* 第十四章 分布式数据库系统

由于计算机网络通信的迅速发展以及地理上分散的公司、团体和组织对数据库更为广泛的应用的需求，20 世纪 80 年代，在集中式数据库系统成熟技术的基础上产生和发展了分布式数据库系统(Distributed Database System)。分布式数据库系统是数据库技术和网络技术两者相互渗透和有机结合的结果。

这一章介绍分布式数据库系统的基本概念，由于分布式数据库系统比集中式数据库系统更复杂，在有限的篇幅中只能论述最基本的概念和技术，作为今后进一步学习分布式数据库系统的基础。

14.1 概 述

14.1.1 分布式数据库系统

什么样的一个数据库系统才算是分布式数据库系统呢？一个粗略的定义是：

“分布式数据库由一组数据组成，这些数据物理上分布在计算机网络的不同结点(亦称场地)上，逻辑上是属于同一个系统”。

这个定义强调了下面两点：

(1) 分布性。数据库中的数据不是存储在同一场地上，更确切地讲，不存储在同一计算机的存储设备上，这就可以和集中式数据库相区别。

(2) 逻辑整体性。这些数据逻辑上是互相联系的，是一个整体(逻辑上如同集中数据库)。这就可以和分散在计算机网络不同结点上的那些集中式数据库或文件的集合相区别。后者各结点的数据之间没有内在的逻辑联系。所以在讨论分布式数据库时就有了全局数据库(逻辑上)和局部数据库(物理上)的概念。

但是，这个定义仍然是不精确的，下面来分析两个例子。

[例 1] 如图 14.1 所示，这个系统中有 3 台服务器，每台服务器有自己的数据库系统，3 台服务器之间通过网络相连，每台服务器有自己的若干客户机。用户可以通过客户机对本

地服务器中的数据库执行某些应用(称之为局部应用),也可以通过客户机对两个或两个以上结点中的数据库执行某些应用(称之为全局应用或分布应用)。这样的系统是分布式数据库系统,而不支持全局应用的系统不能称为分布式数据库系统。一个典型的全局应用的例子是银行转账。这个应用要求从一个分行的账户(设在 DB1 数据库)中转移若干金额到另一个分行的账户(设在 DB3 数据库)中去,因此要同时更新两个结点上的数据库。

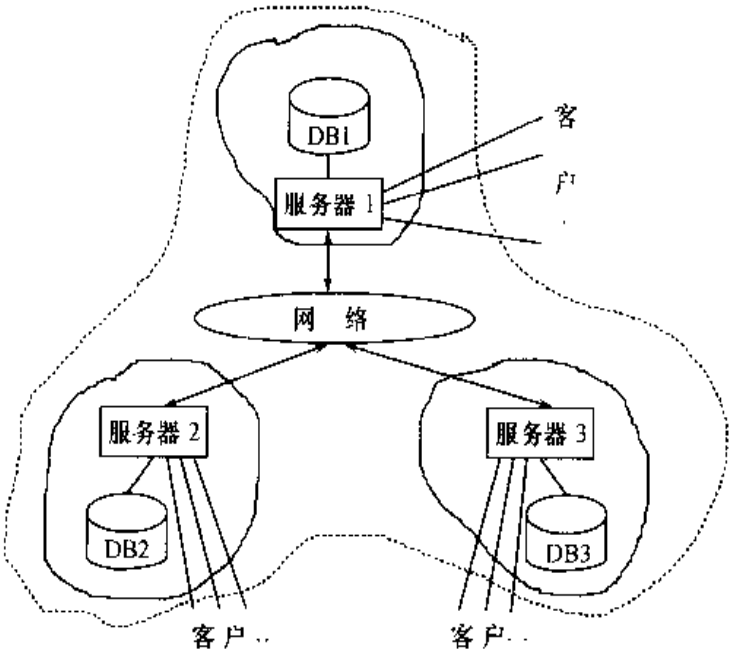


图 14.1 一个分布式数据库系统

这个例子说明,区分一个系统是若干集中式数据库的简单连网还是分布式数据库系统的技术要点在于,系统是否支持全局应用。所谓全局应用是指应用中涉及两个或两个以上结点的数据库。它和只存取本结点数据库的应用不同,和只存取另一个结点(或远程结点)数据库的应用也是不同的。支持全局应用的技术要复杂困难得多。在图 14.1 中用实线示意局部数据库及局部应用,用虚线示意全局数据库及全局应用。

[例 2] 如图 14.2 所示,这个系统的配置和图 14.1 的不同点是:3 个数据库通过后台服务器和网络相连。服务器执行数据库管理功能。所有的应用都由客户机处理。尽管发出这些应用的客户可以分布在不同的场地,数据库也物理地分布在各个不同的服务器上,但这样的系统不是分布式数据库系统而是一个多处理机系统(SN 并行结构),其原因是没有局部应用,即每个后台服务器不能由它本身来执行自己的局部应用,数据的物理分布不是对应用观点而言的分布,而分布式数据库不仅要求数据的物理分布,而且要求这种分布是面向处理、面向应用的。

总结上面两个例子,可以得出分布式数据库的一个更确切的定义:

分布式数据库是由一组数据组成的,这组数据分布在计算机网络的不同计算机上,网络

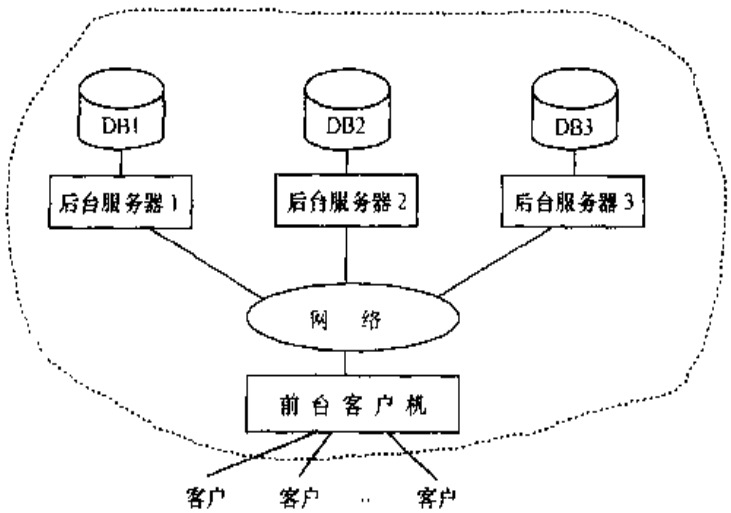


图 14.2 一个多处理机系统(SN 并行结构)

中的每个结点具有独立处理的能力(称为场地自治),可以执行局部应用。同时,每个结点也能通过网络通信子系统执行全局应用。

这个定义比前一个定义更强调了场地**自治性**以及自治场地之间的**协作性**。这就是说,每个场地是独立的数据库系统,它有自己的数据库、自己的用户、自己的 CPU,运行自己的 DBMS,执行局部应用,具有高度的自治性。同时各个场地的数据库系统又相互协作组成一个整体。这种整体性的含义是,对于用户来说,一个分布式数据库系统逻辑上看如同一个集中式数据库系统一样 ,用户可以在任何一个场地执行全局应用。

14.1.2 分布式数据库系统的特点

分布式数据库系统是在集中式数据库系统技术的基础上发展起来的。它具有自己的特性和特征。集中式数据库的许多概念和技术,如数据独立性、数据共享和减少冗余度、并发控制、完整性、安全性和恢复等,在分布式数据库系统中都有了更加丰富的内容。

一、数据独立性

数据独立性是数据库技术追求的主要目标之一。在集中式数据库系统中,数据独立性包括两个方面:数据的逻辑独立性与数据的物理独立性。其含义是用户程序与数据的全局逻辑结构及数据的存储结构无关。

在分布式数据库系统中,数据独立性这一特性更加重要,并具有更多的内容。除了数据的逻辑独立性与物理独立性外,还有数据分布独立性亦称分布透明性(Distribution Transparency)。分布透明性指用户不必关心数据的逻辑分片,不必关心数据物理位置分布的细节,也不必关心重复副本(冗余数据)一致性问题,同时也不必关心局部场地上数据库支持哪种数据模型。分布透明性也可以归入物理独立性的范围。

分布透明性的优点是很明显的。有了分布透明性,用户的应用程序书写起来就如同数

据没有分布一样。当数据从一个场地移到另一个场地时不必改写应用程序。当增加某些数据的重复副本时也不必改写应用程序。数据分布的信息由系统存储在数据字典中。用户对非本地数据的访问请求由系统根据数据字典予以解释、转换和传送。

在集中式数据库系统中,数据独立性是通过系统的三级模式(外模式、模式、内模式)和它们之间的二级映像得到的。在分布式数据库系统中,分布透明性则是由于引入了新的模式和模式间的映像得到的。这将在 14.2.1 节中详细讨论。

二、集中与自治相结合的控制结构

数据库是多个用户共享的资源。在集中式数据库系统中,为了保证数据库的安全性和完整性,对共享数据库的控制是集中的,并由 DBA 负责监督和维护系统的正常运行。

在分布式数据库系统中,数据的共享有两个层次:

(1) 局部共享。即在局部数据库中存储局部场地上各用户的共享数据,这些数据是本场地用户常用的。

(2) 全局共享。即在分布式数据库系统的各个场地也存储供其他场地的用户共享的数据,支持系统的全局应用。

因此,相应的控制机构也具有两个层次:集中和自治。分布式数据库系统常常采用集中和自治相结合的控制机构。各局部的 DBMS 可以独立地管理局部数据库,具有自治的功能。同时,系统又设有集中控制机制,协调各局部 DBMS 的工作,执行全局应用。当然,不同的系统,集中和自治的程度不尽相同。有些系统高度自治,连全局应用事务的协调也由局部 DBMS、局部 DBA 共同承担,而不要集中控制,不设全局 DBA。有些系统则集中控制程度较高,场地自治功能较弱。

三、适当增加数据冗余度

在集中式数据库系统中,尽量减少冗余度是系统目标之一。其原因是,冗余数据不仅浪费存储空间,而且容易造成各数据副本之间的不一致性,为了保证数据的一致性,系统要付出一定的维护代价。减少冗余度的目标是用数据共享来达到的。

而在分布式数据库系统中却希望存储必要的冗余数据,在不同的场地存储同一数据的多个副本,其原因是:

1. 提高系统的可靠性、可用性

当某一场地出现故障时,系统可以对另一场地上的相同副本进行操作,不会因一处故障而造成整个系统的瘫痪。

2. 提高系统性能

系统可以选择用户最近的数据副本进行操作,减少通信代价,改善整个系统的性能。

但是,数据冗余会带来和集中式数据库系统中一样的问题。尤其是冗余副本之间数据不一致的问题是分布式数据库系统必须着力解决的问题。

一般地讲,增加数据冗余度方便了检索,提高了系统的查询速度、可用性和可靠性,但不

利于更新，增加了系统维护的代价。因此应在这些方面作出权衡，进行优化。

四、全局的一致性、可串行性和可恢复性

分布式数据库系统中各局部数据库应满足集中式数据库的一致性、并发事务的可串行性和可恢复性。除此以外还应保证数据库的全局一致性、全局并发事务的可串行性和系统的全局可恢复性。这是因为在分布式数据库系统中全局应用要涉及两个以上结点的数据，全局事务可能由不同场地上的多个操作组成。例如 14.1.1 节的[例 1]中银行转账事务包括两个结点上的更新操作。这样，当其中某一个结点出现故障，操作失败后如何使全局事务回滚呢？如何使另一个结点撤销(UNDO)已执行的操作(若操作已完成或完成一部分)或者不必再执行事务的其他操作(若操作尚未执行)？这些技术要比集中式数据库系统复杂和困难得多，分布式数据库系统必须解决这些问题。

14.2　分布式数据库系统的体系结构

这一节讨论满足 14.1 节中给出的分布式数据库系统特点的系统结构(包括模式结构和分布式数据库管理系统的结构)。同时进一步阐述分布式数据库系统的基本概念。

14.2.1　分布式数据库系统的模式结构

图 14.3 是分布式数据库系统一种模式结构的示意图，实际的系统并非都具有这种结构。在这个结构中各级模式的层次清晰，可以概括和说明分布式数据库系统的概念和结构。

图 14.3 的模式结构从整体上可以分为两大部分：下部是集中式数据库系统的模式结构，代表了各局部场地上局部数据库系统的基本结构；上部是分布式数据库系统增加的模式级别，其中包括：

(1) 全局外模式(Global External Schema)，它们是全局应用的用户视图，是全局概念模式的子集。

(2) 全局概念模式(Global Conceptual Schema)，它定义分布式数据库中数据的整体逻辑结构，使得数据如同没有分布一样。全局概念模式中所用的数据模型应该易于向其他模式映像，通常采用关系模型。这样，全局概念模式包括一组全局关系的定义。

(3) 分片模式(Fragmentation Schema)，每一个全局关系可以分为若干不相交的部分，每一部分称为一个片段(Fragment)。分片模式定义片段以及全局关系到片段的映像。这种映像是一对多的，一个全局关系可对应多个片段，而一个片段只来自一个全局关系。

(4) 分布模式(Allocation Schema)，片段是全局关系的逻辑部分，一个片段在物理上可以分配到网络的不同结点上。分布模式定义片段的存放结点。分布模式的映像类型确定了分布式数据库是冗余的还是非冗余的。若映像是一对多的，即一个片段可分配到多个结点上存放，则是冗余的分布式数据库，若映像是一对一的，则是非冗余的分布式数据库。

根据分布模式提供的信息，一个全局查询可分解为若干子查询，每一子查询要访问的数据属于同一场地的局部数据库。

由分布模式到各局部数据库的映像（映像4）把存储在局部场地的全局关系或全局关系的片段映像为各局部概念模式（Local Conceptual Schema），局部概念模式采用局部场地的DBMS所支持的数据模型。

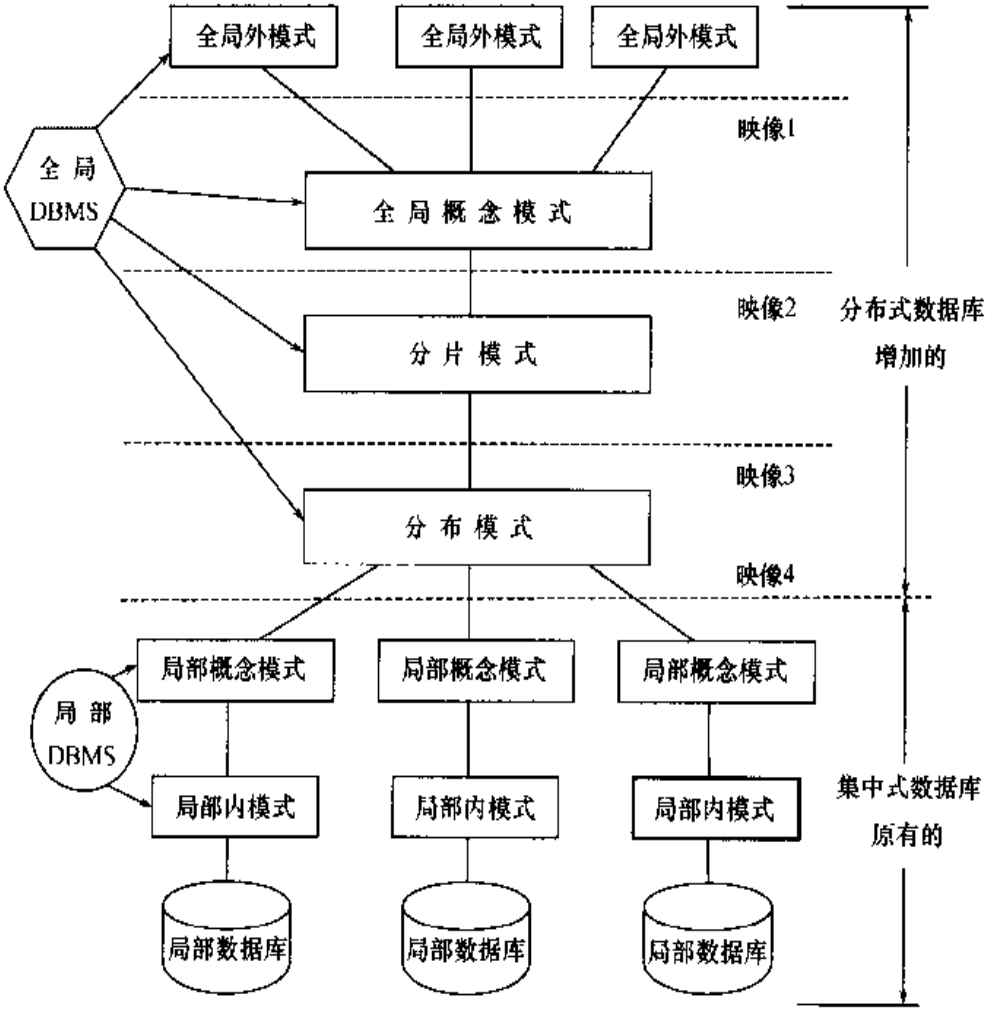


图14.3 分布式数据库系统的模式结构

分片模式和分布模式均是全局的，分布式数据库系统中增加的这些模式和相应的映像使分布式数据库系统具有了分布透明性。

14.2.2 数据分片

将数据分片，使数据存放的单位不是关系而是片段，这既有利于按照用户的需求较好地组织数据的分布，也有利于控制数据的冗余度。

分片的方式有多种，水平分片和垂直分片是两种基本的分片方式，混合分片和导出分片是较复杂的分片方式。

水平分片是指按一定的条件将关系按行(水平方向)分为若干不相交的子集,每个子集为关系的一个片段。

垂直分片是指将关系按列(垂直方向)分为若干子集。垂直分片的诸片段必须能够重构原来的全局关系,即可以用连接的方法恢复原关系,因此垂直分片的诸片段通常都包含关系的码。

导出分片是指导出水平分片,即水平分片的条件不是本身属性的条件而是其他关系的属性的条件。例如学生选课关系 SC(Sno,Cno,Grade),若不是按照学号或课程号或成绩的某类条件分片,而是按照学生年龄 >18 岁和≤18 岁来分片(学生年龄是学生关系 Student 的属性),则这类分片称导出分片,可以用 SQL 语句来表达这两个片段。

年龄 >18 岁的学生选课片段 SC_A(Sno,Cno,Grade)由下面的查询结果组成:

```
SELECT Sno,Cno,Grade
FROM S,SC
WHERE S.Sno = SC.Sno AND S.Sage > 18;
```

年龄≤18 岁的片段 SC_B 由下面的查询结果组成:

```
SELECT Sno,Cno,Grade
FROM S,SC
WHERE S.Sno = SC.Sno AND S.Sage≤18;
```

混合分片是指按上述三种分片方式得到的片段继续按另一种方式分片。例如,先按水平分片得到的某一片段再进行垂直分片(如图 14.4(a)所示),或者按垂直分片得到的某一片段又按水平分片(如图 14.4(b)所示)方式继续分为若干片段。

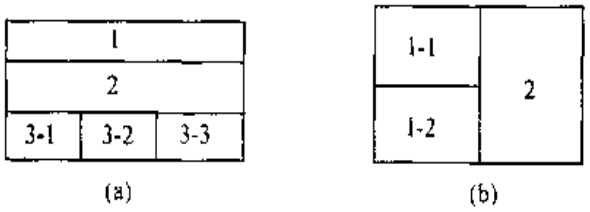


图 14.4　混合分片

无论哪种分片方式都应满足以下条件:

(1) 完全性。一个全局关系中的数据必须完全地划分为若干片段,不允许某些数据属于全局关系但不属于任何一个片段。

(2) 不相交性。不允许一个全局关系的某些数据既属于该全局关系的某一个片段又属于该全局关系的另一个片段(垂直分片中的码属性除外)。

(3) 可重构性。可以由片段重构全局关系,对于垂直分片可以用连接操作重构全局关系。例如,R_1,R_2,R_3 为全局关系 R 的垂直分片,R 的码属性 A,则

$$R = R_1 \underset{R_1.A=R_2.A}{\bowtie} R_2 \underset{R_2.A=R_3.A}{\bowtie} R_3$$

对于水平分片，可以用并操作重构全局关系，例如，SC_A，SC_B 为 SC 的两个水平分片，则

$$SC = SC\_A \cup SC\_B$$

14.2.3 分布透明性

从图 14.3 的模式结构可以看到分布透明性包括：分片透明性、位置透明性和局部数据模型透明性。

分片透明性是分布透明性的最高层次。所谓分片透明性是指用户或应用程序只对全局关系进行操作而不必考虑关系的分片。当分片模式改变了，由于全局模式到分片模式的映像（映像 2），全局模式不变，应用程序不必改写，这就是分片透明性。

位置透明是分布透明的下一层次。所谓位置透明性是指，用户或应用程序不必了解片段的存储场地，当存储场地改变了，由于分片模式到分布模式的映像（映像 3），应用程序不必改变。同时，若片段的重复副本数目改变了，数据的冗余度改变了，用户也不必关心如何保持各副本的一致性，这就是重复副本的透明性。

局部数据模型透明性是指用户或用户程序不必了解局部场地上使用的是哪种数据模型，模型的转换以及数据库语言的转换均由映像 4 完成。

下面讨论一个简单的应用，用它来说明对于不同层次上的透明性，应如何编写应用程序。可以发现，透明性层次越高，应用程序的编写越简单。

[例 1] 设在分布式数据库系统中有全局关系 Student(Sno,Sname,Sdept,Sage)，Student 关系被划分为两个片段 S_A 和 S_B。S_A 代表理学院的学生，S_B 代表文学院的学生。S_A 存储在场地 1(Site1)，S_B 冗余地存储在场地 2 和场地 3 上（如图 14.5 所示）。

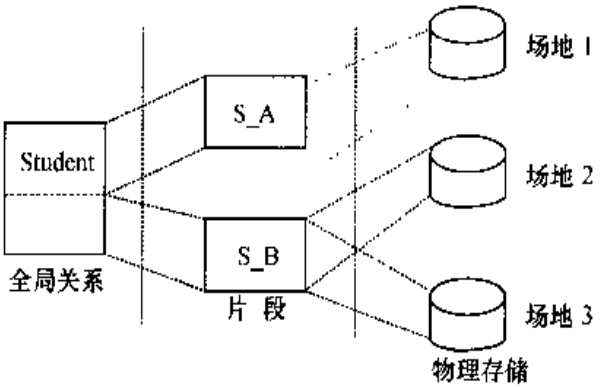


图 14.5 全局关系、片段及物理存储的关系

这里有一个查询，从终端读入一个学号，查找该学号的学生姓名、年龄，并把它们显示在屏幕上。

设应用程序是用嵌入 SQL 语句的 C 语言写的。这里略去了程序细节，仅给出查询部分的算法思想。

情况 1 若系统具有分片透明性，则

```
Scanf("% s",Snumber);                    /* 从终端读入学号到变量 Snumber 中 */
EXEC SQL SELECT Sname,Sage INTO :NAME,:AGE
```

```
                                        /* NAME,AGE 为程序变量 */
FROM Student                            /* 在全局关系 Student 中查找 */
WHERE Sno = :Snumber;
Printf("% s,% d",NAME,AGE);             /* 把 NAME,AGE 打印在屏幕上 */
```

情况 2 若系统具有位置透明性,但不具有分片透明性,则

```
Scanf("% s",Snumber);
EXEC SQL SELECT Sname,Sage INTO :NAME,:AGE
FROM S_A                                /* 先在片段 S_A 中查找 */
WHERE Sno = :Snumber;
If(! FOUND){
EXEC SQL SELECT Sname,Sage INTO :NAME,:AGE
FROM S_B                                /* 再在片段 S_B 中查找 */
WHERE Sno = :Snumber;
}
Printf("% s,% d",NAME,AGE);
```

情况 3 若系统只具有局部数据模型透明性,不具有位置透明性(当然也就不具有分片透明性),则

```
Scanf("% s",Snumber);
EXEC SQL SELECT Sname,Sage INTO :NAME,:AGE
        FROM S_A AT Site1               /* 先在场地 1 的片段 S_A 中查找 */
        WHERE Sno = :Snumber;
If(! FOUND){
EXEC SQL SELECT Sname,Sage INTO :NAME,:AGE
        FROM S_B AT Site2               /* 再在场地 2 的片段 S_B 中查找 */
        WHERE Sno = :Snumber ;          /* 也可以在场地 3 的片段 S_B 中查找 */
}
Printf("% s,% d",NAME,AGE);
```

因为 S_B 重复存放在场地 2 和场地 3,所以用户程序可以对场地 2,或场地 3 访问,用户可选择离他较近的某个场地进行访问,这样可以提高查询效率。

从这里看到,若没有分片透明性,用户必须了解分片的情况。若没有位置透明性,场地的选择由用户程序负责,用户不仅要了解分片的情况,还必须了解片段存放的位置,场地分布的远近,以及通信线路的质量。若系统具有位置透明性,则这些任务均由系统承担,并且当访问场地 2 时,若该线路繁忙或有了通信故障,则系统可以自动地改为访问场地 3,而这种优化工作用户程序是做不到的。

14.2.4 分布式数据库管理系统

分布式数据库管理系统(Distributed Data Management System,D-DBMS)是建立、管理和

维护分布式数据库的一组软件。这里给出一种 D-DBMS 的结构，分析它的主要成分和功能（如图 14.6 所示）。

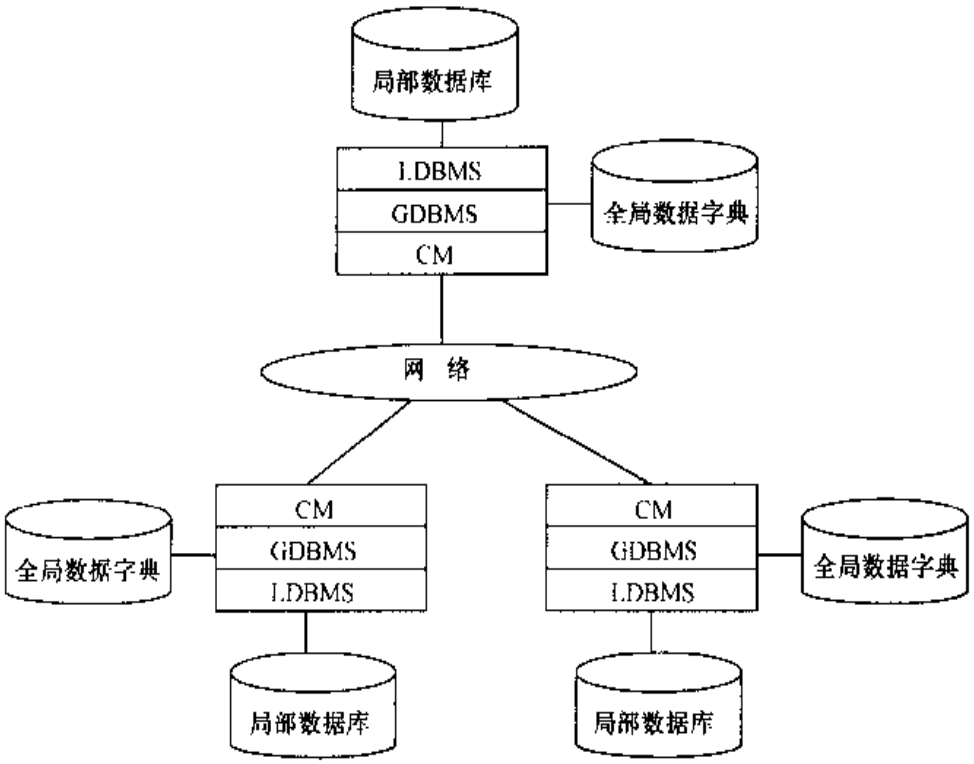


图 14.6 分布式数据库管理系统的结构

D-DBMS 由四部分组成：

（1）LDBMS（Local DBMS），局部场地上的数据库管理系统，其功能是建立和管理局部数据库，提供场地自治能力，执行局部应用及全局查询的子查询。

（2）GDBMS（Global DBMS），全局数据库管理系统，主要功能是提供分布透明性（如图 14.3 所示），协调全局事务的执行，协调各局部 DBMS 以完成全局应用，保证数据库的全局一致性，执行并发控制，实现更新同步，提供全局恢复功能等。

（3）全局数据字典（Global Data Directory，GDD），存放全局概念模式、分片模式、分布模式的定义以及各模式之间映像的定义，存放有关用户存取权限的定义，以保证用户的合法权限和数据库的安全性，存放数据完整性约束条件的定义，其功能与集中式数据库的数据字典类似。

（4）通信管理（Communication Management，CM），通信管理系统在分布式数据库各场地之间传送消息和数据，完成通信功能。

D-DBMS 功能的分割和重复以及不同的配置策略就导致了各种不同的体系结构。

一、按全局控制方式分类

1. 全局控制集中的 D-DBMS

这种结构的特点是全局控制成分 CDBMS 集中在某一结点上，由该结点完成全局事务的

协调和局部数据库转换等一切控制功能。全局数据字典只有一个,也存放在该结点上,它是GDBMS执行控制的主要依据。

这种结构的优点是控制简单,容易实现更新一致性,但由于控制集中在某一特定的结点上,不仅容易形成瓶颈而且系统较脆弱,一旦该结点出故障,整个系统就将瘫痪。

2. 全局控制分散的D-DBMS

这种结构的特点是全局控制成分GDBMS分散在网络的每一个结点上,全局数据字典也在每个结点上存放一份。每个结点都能完成全局事务的协调和局部数据库转换的控制功能,每个结点既是全局事务的参与者又是全局事务的协调者。图14.6就是这种体系结构。一般称这类结构为完全分布的D-DBMS。

它的优点是结点独立,自治性强,单个结点退出或进入系统均不会影响整个系统的运行,但是全局控制的协调机制和一致性的维护都比较复杂。

3. 全局控制部分分散的D-DBMS

这种结构是根据应用的需要将GDBMS和全局数据字典分散在某些结点上,是介于前两种情况的体系结构。

二、按局部DBMS的类型分类

区分不同D-DBMS的一个重要性质是:局部DBMS是同构的还是异构的。

同构和异构的级别可以有三级:硬件、操作系统和局部DBMS。其中最主要的是局部DBMS这一级,因为硬件和操作系统的不同将由通信软件处理和管理。所以,定义同构型D-DBMS为:每个结点的局部数据库具有相同的DBMS,如都是Oracle关系数据库管理系统,即使操作系统和计算机硬件并不相同;异构型D-DBMS为:各结点的局部数据库具有不同的DBMS,如有的是Oracle,有的是DB2,有的是IMS层次数据库管理系统。

异构型D-DBMS的设计和实现比同构型D-DBMS更加复杂。

14.3 查询处理和优化

分布式数据库系统中的查询处理较集中式数据库系统复杂,查询优化较集中数据库系统更重要,效果更显著。下面首先讨论一个实例,说明在分布环境下选择一个好的处理查询策略是多么重要,并了解查询处理和优化涉及的问题,然后讨论查询优化的一般目标和策略。

14.3.1 一个实例

下面是一个很有说服力的例子,说明存取策略优化的重要性。

- 数据库:简化了的供应商和零件数据库

  S(Sno,City)　　10^4 个元组,存放在场地A;

P(Pno,Color)　　10^5 个元组,存放在场地 B;

SP(Sno,Pno)　　16^6 个元组,存放在场地 A;

设每个关系的元组均为 100 字节长。

- 查询:求供应红色零件的、北京的供应商号

```
SELECT S. Sno
FROM S,P,SP
WHERE S. City ='北京' AND
        SP. Pno = P. Pno AND
        P. Color ='红色'
```

- 估算值(某些中间结果的元组数)

  红色零件数 = 10

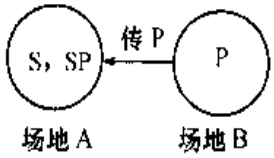
  北京供应商的装运单数 = 10^5

- 对通信系统的假定

  数据传输速度 = 10^4 字节/秒

  传输延迟 = 1 秒

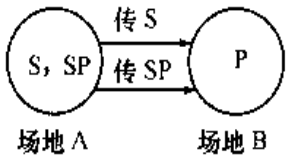
现在,考虑下面 6 种可能的查询存取策略,对各种策略 i,根据下面的公式分别计算通信时间 T[i]:



T[i] = 总传输延迟 + 总数据量/数据传输速度(单位:b/s)

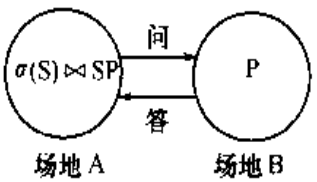
策略 1　把关系 P 传送到场地 A,在 A 地进行查询处理,所以

$$T[1] = 1 + 10^5 \times 100/10^4 = 10^3 \text{ 秒}(16.7 \text{ 分})$$



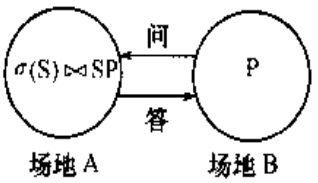
策略 2　把关系 S、SP 传到场地 B,在 B 地执行查询处理,所以

$$T[2] = 2 + (10^4 + 10^6) \times 100/10^4 \approx 10\,100 \text{ 秒}(2.8 \text{ 小时})$$



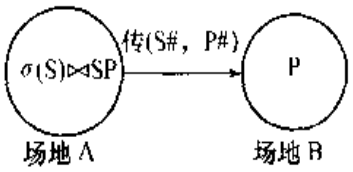
策略 3　在场地 A 连接关系 S 和 SP,选出城市为北京的元组(10^5 个),然后对这些元组中的每个元组的 Pno,询问场地 B,看此零件是否红色。所以共问答 10^5 次,由于不是传送数据,只是消息的问答,所以

$$T[3] = 2 \times 10^5 \text{ s}(2.3 \text{ 天})$$



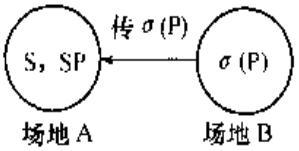
策略 4　在场地 B 选出红色零件的元组(10 个),然后对每一个元组逐一检查场地 A,看北京供应商的装运单中是否有这个零件装运单(若有则选出 S#)每做这样一次检查包括 2 次消息,共问一答 10 次,所以

$$T[4] = 2 \times 10 = 20 \text{ 秒}$$



策略 5　在场地 A 选出北京的供应商的装运单把结果送到场地 B，在场地 B 完成最后处理，所以

$$T[5] = 1 + (10^5 \times 100)/10^4 \approx 1\,000 \text{ 秒}(16.7 \text{ 分})$$



策略 6　在场地 B 的关系 P 中选出红色的元组(10 个)，把结果送到场地 A 完成最终处理。所以

$$T[6] = 1 + (10 \times 100)/10^4 \approx 1 \text{ 秒}$$

表 14.1 综合了上述结果。查询处理时间还应包括在某一场地上的处理时间。这和集中数据库系统中查询处理时间计算是一样的，主要包括 I/O 时间和 CPU 时间。由于在分布式数据库中通信时间是最主要的开销，这里只考虑查询处理中的通信时间。在这个例子中，忽略了发出查询的地点，即最后结果传送到哪个站的问题。在实际系统中还应加上传送结果的开销。

表 14.1　分布环境下查询策略实例比较

策　略	通信时间	方　法
1	16.7 分	把 P 传到场地 A
2	2.8 小时	把 S,SP 传到场地 B
3	2.3 天	对每一北京的装运单，检查相应零件是否红色
4	20 秒	对每一红色零件，检查北京供应商中是否有人供应
5	16.7 分	把北京供应商的装运单传送到场地 B
6	1 秒	把红色零件传送到场地 A

这个例子说明了：

(1) 不同的存取策略通信时间相差很大，达多个数量级！因此必须进行优化。

(2) 在有些策略中数据传输速度和传输延迟都要考虑；而在有些策略中(如例子中策略 3、策略 4)主要考虑传输延迟，还有一些策略中(如例子中策略 1、策略 2、策略 5)数据传输量大，主要考虑传输时间。

此外，有些策略还允许在两个场地上并行处理。例如策略 6，可以在场地 A 完成对北京供应商装运单选择、连接的同时，在场地 B 选出红色零件的元组，因此总的查询时间还可能比集中数据库系统中所花费的要少。

14.3.2　查询处理和优化要解决的问题

绝大多数分布式数据库系统都是关系型的，由于关系查询的语义级别较高，为查询优化提供了可能。系统执行查询可以有多种策略，而且彼此之间性能会有很大差别。分布 DBMS 极其复杂，系统开销非常可观，为了提供能够接受的效率，查询优化是十分必要的。

在分布式数据库系统中有三类查询:局部查询、远程查询和全局查询。局部查询和远程查询都只涉及单个结点上的数据(本地的或远程的),所以查询优化采用的技术就是集中式数据库的查询优化技术(代数优化和非代数优化)。全局查询涉及多个结点的数据,因此查询处理和优化要复杂得多,下面要讨论的是全局查询处理和优化涉及的问题,优化的目标及连接查询的优化方法。

为了执行全局查询和确定一个好的查询处理策略,要做许多判断、计算工作,但总体上可分为三类。

1. 查询分解

对于全局查询必须把它们分为若干子查询,每个查询只涉及某一结点的数据,可以由局部 DBMS 处理。在分布式数据库中一个关系可分为若干逻辑片段,这些片段又可以在系统的多个结点上存放。所以,对一个查询中所涉及的关系需要确定一个物理片段,选择不同的物理片段执行查询操作会直接影响查询执行的效率。因此必须选择查询开销最省的那些物理片段。

2. 选择操作执行的次序

主要是确定连接和并操作的次序,其他的操作顺序是不难确定的,例如选择和投影操作总是应尽量提前执行。这个原则和集中数据库中代数优化的策略相同。但是,涉及不同结点上关系的连接和并操作的次序是必须认真考虑的。

3. 选择执行操作的方法

这包括将若干操作组合在对数据库的一次存取中执行;选择可用的存取路径(如索引)以及选择某一种算法等问题。连接(Join)是查询中最费时的操作,因此连接的执行方法是研究的重点。在分布式数据库中提出了半连接(Semi-Join)的方法。

显然,这 3 个问题不是独立的。为了找到一个全局查询的最优分解,确定执行查询的物理片段,必须了解查询中各操作执行的次序,而这又依赖于对每个操作的执行方法。而且,这 3 个问题之间的联系不是单向的或线性的,而是互相制约的。不过,这 3 个问题中关键是第二个问题。因为第一、第三个问题可将"使应用的局部性最大化"和"使应用的并发性最大化"作为准则,而第二个问题则需要大量计算比较,需要关于数据库的环境信息(如关系的大小,连接属性上的值的分布,通信线路的有关指标等)才能解决。

14.3.3 查询优化的目标

无论是在集中式数据库中还是在分布式数据库中一个查询处理策略的选择都是以执行查询的预期代价为依据的。不同的只是构成一个查询代价的主要因素在这两类系统中不完全一样。

在集中式数据库中,查询执行开销主要是

I/O 代价 + CPU 代价

而在分布式数据库中,除上面两种开销外还有数据在网络上的传输代价:

I/O 代价 + CPU 代价 + 通信代价

对于分布式数据库中数据传输代价应区别不同情况考虑。在远程通信网或数据传输率较低的系统中,通信代价可能会比查询执行中的 I/O 及 CPU 开销大得多,因而作为首要的优化目标来考虑。在局域网且传输率高的系统中,通信代价和局部处理的开销差不多,此时就应在优化中将它们平等对待。

但是,在查询优化过程中将通信代价作为一个首要问题单独列出来研究,从以下两点看来似乎更合理:

首先,通信代价很好估计,通常是数据传输量的一个函数,这个特点是 I/O 开销所不具备的。

其次,分布查询可分为两部分:存取策略的分布优化和局部优化,二者可以分别予以解决。其中,局部优化可以采用集中式数据库中的技术,而分布优化是分布系统要考虑的,一般来说比局部优化更重要。

通信代价可以用下面的公式粗略计算:

$$TC(X) = C_0 + X * C_1$$

其中 X 是数据传输量,这里以 b(位)为单位计算;C_0 为两结点之间初始化一次传输所花费的开销,它由通信系统决定,近似为一个常数,单位为 s(秒);C_1 为单位数据(b)传输的代价,单位为(s/b)。

总之,在分布式数据库中查询优化的首要目标是:使该查询执行时其通信代价最省。

不同结点之间的连接操作和并操作是数据传输的主要原因,因此连接查询的优化在分布优化中是举足轻重的。

14.3.4 连接查询的优化

对连接查询的优化可以采取两种策略,一种是使用半连接来缩减关系(或片段)进而节省传输开销;另一种是不采用半连接,而是用直接连接的优化方案。这里重点讨论半连接的方法。

一、用半连接来优化连接的方法

设 R、S 是两个关系,A、B 分别是 R、S 上的两个属性。

首先引入半连接的记号:$\ltimes$

R、S 在 A、B 上的半连接可表示为:

$$R \underset{A=B}{\ltimes} S = R \underset{A=B}{\bowtie} (\pi_B(S))$$

一个连接运算可以用半连接作为中间步骤来实现:

$$R \underset{A=B}{\bowtie} S = (R \underset{A=B}{\ltimes} S) \underset{A=B}{\bowtie} S$$

具体操作过程和执行代价如下:

设关系 R、S 分别存放在结点 r、s 上(如图 14.7 所示)。

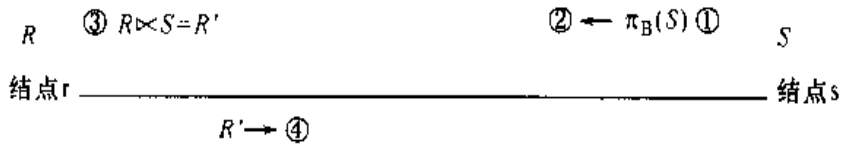


图 14.7

① 在结点 s 作关系 S 在属性 B 上的投影 $\pi_B(S)$

② 把 $\pi_B(S)$ 送到结点 r,代价为

$$C_0 + C_1 \times \text{size}(B) \times \text{val}(B[S])$$

其中,size(B)表示属性 B 的长度,val(B[S])表示关系 S 中属性 B 上不同值的个数。

③ 在结点 r 计算半连接,设结果为 R',即

$$R' = R \underset{A=B}{\ltimes} S$$

④ 把 R' 从结点 r 送到结点 s,代价为:

$$C_0 + C_1 \times \text{size}(R) \times \text{card}(R')$$

card(R')表示 R' 的元组数。

⑤ 在结点 s 执行连接操作:

$$R' \underset{A=B}{\bowtie} S$$

在上面的操作过程中,忽略了局部场地上执行操作的代价,仅考虑了通信代价。

直观地看,使用半连接可以缩减关系 R 的大小使得从结点 r 传送的关系由 R 缩减为 R'(一般 $\text{card}(R) > > \text{card}(R')$),从而减少了通信代价,从中获得好处。

但是,由于执行半连接本身要传输数据($\pi_B(S)$),需要通信代价,因而究竟是否采用半连接还要计算和比较其好处和开销之后再作决定。

采用半连接方案的总代价为

$$C_{SJ} = 2C_0 + C_1(\text{size}(B) \times \text{val}(B[S]) + \text{size}(R) \times \text{card}(R'))$$

不用半连接,直接把 R 送到 S 执行连接的代价为

$$C_{JN} = C_0 + C_1 \times \text{size}(R) \times \text{card}(R)$$

只有当 $C_{SJ} < C_{JN}$ 时,采用半连接方案才是合适的。

上面分析的是简单的连接查询,对于复杂的连接查询,即包括多个关系的连接,则可能存在许多可应用的半连接方案,而其中总有一个方案的效果最佳,因此优化的第一步是从所有半连接方案中找出一个最优的。第二步是与不用半连接的方案进行比较,选出较优者。

由美国计算机公司(Computer Corporation of America)研制的 SDD-1 分布式数据库系统就是采用的半连接的优化方案。

二、用直接连接的方案

上面已经讲过,用半连接方案并不总是有利可图的,因此有些系统如 R*(美国 IBM San

Jose 研究室研制的系统 R 的分布版本)就采用了直接做连接的方案。直接做连接也有不同的操作方法可选用。R* 给出了以下几种可选的方法,它们的算法思想与集中库中的相同:

- 嵌套循环法;
- 排序 - 合并方法。

R* 支持两种传输方式:

- 整体传输:把整个关系传至指定结点再作处理;
- 按需传输:逐个取元组,只传所需要的元组。

以上两种连接算法和传输方式的组合构成了 R* 优化过程中的候选方案,经过代价计算选出其中最省的方案作为最终的查询计划。

14.4 分布事务管理

一个事务的执行必须保持其原子性,即它所包含的所有更新操作要么全部都做,要么都不做。

在分布式数据库系统中,一个全局事务会涉及多个场地上的数据更新。因此,事务是分布执行的。可以把一个事务看成由不同场地上的若干子事务组成。分布事务的原子性就应该是:组成该事务的所有子事务要么一致地全部提交,要么一致地全部回滚。

在多用户系统中,还必须保证分布事务的可串行性。因此,分布事务管理主要包括两个方面:事务的恢复和并发控制。这一节简要地讨论这两个问题。

14.4.1 分布事务的恢复

数据库在运行过程中故障和错误总是难免的。不同的故障会造成数据库不同程度的损害,轻则事务不能正常运行,重则造成数据库数据不一致性。更严重的情况,会使部分或整个数据库遭到破坏。我们已在第七章中讨论了集中式数据库系统中的故障类型和恢复技术,在集中式数据库系统中利用日志文件(必要时加上后备副本)总可以把数据库恢复到某种正确的状态。

分布式数据库系统中,各个场地除了可能发生如同集中式数据库的那些故障外,还会出现通信网络中通信信息丢失、长时间延迟、网络线路中断等事故。因此,情况比集中式数据库更复杂,相应的恢复过程也就更复杂些。

为了执行分布事务,通常在每个场地都有一个局部事务管理器,用来管理局部子事务的执行,保证子事务的完整性。同时这些局部管理器之间还必须相互协调,保证所有场地对它们所处理的子事务采取同样的策略:要么都提交,要么都回滚。为了保证这一策略,最常用的技术是两段提交协议(2-Phase-Commitment Protocol,2PC)。

两段提交协议把一个分布事务的事务管理分为两类:一个是协调者,所有其他的是参与

者。协调者负责作出该事务是提交还是撤销(Abort)的最后决定。所有参与者负责管理相应子事务的执行及在各自局部数据库上执行写操作。

两段提交协议的内容是:

(1) 第一阶段:协调者向所有参与者发出"准备提交"信息。如果某个参与者准备提交,就回答"就绪"(Ready)信息,否则回答"撤销"信息。参与者在回答前,应把有关信息写入自己的日志中。协调者在发出准备提交信息前也要把有关信息写入自己的日志中。

如果在规定时间内协调者收到了所有参与者"就绪"的信息,则将作出提交的决定,否则将作出撤销的决定。

(2) 第二阶段:协调者将有关决定的信息先写入日志,然后把这个决定发送给的所有的参与者。所有参与者收到命令之后首先往日志中写入"收到提交(或撤销)"决定的信息,并向协调者发送"应答(ACK)"消息,最后执行有关决定。协调者收到所有参与者的应答消息后,一个事务的执行到此结束,有关日志信息可以脱机保存。

采用两段提交协议后,当系统发生故障时,各场地利用各自有关的日志信息便可执行恢复操作,恢复操作的执行类似集中式数据库。

可以看出,在两阶段提交协议中,各结点采取的是完全同步的方法来保证数据库的一致性,称为紧密一致性(Tight Consistency)。

紧密一致性能绝对保证任何时刻整个分布式数据库系统的数据一致性和全局事务的原子性,但从实际应用的角度来说,它的缺陷也是非常明显的。

(1) 全局事务可靠性低,任何一个参与结点的失败或网络的中断都将导致整个事务的回滚;

(2) 系统效率低下,由于采用完全同步的方法来保证数据库的一致性,系统的性能将取决于系统中最慢的结点。

为了解决紧密一致性的缺陷,相应地提出了松散一致性(Loose Consistency)的概念。数据各副本的修改是异步的,也就是说各副本将不保证任何时刻数据库绝对的一致性;各副本同步的延迟时间可长可短,视系统的具体情况而定。与紧密一致性相比,松散一致性比较灵活,系统的可用性大为提高,适用于对数据一致性要求不是很高的应用,但对于银行转账这样的关键性应用显然是不适合的。

14.4.2 并发控制

集中式数据库系统中并发控制一般采用封锁技术。

锁可分为不同类型,常用的是共享锁(SLock)、排它锁(XLock)。封锁的对象可以是表一级的或记录一级的。

为了保证并发事务的可串行性,必须做到:

- 遵守锁的相容性规则;

• 遵守两段锁(2-Phase Lock)协议,即如果一个事务已释放了一个锁,那么它就不能再请求新的封锁。

在分布式数据库系统中并发控制也可采用封锁技术,不过与集中式数据库系统相比,由于下面两个原因,使并发控制更为复杂。

(1) 分布式数据库系统支持多副本的特点;

(2) 由于事务的分布执行,封锁的方法会引起全局死锁。

因此分布式数据库系统中基于封锁技术的并发控制必须解决这两个问题。

分布式数据库系统支持多副本的特点给并发控制带来了困难。例如事务 T_1 在场地 1 获得了对数据 d 的 X 封锁,而事务 T_2 在场地 2 对数据 d 的副本申请 X 封锁,场地 2 的事务管理器并不知道场地 1 上的 d 已被 T_1 加了 X 封锁,因此事务 T_2 仍获得对 d 的 X 封锁,这样,封锁机制就失去了原来的功能而完全无用了。

T_1在场地 1	T_2在场地 2
$Xlock(d_{site1})$	$Xlock(d_{site2})$

为了避免这个问题,分布事务管理就要把"事务 T_1 对 d 的 X 封锁"这件事让 d 副本所在场地上的事务管理器都知道,一个简单的方法是向这些场地的事务管理器发出局部封锁请求,这个办法是有效的。但封锁的冗余度很大,局部封锁的数目和副本数相同。为了减少系统开销,处理多副本的封锁可采取如下几种方法:

(1) 对写操作,要申请对所有副本的 X 锁。对于读操作,只要申请对某个副本的 S 锁。

(2) 无论是写操作还是读操作都要对多数(大于半数)副本申请 X 锁或 S 锁。

(3) 规定某个场地上的副本为主副本,所有的读写操作均申请对主副本的封锁。

这 3 个方法均可有效地发现冲突,协调并发事务的执行。

基于封锁的并发控制方法在分布环境下还必须解决全局死锁的问题,所谓全局死锁即包括两个以上场地上的死锁。

和集中式数据库系统相似,通常采用分布等待图的方法来检测死锁。不同的是,分布情况下死锁检测涉及多个场地,多个局部数据库,需要较多的通信和验证,开销较大。

除了用死锁检测及解除方式来解决死锁问题外,还可以采用死锁预防方法,即不让死锁发生。典型的一种解决方法是对事务按某一标准进行排序,只允许它们沿着这一次序单向等待,这就不会发生死锁了。

在分布式数据库中还研究了基于时标(Time Stamp,亦称为时间戳)的方法和乐观方法等并发控制技术,但实际系统则大都是基于封锁的方法。

14.5 小 结

由于应用的驱动,数据库技术和网络技术的进展,分布式数据库技术得到了长足的

发展。

这一章讨论了分布式数据库系统的基本概念、这一领域内的某些主要问题以及解决这些问题所采用的基本技术和基本方法。

分布式数据库系统以集中式数据库系统技术为基础，但不是简单地把集中式数据库连网就能实现的。分布式数据库系统具有自己的性质和特征。集中式数据库的许多概念和技术，如数据独立性、数据共享和减少冗余度、并发控制、完整性、安全性和恢复等在分布式数据库系统中都有了不同的更加丰富的内容。

习　题

1. 什么样的数据库系统是分布式数据库系统？图 14.1 的系统配置在什么情况下只能算分散的数据库系统？在什么条件下才是分布式数据库系统？

2. 分布式数据库系统有什么特点？

3. 试述研制分布式数据库系统的目的和动机。

4. 试述分布式数据库系统的模式结构。

5. 什么是数据分片？有几种分片方式？数据分片的目的是什么？有什么优点？

6. 试述分布透明性的内容。

7. 什么是同构型 D-DBMS？什么是异构型 D-DBMS？

8. 设在 14.2.3 节的分布式数据库系统例子中，还有全局关系 SC(SNO,CNO,G)，它具有两个导出分片 SC_A,SC_B，分别存储理学院和文学院学生的选课记录。SC_A 存放在场地 4，SC_B 存放在场地 5。今有一个稍复杂的查询，从终端输入一个课程号，查找选修该课程的学生学号和姓名，并把它们显示在屏幕上。请写出具有不同层次分布透明性(类比例子中的三种情况)的应用程序。不必给出细节，只需写出算法思想。

9. 对 14.3.1 节的例子中介绍的六种策略改用下列估算值后分别计算通信时间：

红色零件数 = 1 000，

北京供应商的装运单 = 10 000。

10. 设关系 R,S 存储在不同的场地，已知：

$\mathrm{card}(R) = 100, \mathrm{size}(R) = 50$

$\mathrm{card}(S) = 50, \mathrm{size}(S) = 5$

今要计算 $R \underset{A=B}{\bowtie} S$，其中 A,B 分别为 R,S 上的属性，并且

$\mathrm{val}(A[R]) = 50, \mathrm{size}(A) = 3$

$\mathrm{val}(B[S]) = 50, \mathrm{size}(B) = 3$

若 $R' = R \underset{A=B}{\ltimes} S, \mathrm{card}(R') = 0.2 * \mathrm{card}(R)$

$S' = S \ltimes R, \mathrm{card}(S') = 0.8 * \mathrm{card}(S)$

请比较采用以下不同策略时的通信代价：

(a) 在 R 所在场地执行连接操作，利用半连接方法；

(b) 在 S 所在场地执行连接操作，利用半连接方法；

(c) 在 R 所在场地执行连接操作，不用半连接方法；

(d) 在 S 所在场地执行连接操作,不用半连接方法;

哪种情况代价最省(不考虑把结果传送到某一场地的代价)?

11. 试述下列概念:两段提交协议(2PC);分布事务的原子性;全局死锁。

12. 在分布式数据库系统中,对多副本的封锁有几种解决方法?

本章参考文献

1 Rothnie J B, Goodman N, A Survey of Research and Development in Distributed Database Management. in Proceedings of LDB, 1977

(本文综述了分布式数据库系统早期的研究结果。)

2 Ceri S et al. Distributed Databases: Principles and Systems. McGraw Hill, NY, 1984

(本书介绍了分布式数据库系统的基本概念、原理和系统。)

3 周龙骧等. 分布式数据库管理系统实现技术. 数据库技术丛书之一, 北京:科学出版社, 1998

4 郑振楣, 于戈, 郭敏. 分布式数据库. 数据库技术丛书之一, 北京:科学出版社, 1998

5 Tamer Ozsu M, Valduriez P. Principles of Distributed Database Systems. Prentice-Hall International Inc. 1991

6 石树刚, 郑振楣, 阮春等. 分布式数据库系统的实用化. 微型计算机, 1989

(本期为分布式数据库的专刊,刊登了十余篇论文(如文献[6],[7],[8],[9],[10],[11]),反映了国内在这一领域中当时的研究的情况和部分成果。)

7 李曦, 王珊, 萨师煊. 一个基于大型机远程通信的分布式数据库查询系统 DQS/SEIS. 微型计算机, 1989

8 杜小勇, 王珊, 萨师煊. 分布式数据库 DQS/SEIS 设计实现中若干问题的探讨. 微型计算机, 1989

9 张霞, 郑怀远. 面向客体和语义关联的异构分布式数据库系统 OHDDB-DGI. 微型计算机, 1989

10 王能斌, 徐立臻, 程琪. SUNDDB:一个分布式数据库管理系统. 微型计算机, 1989

11 周晓方, 金志权, 柳诚飞. LSZTM:一个分布式数据库系统的管理. 微型计算机, 1989

12 王珊等. The Global Conceptual Model Design in DQS/SEIS. 第二届泛太平洋计算机国际会议 PPCC-3 论文集. 1989

(本文讨论分布式数据库系统全局概念模式的设计,以中国人民大学为国家经济信息系统研制的分布查询系统 DQS/SEIS 为实际背景。)

13 周龙骧, 顾君忠. POREL:一个分布式关系型数据库系统. 计算机科学, 1982

14 Ceri S, Navathe B, Wiederhold G. Distribution Design of Logical Database Schemas. IEEE Transactions on Software Engineering, Vol. SE-9, No. 4, 1983

(本文讨论了分布式数据库系统的设计问题。)

15 Epstein R, Stonebraker M R. Analysis of Distributed Database Processing Strategies. in Proceedings of VLDB, 1980

(本文讨论了分布式数据库中的查询处理策略。)

16 Apers P M G. Yao S B. Optimization Algorithms for Distributed Queries, IEEE Transactions on Software Engineering, Volume SE-9, Number 1, 1983

(本文讨论了分布查询处理的优化算法。)

17 Daniels D, Selinger P G, et al. An Introduction to Distributed Query Compilation in R*, in Distributed Data

Bases. Proceedings of International Symposium on Distributed Databases, 1982

（本文全面介绍了分布式数据库系统 R*，介绍 R* 的论文很多，这只是其中之一。）

18 Berntstein P A, Chiu D W. Using Semijoins to Solve Relational Queries. Journal of the ACM, Volume 28, Number 1, 1981

（本文讨论了分布查询处理的半连接优化方法。）

19 Kambayashi Y, Yoshikawa M, Yajima S. Query Processing for Distributed Databases Using Generalized Semi-joins. Proceedings of the ACM SIGMOD, 1982

（本文讨论了分布查询处理的半连接优化方法。）

20 Traiger I L, et al. Transactions and Consistency in Distributed Database Management Systems. ACM TODS, Volume 7, Number 3, 1982

（本文讨论了分布式数据库系统中事务及其一致性的概念。）

21 Bernstein P A, et al. Concurrency Control in a System for Distributed Databases (SDD - 1). ACM TODS, Volume 5, Number 1, 1980

（本文讨论了分布式数据库的并发控制问题。）

22 Bernstein P A, Goodman N. Timestamp-based Algorithms for Concurrency Control in Distributed Database Systems. in Proceedings of VLDB, 1980

（本文讨论了分布式数据库系统中基于时间戳的并发控制方法。）

23 Kohler W H. A Survey of Techniques for Synchronization and Recovery in Decentralized Computer Systems. ACM Computing Surveys, Volume 13, Number 2, 1981

（本文讨论了分布式数据库系统中的同步和数据库恢复问题。）

24 Lampson B, Sturgis H. Crash Recovery in a Distributed Data Storage System. Technical Report, Computer Science Laboratory, Xerox, Palo Alto Research Center, Palo Alto, CA, 1976

（本文提出了分布式数据库系统中事务的两段式提交协议。）