利用 IBM Informix 分片技术有效管理海量数据

Using IBM Informix Fragment statement to manage huge data

苏 俊 （中国人民大学信息学院 100872）

摘要：本文通过介绍 IBM Informix 分片技术提出了一种有效管理海量数据的实用方法。在简要说明了 IBM Informix 磁盘空间管理机制之后，阐述了在实用系统中不仅要保证数据操作的正确性，更要关注数据管理的实用性和高效性。

关键词：分片 IBM Informix 页 DML 语句

1 引言

目前，IBM Informix 数据库广泛应用于电信、金融、保险、自动控制等应用领域。数据库系统的应用水平不仅体现在数据库设计、开发程序等方面，更重要的是体现在数据管理水平上。所以，在实际应用中，随着数据类型的多种多样、数据量的不断增大，数据库系统的运行性能越来越成为瓶颈。影响数据库系统性能的因素很多，管理数据存储不合理是主要因素之一。

本文介绍一种充分利用分片技术来有效管理数据存储的方法。

2 IBM Informix 存储技术概述

这里简要说明一下 IBM Informix 是如何管理数据存储空间的。IBM Informix 首先从数据库外部（操作系统或原始设备）获得磁盘空间。管理外部空间的单位称为 Chunk，它是 IBM Informix 获得的一段连续磁盘空间。Chunk 可能是生设备（基于字符的设备）、生设备的一块、一个熟设备（基于块的设备）、熟设备的一块、或者一个 UNIX 文件（熟文件）。

IBM Informix 一旦获得可以管理和使用物理空间，就纳入到其内部管理机制中。Informix 存储管理的内部机制采用四层结构：数据空间（dbspace）、表空间（tablespace）、区间（extent）和页（page）。为了表述方便，本文以下使用 dbspace、tablespace、extent 和 page 表示数据空间、表空间、区间和页。

Dbspace 是 chunk 的集合，每个 dbspace 必须至少分配一个 chunk，其第一个 chunk 称之为初始 chunk，dbspace 根据需要可以分配尽可能多的 chunk。如果一个 dbspace 所管理空间大部分被占用之后，可以追加新 chunk 来增加该 dbspace 的空间。一旦外部 chunk 空间加入到 dbspace，都要被初始化为页（page）。page 是 IBM Informix 进行输入/输出的基本单位。所有数据都存储在页中。页的尺寸目前有两种：2K 和 4K 字节。页尺寸随数据库系统版本的不同而不同，并且是不可更改的。

表空间（Tablespace）是一张表所占用存储空间的集合。当新建一张表或者一张表的存储空间不够使用时，tablespace 都会向 dbspace 提出申请并获得空间，其获得空间的形式是 extent。所谓 extent 是一段物理上连续存储的 page 集合。当新建一张表时，dbspace 第一次分给该表的区间称为初始区间，一般缺省值为 8 页。随着新建表中数据量的不断增加，dbspace 随后分配给该表的空间称为追加区间，其取值在 Create Table 语句中说明。

IBM Informix 的页结构如图 1 所示。从图中可以看出，IBM Informix 页分为三个部分：

（1）页头：包含有 24 字节信息，主要用来跟踪记录该页的数据情况。

（2）数据存储区：存储数据或索引信息。

（3）页尾：这里又分为：

① 槽表区：对插入到本页的每一条记录都有 4 字节的结构，其中包括数据行在本页的起始位置和行长度，槽表主要是用于在一页中定位数据行的存储位置。

② 页尾时间戳：每页的最后 4 字节，用于保证写

的有效性。在页头同样有一个时间戳,如果两个时间戳不一致,说明该页是"脏"页,也就是说,该页的内存映像与对应的磁盘存储映像是不同的,这需要磁盘 I/O 操作进行同步。

从数据库系统内部来看,每当新建一张表时,必须指明所建新表的存储空间,即新表存储在哪个 dbspace 中以及分配空间的策略。IBM Informix 首先检查 Create Table 语句的语法和语义,如果没有问题,则要从指定 dbspace 中获取初始区间(缺省值为 8 页)。例如,新建表之后立即插入一批数据,如果该初始区间中的空间被全部占用,则要根据建立表时所定义的追加区间大小向 dbspace 申请空间,申请到的空间归 Tablespace 管理。所以,当对一张表分别在不同时间插入大量数据时,会出现表空间的分布,甚至同一张表中数据分布在不同磁盘上。

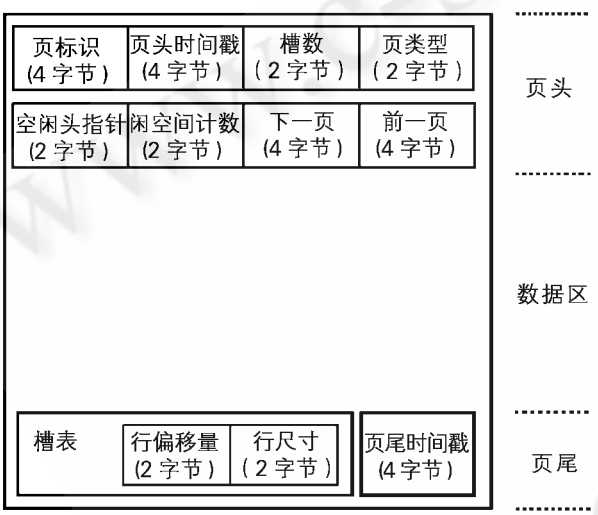

图 1　页结构

当对一张表进行 DML(插入、删除和修改)操作时,会影响到页的存储布局。

当对表插入(insert)一行数据时,Informix 首先从 tablespace 中找到正在使用的最后一页,调入内存,并在该页槽表中分配一个槽号,在页中数据存储区申请空间存储数据,修正页头中相关控制信息,重置页尾时间戳。一旦出现槽号和数据空间不足的情况,则会申请新的一页或者新的区间等,引起瀑布式地申请和分配空间。

当对表删除(delete)一行数据时,首先定位到要删除行所在的页,调入内存,释放对应槽号以及该行所占用的数据存储空间,修改页头中相关控制信息,重置页尾时间戳。删除操作会影响到页的充满度,也影响到数据库系统的 I/O 效率(调入内存的页中包含的有用信息较少)。一行数据在表中的标识 ROWID 由页标识和槽号所组成。所以已删除数据行的槽号不能用于以后插入的数据行。请参见图 2,其中阴影部分为删除内容。

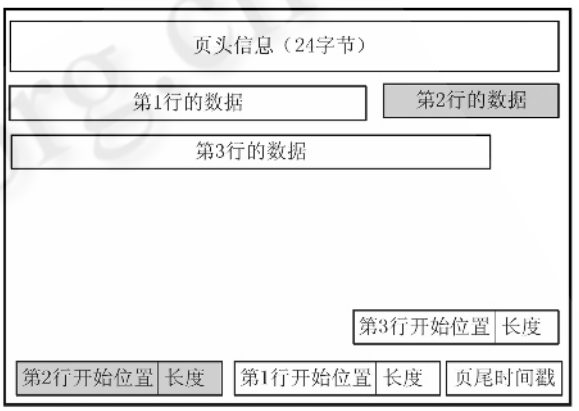

图 2　DML 操作对页映像的影响

当对表修改(update)一行数据时,首先找到要修改数据行所在页,调入内存,根据槽号定位到要修改数据的存储位置,修改结果有两种情况:一种情况是新数据比原来数据占用更多空间,如果溢出许多数据,在极端情况下,会使用一个新页(链接页)来存储溢出数据;另一种情况是新数据比原来数据占用空间少。无论哪种情况,都会使得页内数据存储区域出现碎片或者数据隔离度增加(一行数据存储在多个页中)。

根据每个 page 存储内容的不同,Informix 分为不同类型的页,例如数据页、索引页等。

由此我们得出:当通过 SQL 语句对数据库进行操作时,内部存储映像也在发生着变化,这种变化可能会严重影响到数据库系统的性能。这也深深地提醒我们在实用系统中不仅要保证操作正确,还要保证存储空间的合理性。

3　海量数据的 DML 操作

在许多实用系统中,常常需要管理海量数据。例如,企业的订单/销售管理、电信行业的计费、银行的储

蓄帐户管理等应用中数据都具有量大、变化大的特点，由此引出对于海量数据如何进行 DML 操作的问题。

这里以学校管理学生选课信息来说明这个问题。

假设在校学生选课情况表的定义如下：

```
Create Table OnLine_SC (
    sno char(10),//学号
    cno char(10),//课程号
    grade int,//年级
    finished_Date//上课结束时间
    marknumeric(6,2),//成绩
    ......
);
```

因为考虑到在校的、不同年级的学生都可以选择四年八个学期的课程，所以，以上 Create Table 语句在选课表中主要说明了学号、课程号、年级、上课时间以及成绩共五个主要列。

当有一届学生（这里假设为 2000 级）毕业离校时，应当对毕业生的相关信息进行处理。例如，把该届毕业生的学习记录从在校学生表 OnLine_SC 中删除，并插入到具有相同结构的历史归档数据表 OffLine_SC 中。一般情况下，使用下列 SQL 语句：

```
insert into OffLine_SC
    select *
    from OnLine_SC
    where grade = '2000';
delete from students where grade = = '2000';
```

由于上面每个语句不仅会产生影响整个数据库服务器性能的长事务，而且会涉及到大量数据库内部存储的变化，这样的操作往往会引起性能问题。在其他应用领域中，类似操作情形常常遇到。例如，在电信计费中，每个手机通话明细数据归档时，可能会花费很长时间。显然这样的操作在数据量较少的情况下，还可以使用。但是多数实用情况下，这种做法会严重甚至阻塞数据库系统的运行。

根据作者的使用经验，在有大量数据需要进行归档的实际应用中，应该有效采用 IBM Informix 的分片技术。

4　分片技术

IBM Informix 提供了本地分片技术来平衡 I/O 操作，以及提供最大的并发度。所谓分片技术是把一张表中数据分布存储到多个数据空间（dbspace）中，如图 3 所示。

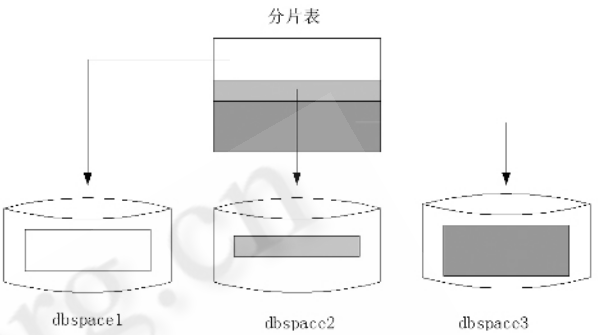

图 3　分片示意图

分片技术有如下优点：

（1）提供并发操作。

（2）平衡 I/O。

（3）提供了适用于备份和恢复的颗粒程度。

（4）使得系统具有较高的可用性。

IBM Informix 分片类型有轮寻（Round robin）和表达式（Expression）两种。

在轮寻方式中，依次把所插入数据行轮换放置在不同分片中。

例 1：根据插入顺序，依次把数据行插入到 OnLine_SC 的三个数据空间中，则使用下列 SQL 语句：

```
Create Table OnLine_SC (
    ....//具体列定义
) fragment by round robin in db1,db2,db3
extent size 10000 next size 5000;
```

在表达式方式中，关键在于表达式的设计。

例 2：使用范围表达式，下列 SQL 语句根据 grade 列值来把插入数据行放置到不同数据空间中：

```
Create Table OnLine_SC (
    ...
    grade int,//年级
    ...
) fragment by expression
    grade in (2000,2004) in db1,
    grade > 2000 and grade < 2003 in db2,
    remainder in db3;
```

则 2000 级、2004 级的数据存储在 db1 中,2001 至 2002 级的数据存储在 db2 中,2003 级以及其他年级的数据存储在 db3 中。

例 3:使用 Hash 函数作为表达式,如下:

```
Create Table OnLine_SC (
  ...
  grade int,//年级
  ...
) fragment by expression
  MOD(grade,3) =0 in db1,
  MOD(grade,3) =1 in db2,
  MOD(grade,3) =2 in db3;
```

如果在校学生有 2000 级至 2003 级的话,那么 2000 级的数据存储在 db3 中,2001 级和 2004 级的数据存储在 db1 中,2002 级的数据存储在 db2 中。

在实际应用中,要根据实际情况恰当选择不同分片策略。因为分片技术是一个双刃剑,对于某些操作效率非常明显,而对于某些操作却适得其反,所以要科学地使用分片技术。

5 分片技术的应用

在大型联机事务处理(OLTP)中,常常需要把大量的联机数据进行归档。如果简单地采用前面所介绍的 INSERT/DELETE 组合 SQL 语句,则会涉及到大量数据的变动,占用大量事务日志空间,出现极大影响数据库服务器性能的长事务。

为了既能够快速归档大量数据,又不影响系统性能,这里提出一种基于分片技术的归档方法。IBM Informix 提供了 ALTER FRAGMENT…DETACH 语句,其中 DETACH 子句的作用是把表的一个分片独立为一张表。

在 OnLine_SC 学生选课表中,根据第 2 节的介绍,所有数据存储在一张表中。当某一届学生毕业时,采用 DELETE 语句把毕业学生的记录从这张表中删除之后,大家可以想象出该张表的存储映像如何混乱,每个数据页中有效数据量较少,也就降低了 I/O 的有效性,整体上降低了数据库系统的性能。

为了解决这个问题,首先要从存储上就把不同年级的学生学习记录分别存储在不同分片中,这样当某一届的学生毕业时,只需要把该分片独立为一个单独的基表。无论是操作过程,还是操作之后的存储映像,都会有良好的性能,不会严重地影响到数据库系统的整体性能。

具体操作如下:

首先建立一个在校学生的分片表的结构,使用下列语句:

```
Create Table OnLine_SC (
  ...
  grade int,
  ...
) fragment expression by
  grade = '2000' in db_2000,
  grade = '2001' in db_2001,
  grade = '2002' in db_2002,
  grade = '2003' in db_2003;
```

当新一届学生(假设为 2004 届)入学时,使用下列语句为新生增加一个新的分片:

```
ALTER FRAGMENT ON TABLE OnLine_SC
  ADD grade = "2004" IN db_2004
  BEFORE db_2001;
```

当有一届学生(假设为 2000 届)毕业时,使用下列语句把 2000 级学生信息从在分片中独立为归档表 students_2000:

```
ALTER FRAGMENT ON TABLE OnLine_SC
  DETACH db_2000 students_2004;
```

以上只是说明了学生的管理问题,这种方法可以推广到电信计费、财务明细帐、企业订单管理等有大量的、具有时间段特征的在线信息,同时又要定时归档处理的应用领域。

参考文献

1 Managing and Optimizing Informix Dynamic Server Databases, Part Number 000 – 5527, Book Number 502 – 57561 – 999999 – 1。

2 System Administration: Informix Dynamic Server, Part Number 502 – 53881 – 999999 – 9。

3 Informix Dynamic Server Performance Tuning, Version 5 05 – 2001000 – 8446。