不同于附图中所标注的顺序发生。例如，两个连续的方框实际上可以基本并行地执行，它们有时也可以按相反的顺序执行，这依所涉及的功能而定。也要注意的是，框图和/或流程图中的每个方框、以及框图和/或流程图中的方框的组合，可以用执行规定的功能或动作的专用的基于硬件的系统来实现，或者可以用专用硬件与计算机指令的组合来实现。

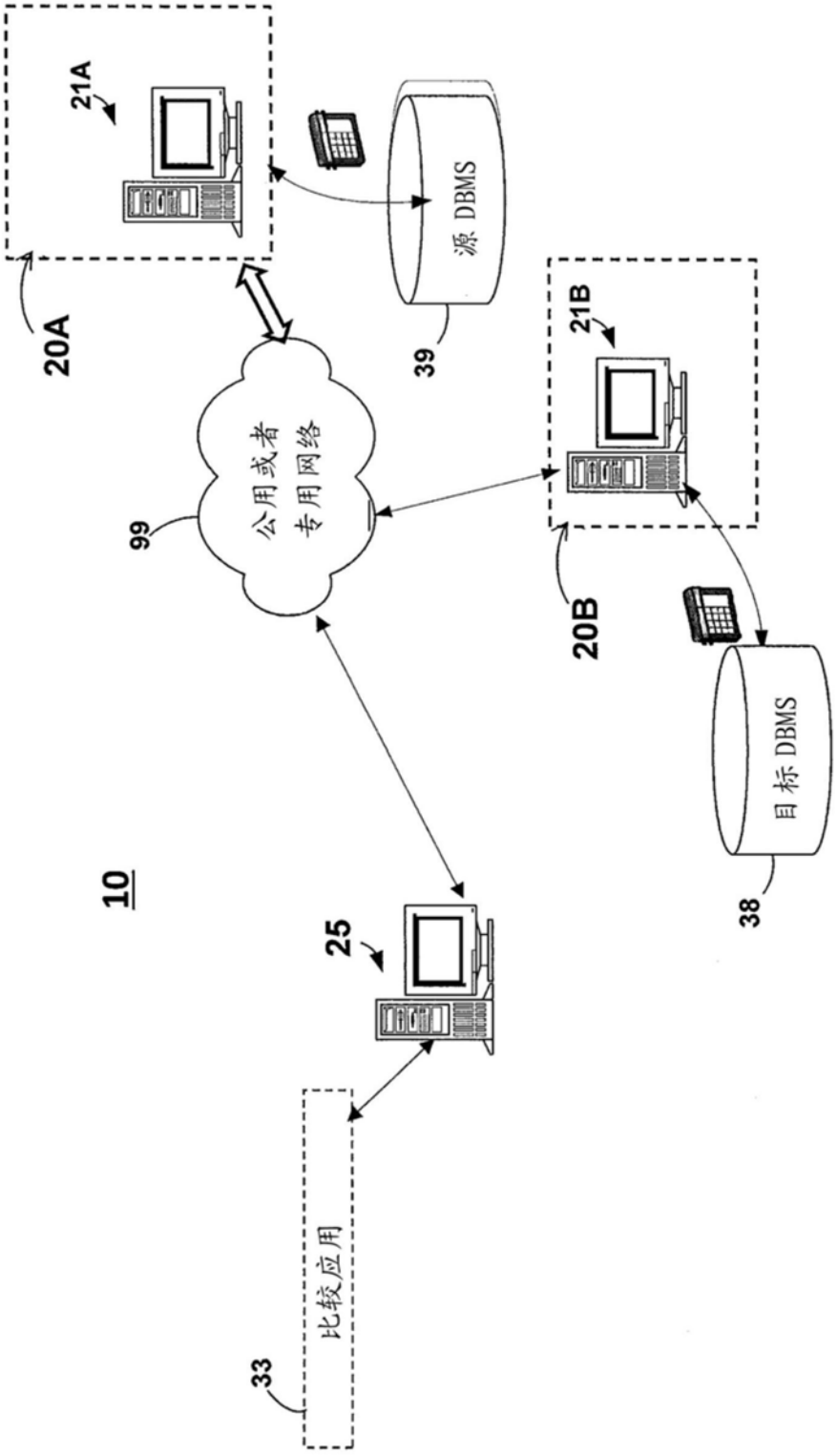


图1

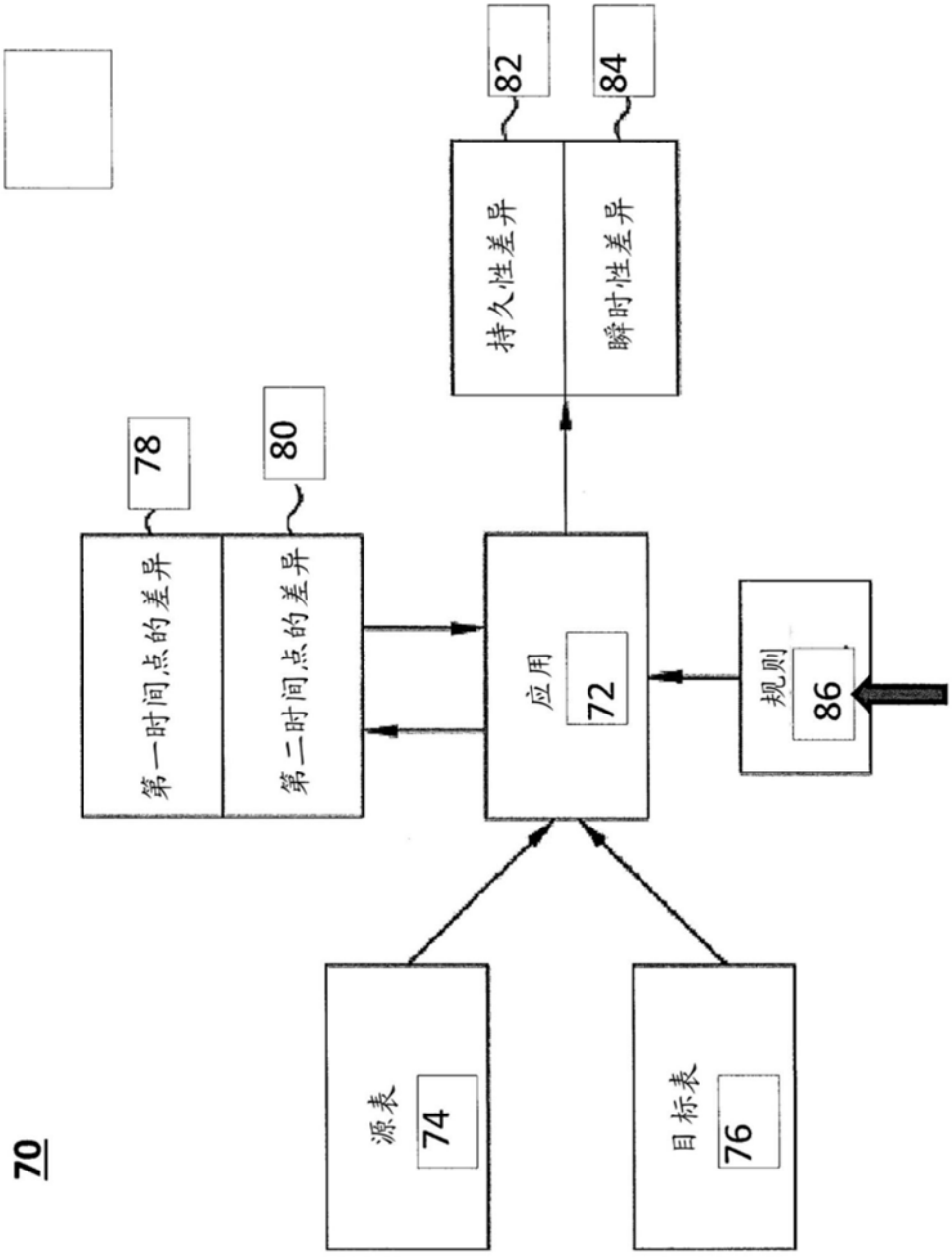

70
源表
74
目标表
76
应用
72
规则
86
第一时间点的差异
78
第二时间点的差异
80
持久性差异
82
瞬时性差异
84

图2A

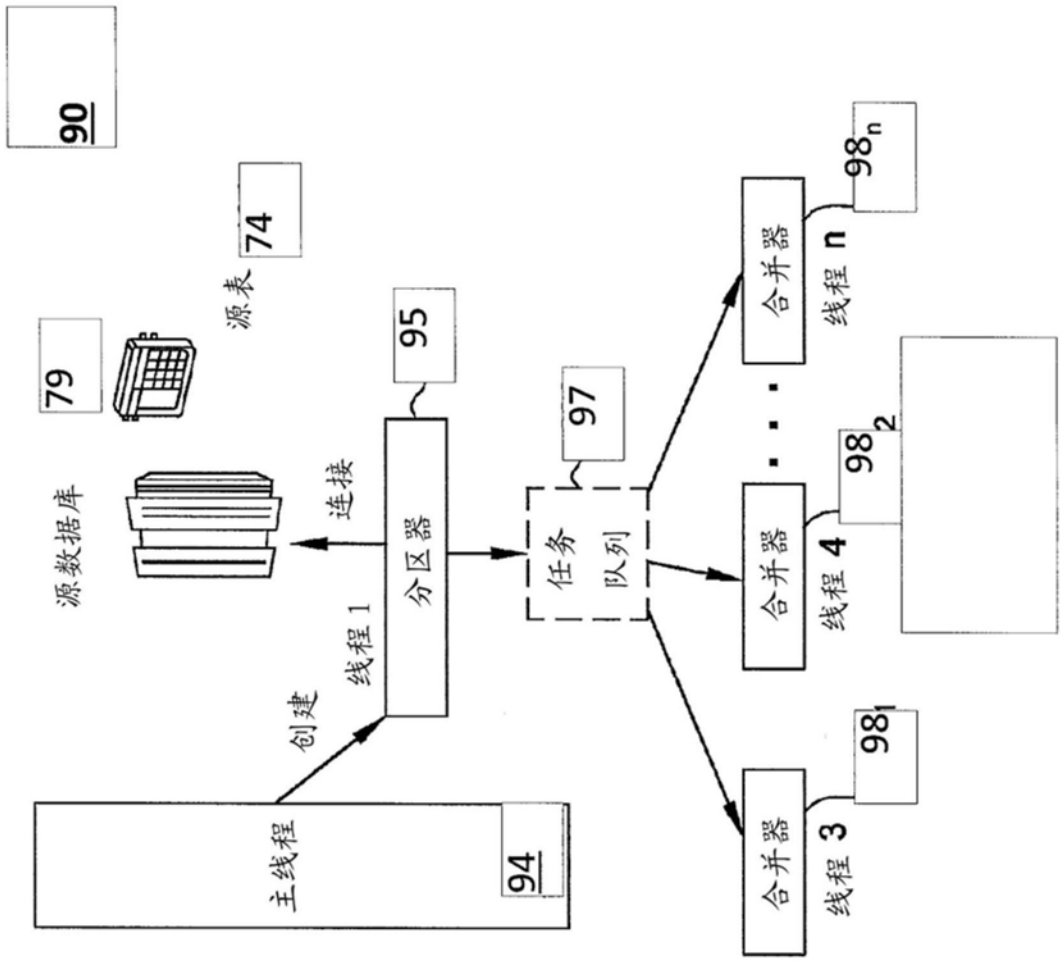

90
74
源表
79
95
源数据库
连接
分区器
97
任务
队列
合并器
线程 n
98n
98 2
合并器
线程 4
创建
线程 1
98 1
合并器
线程 3
主线程
94

图2B

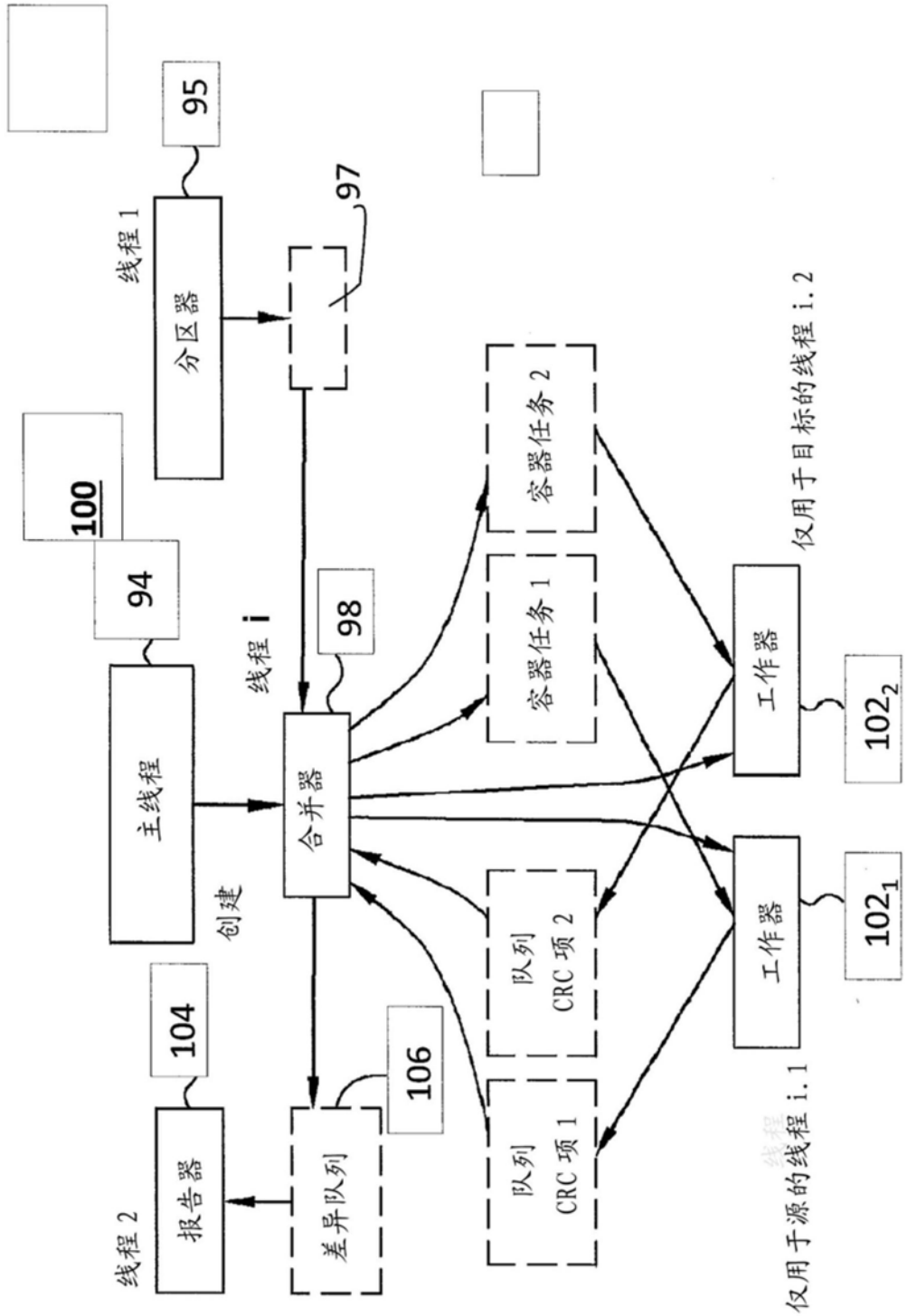

95
线程 1
分区器
97
100
94
主线程
创建
线程 i
98
合并器
容器任务 2
容器任务 1
工作器
仅用于目标的线程 i.2
102_2
队列
CRC 项 2
队列
CRC 项 1
工作器
102_1
仅用于源的线程 i.1
104
106
线程 2
报告器
差异队列

图2C

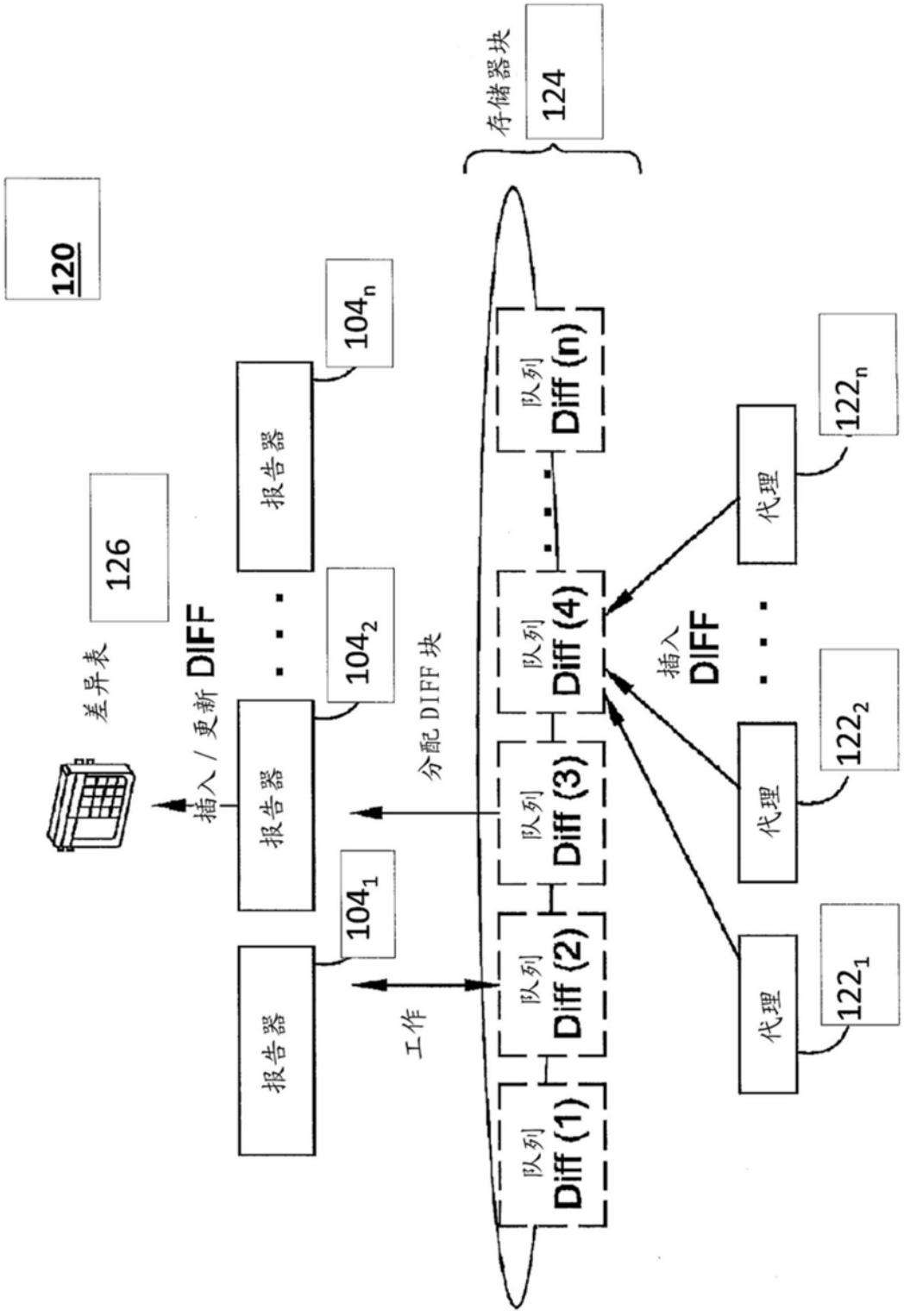

120
126
差异表
插入 / 更新 DIFF
报告器
报告器
报告器
104_n
104_2
104_1
分配 DIFF 块
工作
存储器块
124
队列 Diff (n)
队列 Diff (4)
队列 Diff (3)
队列 Diff (2)
队列 Diff (1)
插入 DIFF
代理
代理
代理
122_n
122_2
122_1

图2D

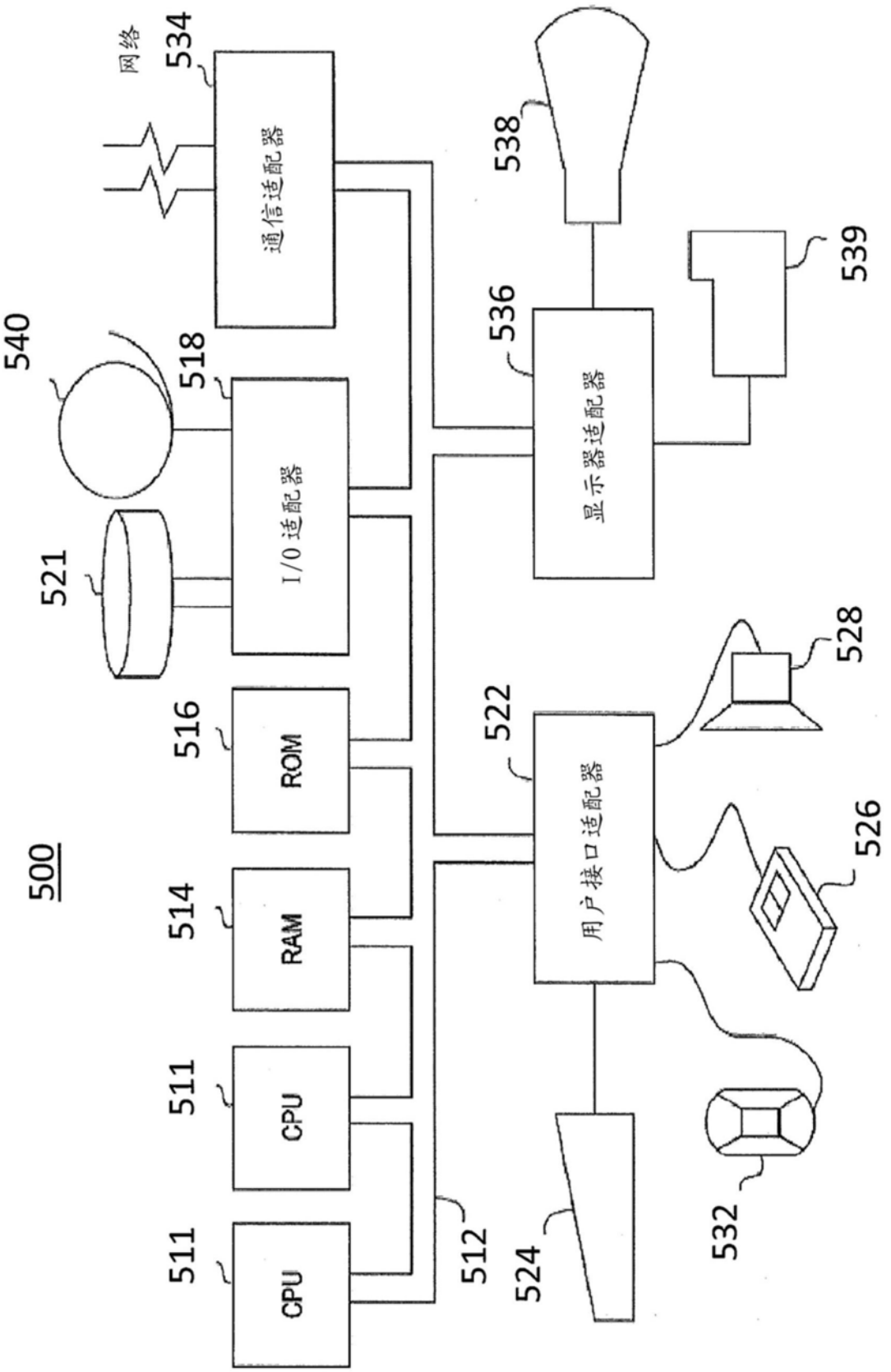

500
CPU
511
CPU
511
RAM
514
ROM
516
I/O 适配器
518
通信适配器
534
网络
显示器适配器
536
用户接口适配器
522
512
538
539
540
521
524
526
528
532

图3

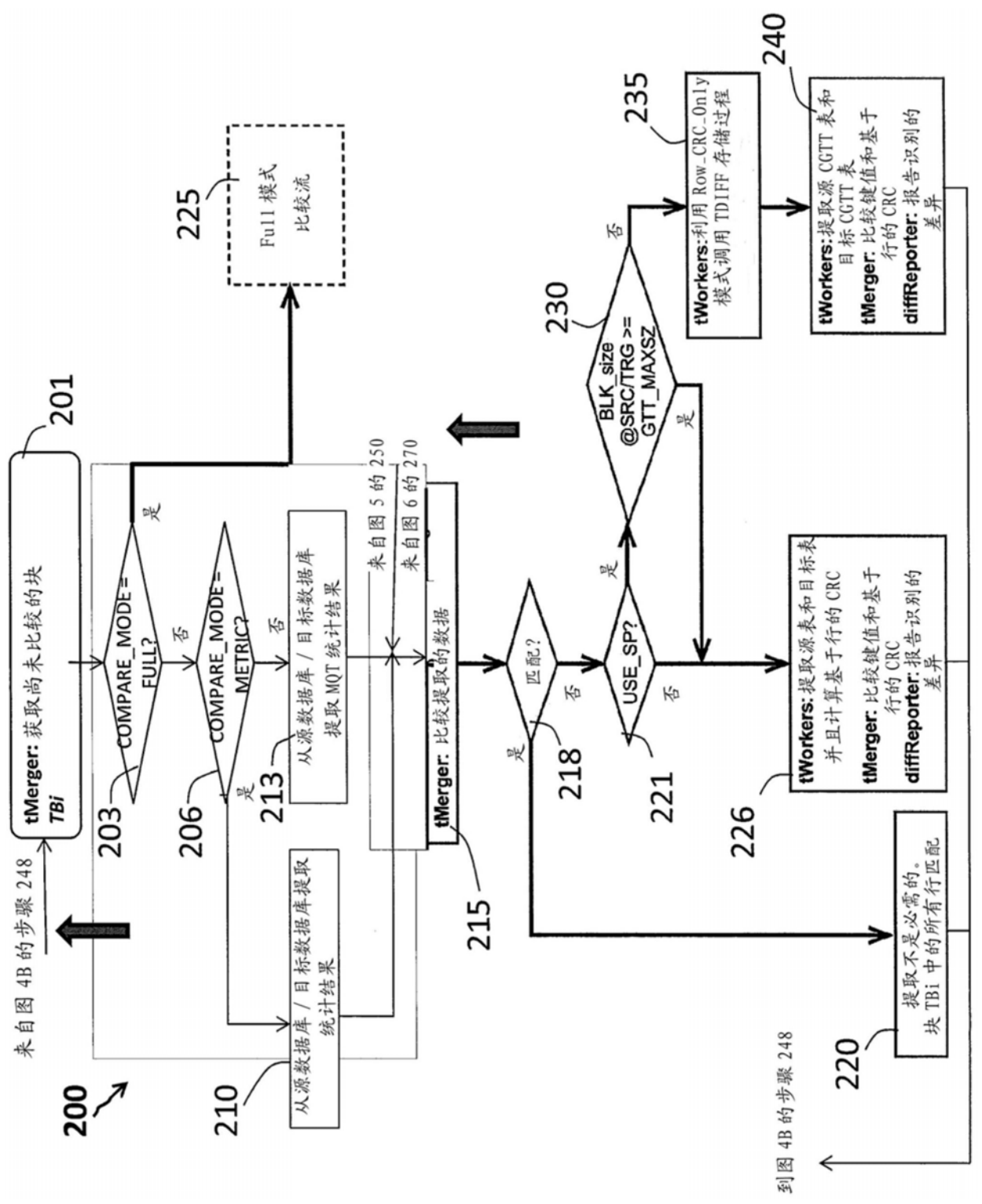

来自图 4B 的步骤 248
200
201
tMerger: 获取尚未比较的块
TBi
203
COMPARE_MODE = FULL?
是
否
225
Full 模式
比较流
206
COMPARE_MODE = METRIC2
是
否
213
从源数据库 / 目标数据库
提取 MQT 统计结果
210
从源数据库 / 目标数据库提取
统计结果
来自图 5 的 250
来自图 6 的 270
tMerger: 比较提取的数据
215
匹配?
是
否
218
USE_SP?
是
否
221
BLK_size @SRC/TRG >= GTT_MAXSZ
是
否
230
tWorkers: 提取源表和目标表
并且计算基于行的 CRC
tMerger: 比较键值和基于
行的 CRC
diffReporter: 报告识别的
差异
226
tWorkers: 利用 Row_CRC_Only
模式调用 TDIFF 存储过程
235
tWorkers: 提取源 CGTT 表和
目标 CGTT 表
tMerger: 比较键值和基于
行的 CRC
diffReporter: 报告识别的
差异
240
提取不是必需的。
块 TBi 中的所有行匹配
220
到图 4B 的步骤 248

图4A

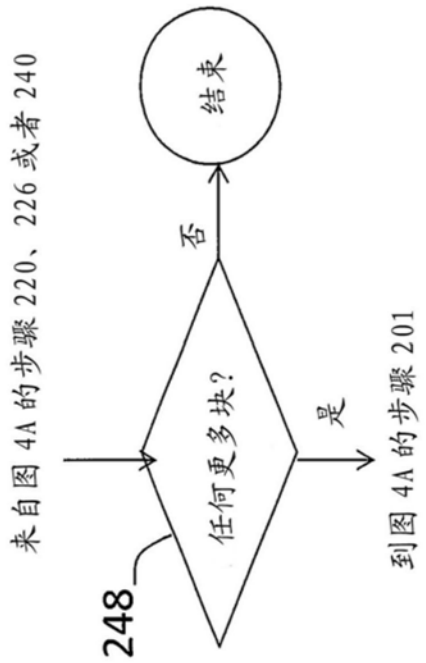


图4B

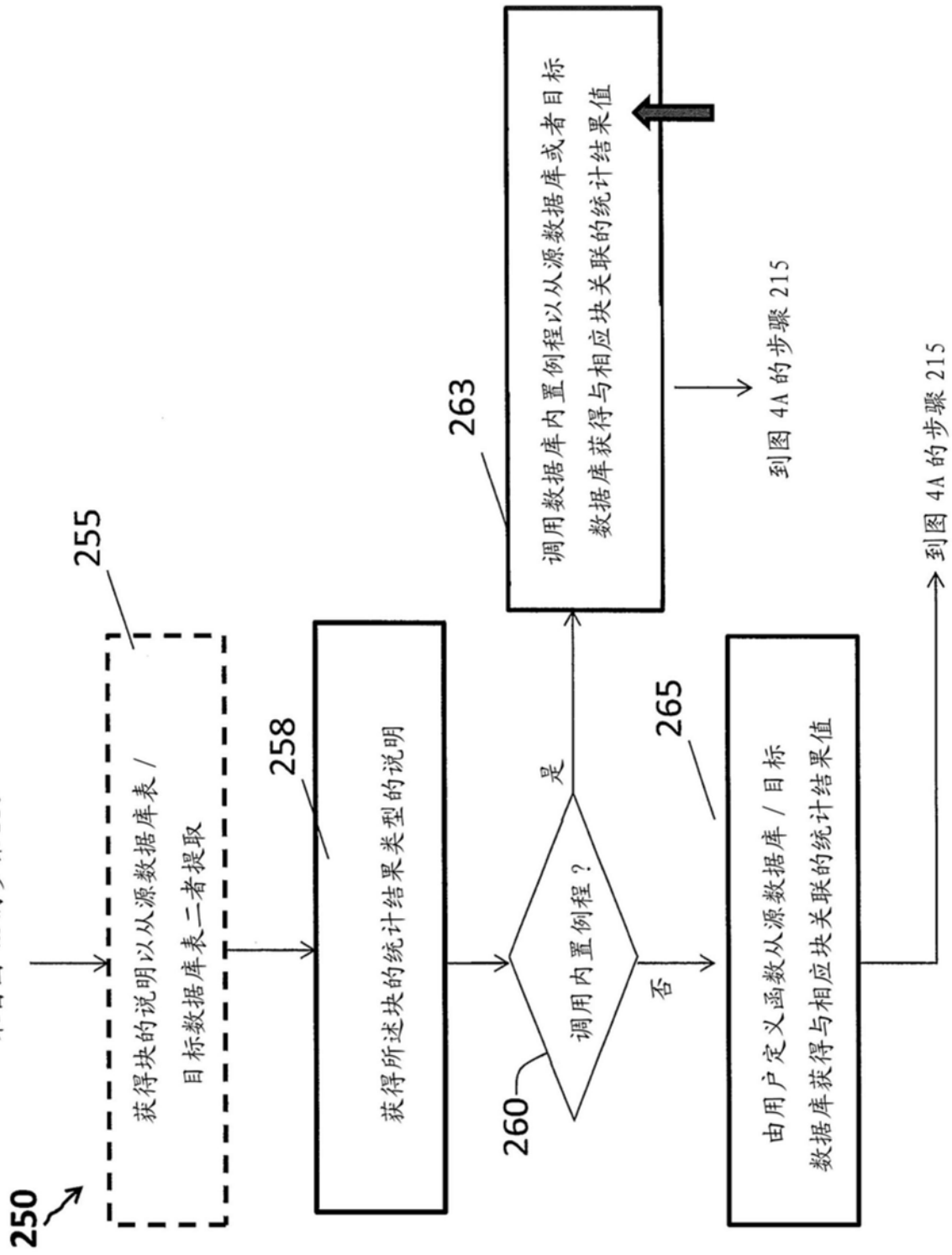

来自图 4A 的步骤 210
250
255
获得块的说明以从源数据库表 /
目标数据库表二者提取
258
获得所述块的统计结果类型的说明
260
调用内置例程？
是
否
263
调用数据库内置例程以从源数据库或者目标
数据库获得与相应块关联的统计结果值
265
由用户定义函数从源数据库 / 目标
数据库获得与相应块关联的统计结果值
到图 4A 的步骤 215
到图 4A 的步骤 215

图5

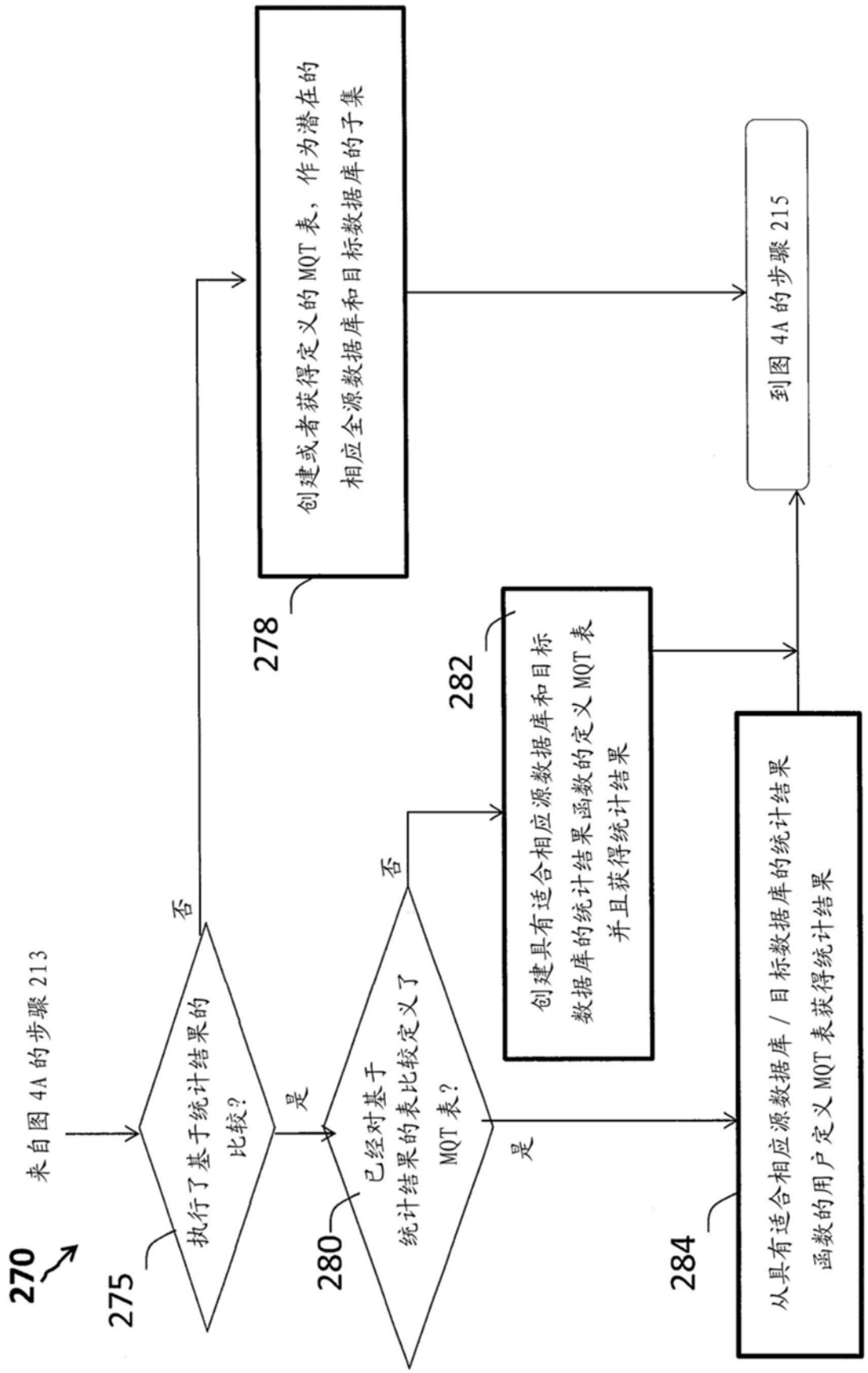

270
来自图 4A 的步骤 213
275
执行了基于统计结果的比较?
否
是
280
已经对基于统计结果的表比较定义了 MQT 表?
否
是
278
创建或者获得定义的 MQT 表，作为潜在的相应全源数据库和目标数据库的子集
282
创建具有适合相应源数据库和目标数据库的统计结果函数的定义 MQT 表并且获得统计结果
284
从具有适合相应源数据库 / 目标数据库的统计结果函数的用户定义 MQT 表获得统计结果
到图 4A 的步骤 215

图6

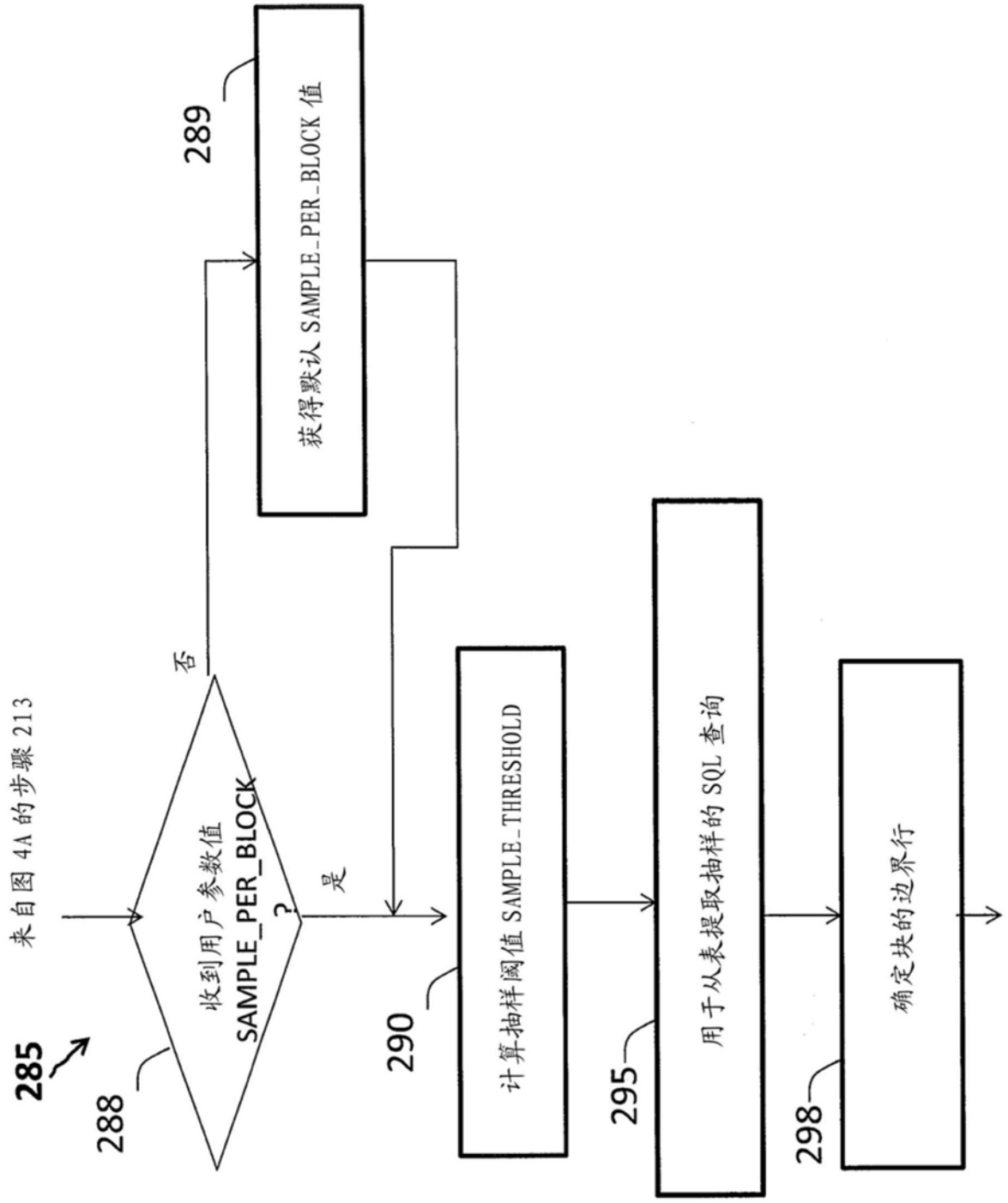

来自图 4A 的步骤 213
285
288
收到用户参数值 SAMPLE_PER_BLOCK ?
否
是
289
获得默认 SAMPLE_PER_BLOCK 值
290
计算抽样阈值 SAMPLE_THRESHOLD
295
用于从表提取抽样的 SQL 查询
298
确定块的边界行

图7

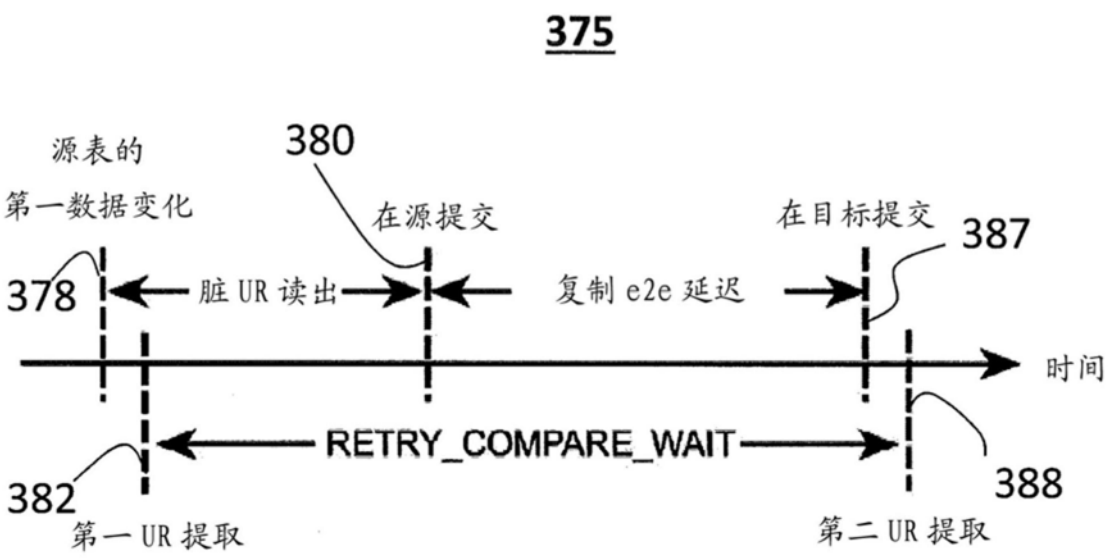

375
源表的
第一数据变化
380
在源提交
在目标提交
387
378
脏 UR 读出
复制 e2e 延迟
时间
RETRY_COMPARE_WAIT
382
388
第一 UR 提取
第二 UR 提取

图8

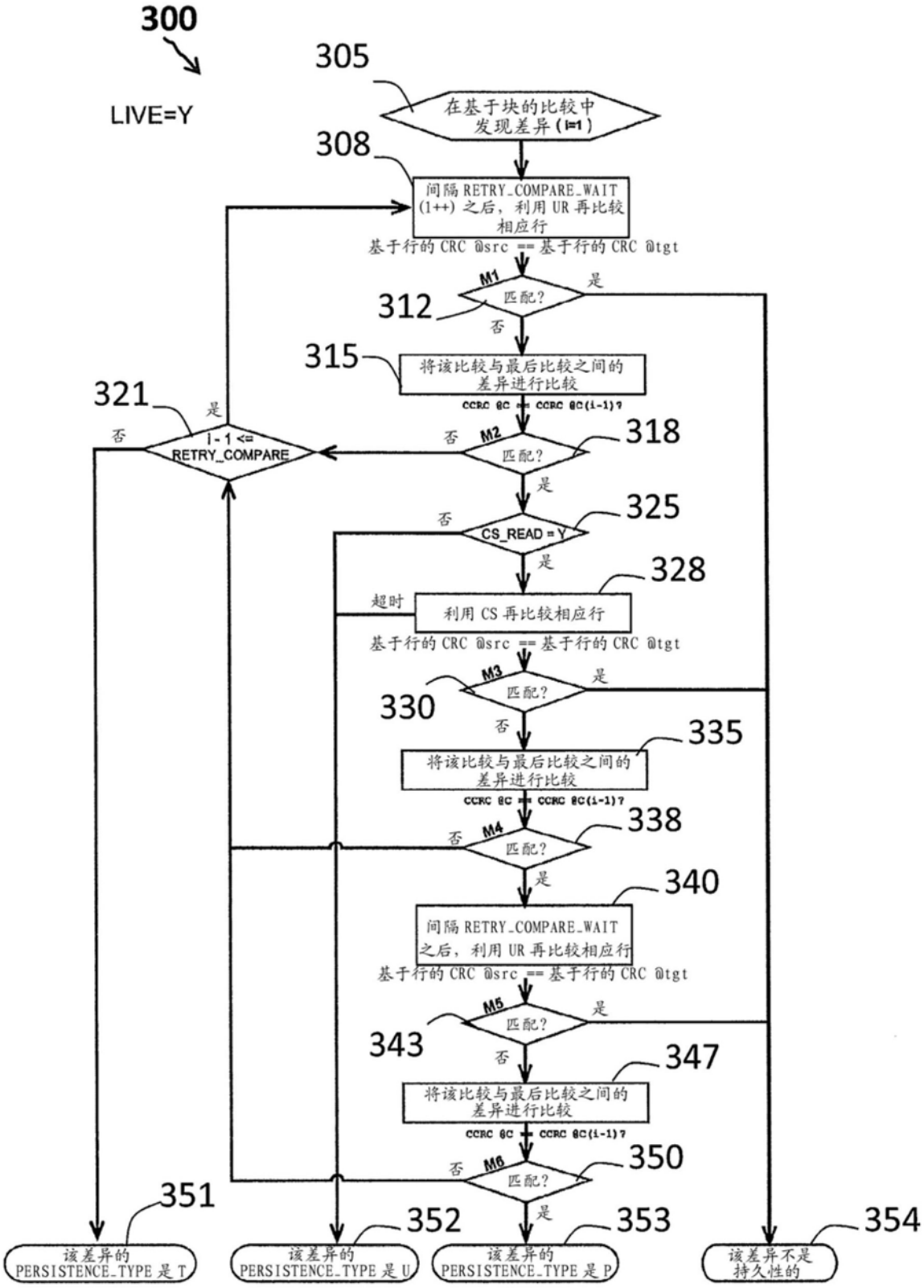

300
LIVE=Y
305
在基于块的比较中发现差异(i=1)
308
间隔 RETRY_COMPARE_WAIT (i++) 之后，利用 UR 再比较相应行
基于行的 CRC @src == 基于行的 CRC @tgt
M1
匹配?
是
否
312
315
将该比较与最后比较之间的差异进行比较
CCRC @C == CCRC @C(i-1)?
M2
匹配?
否
是
318
321
i - 1 <= RETRY_COMPARE
否
是
CS_READ = Y
否
是
325
328
超时
利用 CS 再比较相应行
基于行的 CRC @src == 基于行的 CRC @tgt
M3
匹配?
是
否
330
335
将该比较与最后比较之间的差异进行比较
CCRC @C == CCRC @C(i-1)?
M4
匹配?
否
是
338
340
间隔 RETRY_COMPARE_WAIT 之后，利用 UR 再比较相应行
基于行的 CRC @src == 基于行的 CRC @tgt
M5
匹配?
是
否
343
347
将该比较与最后比较之间的差异进行比较
CCRC @C == CCRC @C(i-1)?
M6
匹配?
否
是
350
351
352
353
354
该差异的 PERSISTENCE_TYPE 是 T
该差异的 PERSISTENCE_TYPE 是 U
该差异的 PERSISTENCE_TYPE 是 P
该差异不是持久性的

图9

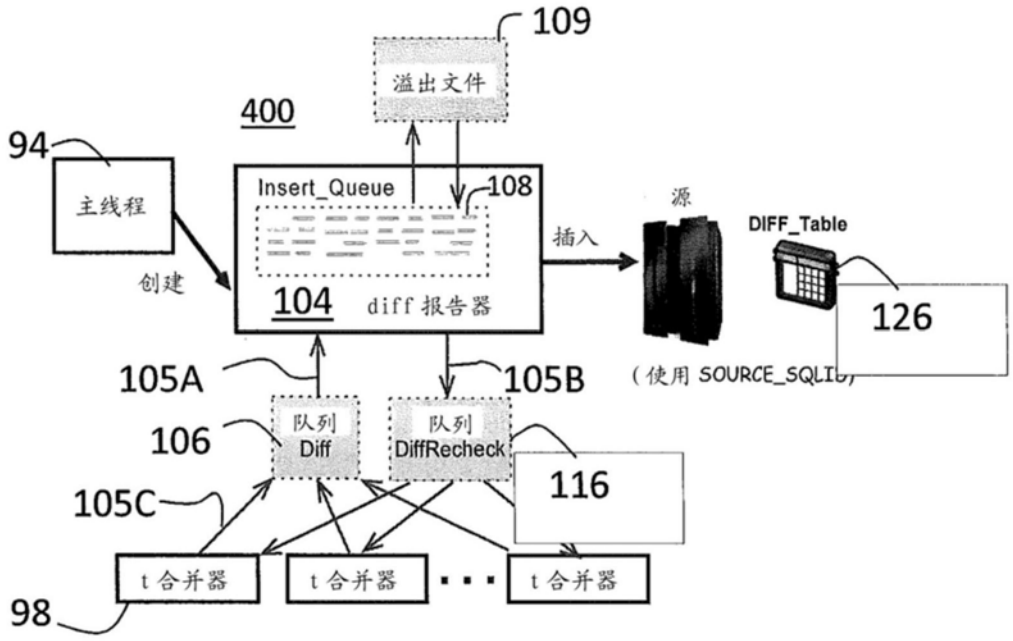

109
溢出文件
400
94
主线程
Insert_Queue
108
源
插入
DIFF_Table
创建
104 diff 报告器
126
105A
105B
(使用 SOURCE_SQLID)
106
队列
Diff
队列
DiffRecheck
116
105C
t 合并器
t 合并器
t 合并器
98

图10

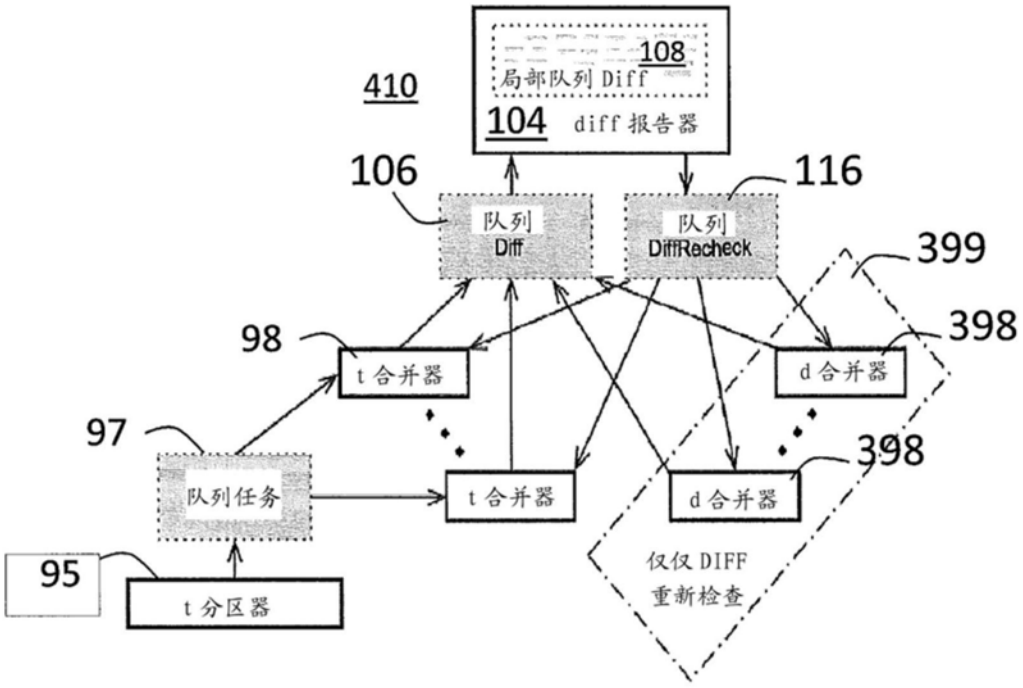

108
局部队列 Diff
410
104 diff 报告器
106
116
队列
Diff
队列
DiffRecheck
399
98
398
t 合并器
d 合并器
97
398
队列任务
t 合并器
d 合并器
95
仅仅 DIFF
重新检查
t 分区器

图11

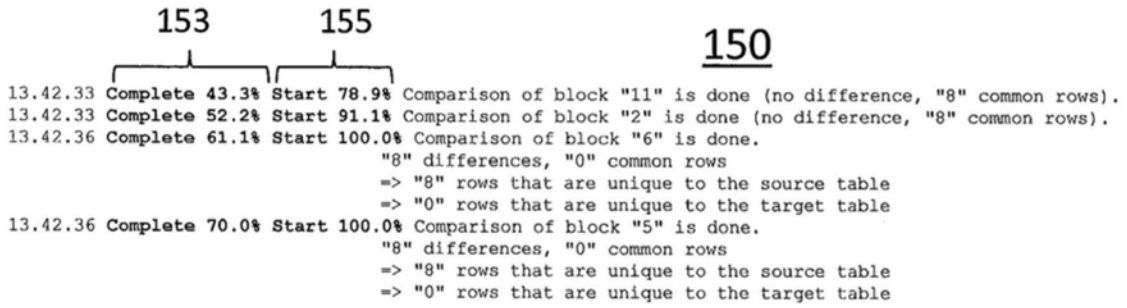


图12

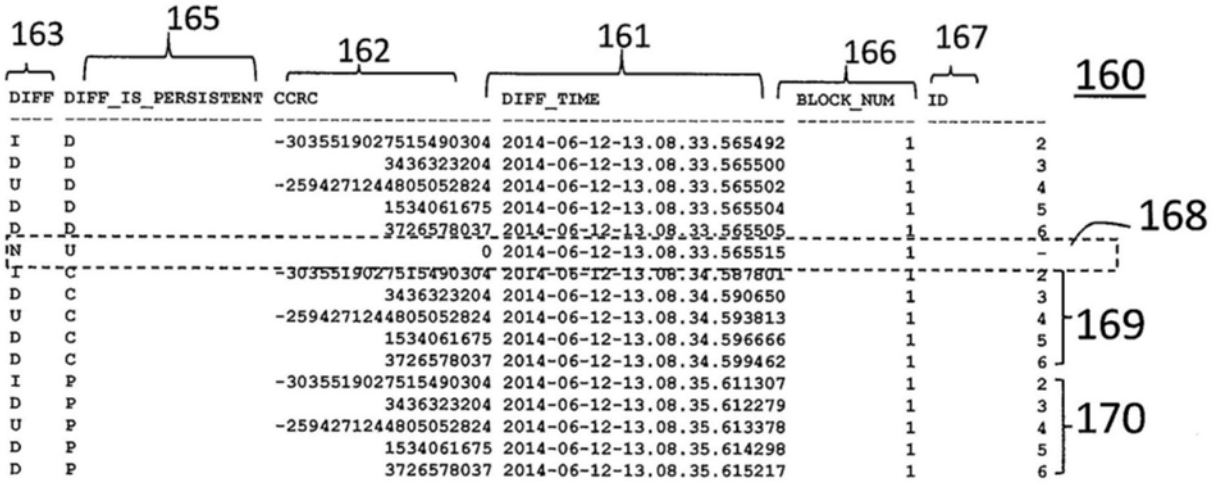


图13

175

```
Num of total rows at source: 12.// number of rows in source by count(*)
Num differences: 11.// number of total differences (including all DIFF_IS_PERSISTENT: T,U,P)
Num duplicated : 1. // number of duplicated differences
Num updates    : 1. // number of updates (DIFF_TYPE=U)
Num source only: 7. // number of source only (DIFF_TYPE=I)
Num target only: 3. // number of target only (DIFF_TYPE=D)
Num CS reads   : 8. // number of CS Reads
Num timeout    : 0. // number of differences (whose DIFF_PERSISTENCE is T)
Num temp       : 0. // number of differences that are not a persistent differences after recompares
Num persistent : 8. // number of differences (whose DIFF_PERSISTENCE is P)
Num unknown    : 3. // number of differences (whose DIFF_PERSISTENCE is U)
Num recompares : 0. // number of re-compares (the number does not count the initial compares)
```

图14