

TPC-C测试标准

面向对象

数据库(14)

设计

计算机科学1998 Vol. 25 No. 3

57-60

TPC-C 测试标准及其在面向对象数据库上的设计与实现^{*}

TPC-C Benchmark and Implementation on Object-Oriented Database Systems

于戈 王欣晖 王[✓]国仁 单吉第
(东北大学计算机科学与工程系 沈阳110006)

TP311.13

摘要 This paper introduces the TPC-C benchmark for online transaction processing with its database model, transaction model and testing qutoas, and presents the design and implementation method on object-oriented database systems.

关键词 Benchmark, Database, Object-oriented design

所谓测试标准(Benchmark)是一种用来测试计算机软件或硬件某一方面性能的规范。目前已经有许多不同种类的测试标准,其中 TPC-C 则是专门用于测试数据库系统在线事务处理系统性能的测试标准^[1]。数据库测试标准主要用途有:数据库系统的开发者通过对其系统的性能测试和评价,而对系统进行改进和完善;用户根据不同系统的测试数据,而选择性能优良的系统。因此,数据库 Benchmark 是更好地开发和应用数据库系统的一个重要工具。

在线事务是最常见的应用,如银行账务,商业销售,民航售票等。在线事务特点是数据量大,响应时间有严格要求。在线事务处理(OLTP)系统由硬件和软件两部分组成,硬件指的是运行该系统的计算机,软件包括操作系统、数据库系统及通讯软件等,其中数据库系统是 OLTP 系统的核心,是影响系统性能的关键。一些世界知名的数据库厂商如 Oracle, Informix 和 Sybase 等,都基于不同的硬件平台对他们的产品进行 TPC-C 测试,以测试结果数据作为评价依据和广告宣传依据。

对一个数据库系统的评价,可以通过建立数学模型,进行理论分析和仿真。但是,由于影响计算机系统性能的因素很多,应用环境很复杂,很难用准确的公式表示出来。因此,进行实际测试,提供直观的数据和图表能给出更客观的结果。TPC-C 测试标准的采用,使人们可以通过运行一个权威的、标准的测试,对不同的数据库系统进行客观的评价。

OO 数据库系统作为新一代的数据库技术,主要用于具有复杂数据结构的场合,如工程设计,多媒体信息处理等。许多 OO 数据库厂家如 ObjectStore, O2, Ontos 等已经采用了 OO 数据库系统的性能测试标准如 OO1, OO7^[2]对他们系统进行了测试。这些测试标准是模拟工程应用,如复杂结构、超长数据等,与在线事务处理应用完全不同。随着数据库技术的发展,如对象关系数据库技术,OO 库也将广泛应用于在线事务处理应用中。在 TPC-C 测试标准中只给出了数据模型、事务模型和测试指标的定义,并不给出具体的实现方法,因为在不同的系统上的实现方法是不同的。在传统数据库系统如关系数据库的 TPC-C 实现方法早已被各厂家所完成,但是,就我们所知,在 OO 数据库上的实现还没有人做过。而我国的数据库测试技术方面开展的工作也不很多。因此,有必要研究和开发数据库测试技术和 TPC-C 在 OO 数据库系统上的实现方法。

一、TPC-C 概述

为了对在线事务处理系统进行合理的测试,TPC-C 测试标准的制定主要考虑了这样几个因素:1)模拟各种复杂情况下的多类型事务,包括:在线式、延时式的事务,事务的性质(ACID),多个在线终端并发执行。2)要求适当的系统和应用执行时间,频繁的磁盘 I/O 操作。3)多种多样的数据访问方式。数据库含有不同长度、不同属性、不同关系,通过主关键字和候选关键字存取,完成数据访问和数据更新

^{*} 本文受到辽宁省自然科学基金的资助。

的并发操作。

TPC-C 所模拟的应用是实际商业销售系统中的批发业务处理,其参考模型如图1所示,这是一种有代表性的在线事务处理。一个销售公司下设若干个仓库(Warehouse),每个仓库负责为10个分店(District)供货,而每一个分店拥有3000名不同的顾客(Customer)。

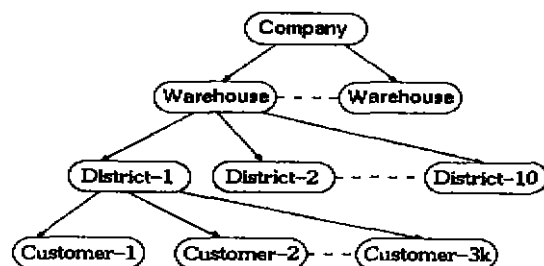


图1 TPC-C应用参考模型

TPC-C 定义了5种典型的事务,它们分别代表顾客订购某些货物、顾客付款、查询顾客订购情况、向顾客发货、查询库存情况等5种应用。这些事务都必须满足事务的基本性质,即原子性,一致性,隔离性和持久性。

TPC-C 测试的结果是一些数值和图表,其中最常用的指标是 tmpC 和 tmpPC/Cost。对一个被测系统进行 TPC-C 测试的过程为:

首先,按 TPC-C 规范的要求,为被测系统编写 TPC-C 测试程序、创建数据库和加载初始数据。然后,同时运行10个测试程序,每个测试程序按照规定的等待时间和事务类型组合运行。每个测试程序的循环运行模式如图2的示。

当程序运行一定长的时间后(最少8小时),统计出该期间内提交的特定事务(New Order 事务)的总数,计算出特定事务吞吐率(每分钟提交事务的个数),称为 tmpC,单位称为 MQTh(Maximum Qualified Throughput)。tmpC 的值越大,说明被测系统的在线事务处理能力越强。

另一个重要指标是性能/价格比(Cost/tmpC)。依据 TPC-C 规范,被测系统的价格(COST)包括做测试所用的硬件(主机,外设,网络设备,电缆等),软件(操作系统,数据库系统,附加软件等),tmpC 的值除以价格就是被测系统的性能价格比,性能价格比越大,表明单位价格下该系统所能提供的在线处理能力越强。

当测试完成后,按照 TPC-C 要求,需要提出一

份全面报告,以书面的形式报告测试的过程,实际测试与 TPC-C 规范的吻合程序,测试结果等。

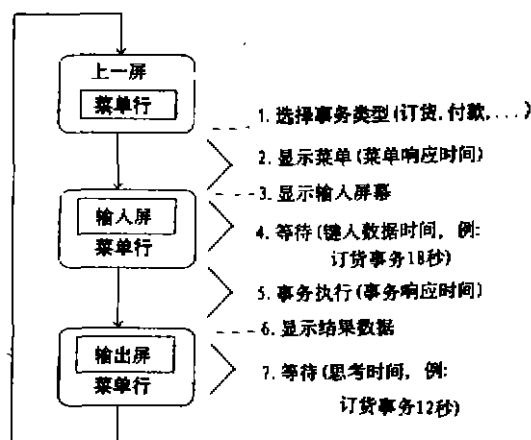


图2 TPC-C的测试程序运行模式

二、数据库定义

TPC-C 数据库由9张二维表构成,用这9张表模拟了实际商业在线事务处理应用的数据库。TPC 委员会在公布 TPC-C 规范时是用关系模型的术语来描述规范全文的,但这并不意味着 TPC-C 只能用于基于关系的在线事务处理系统的测试,事实上任何数据库系统,甚至文件系统都能做 TPC-C 测试。在做 TPC-C 测试时,先初始化这些表,即按 TPC-C 规范向表内填入原始数据,然后执行顾客订购,顾客付款,商店查询订购情况,商店查询库存情况,商店付货等五种事务,其中有些是只读型事务,有些要执行对数据库的修改操作。在这些事务运行期间,表内的数据动态地改变。图3给出了完整的 TPC-C 测试数据库模式。

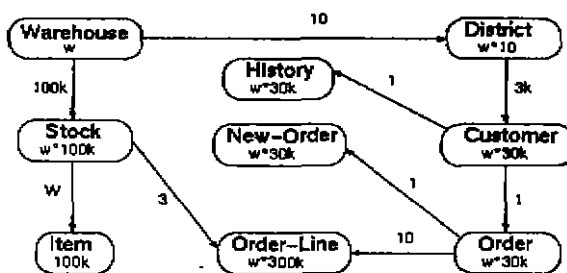


图3 TPC-C数据库模式

1) 仓库(Warehouse)表,共有9个字段,如下所示,例如 w id 是一个整型字段,表示仓库的 ID,为该表的关键字。w name 是一个可变长字符型字段,表示仓库的名字。

属性名	w id	w name	w street1	w street2	w city	w state	w zip	w tax	w ytd
类型	int2	vchar[10]	vchar[20]	vchar[20]	char[20]	char[2]	char[9]	int4	int12

2)分店(District)表,共有11个字段,包括分店的ID,所属的仓库的ID,它们共同构成了该表的关键字,且仓库ID为外关键字。

3)顾客(Customer)表,共有21个字段,包括顾客的ID,所在商店的ID,所在仓库的ID,这3个字段共同构成了该表的关键字,且后两个字段为外关键字。

4)历史(History)表,由8个字段组成,包括历史所属的顾客的ID,该顾客所在分店的ID,该顾客所在仓库的ID,该表作为顾客订货的日志记录,无关键字,这3个ID字段均为外关键字。

5)订单(Order)表,共有8个字段,包括订单的ID,订货的商店的ID,订货的仓库的ID,它们共同构成了该表的关键字,均为外关键字。

6)订单明细(OrderLine)表,由10个字段组成,包括订单明细的序号,订单明细所属的订单的ID,订单所在的仓库的ID,订单所在的商店的ID,这4个字段共同构成了该表的关键字,并且,后3个字段为外关键字。

7)新订单(NewOrder)表,包括3个不同长度和类型的字段,有新订货的仓库的ID,新订货的商店的ID,新订单所属的订单的ID,它们共同构成了该表的关键字,且均为外关键字。

8)商品(Item),该表由4个字段组成,包括商品的ID,商品的名称,商品ID是该表的关键字。

9)库存(Stock),该表由17个字段组成,包括库存所在的仓库的ID,库存所属的商品的ID,它们是该表的关键字,且均为外关键字。

各表的字段数、行数、每行长度及表的尺寸如下表所示:

表名	字段数	序数 (千行)	行长度 (字节)	典型表长度 (KB)
仓库	9	0.001	89	0.089
分店	11	0.01	95	0.950
顾客	21	30	655	19,650
历史	8	30	46	1,380
订单	8	30	24	720
订单明细	10	300	54	16,200
新订单	3	9	8	72
库存	4	100	82	8,200
货物	17	100	306	30,600
TPC-C 数据库大小	76,823.039KB			

三、事务定义

TPC-C 共定义了5种事务,这些事务的执行构成了整个 TPC-C 测试过程,下面具体进行描述:

1)订购事务,对顾客提出的新的订货要求进行处理。每张订单含有5至15行商品,该事务描述了商业环境中的完整的订购过程,代表了执行频繁,响应时间短的数据更新事务。该事务占 TPC-C 测试的比重最大,并且测试中1%的订购事务需要反做(UNDO),用此来模拟操作失败的情况。

2)付款事务,对顾客的付款进行处理,需处理修改顾客表,分店表,仓库表的数据,其中,40%的顾客按顾客号检索,60%的顾客按顾客名检索,描述了真实的付款过程,代表了执行频繁,响应时间短的数据更新事务。

3)订购情况事务,对顾客的最后一个定单进行查询,代表了执行不很频繁,响应时间短,不对数据进行修改的只读型事务。

4)发货事务,完成与仓库下属的各分店中最久未付货的订单配送手续,以批处理的方式,一次处理十个未发送的定单,一个数据库事务内可以完成一个或多个(至多十个)定单的付货过程,但每一个定单的付货都要求在一个数据库事务内完成,代表了执行不很频繁,响应时间长的数据更新事务。实际测试时,该事务的处理不是以交互方式,而是以后台执行方式,同时,它的结果不是显示在终端上,而是记录在结果数据文件内。

5)库存情况事务,查询最近售出的,库存在一定数量以下的所有商品,代表了执行不很频繁,响应时间长,一致要求较低的只读型事务。

事务要求按照一定的比率 and 响应时间执行,如下表所示。

事务类型	最小混合度	最小键盘等待时间(秒)	90%事务响应时间期限(秒)	最小事务间隔时间(秒)
订购	(剩余)	18	<5	12
付款	>43.4%	3	<5	12
订购情况	>4.3%	2	<5	10
发货	>4.3%	2	<5	5
库存情况	>4.3%	2	<20	5

四、OO 库上的设计与实现

我们实现 TPC-C 的面向对象数据库所采用的是对象数据库管理组织(ODMG)制定的面向对象数

数据库的标准 ODMG-93模型,对于一个 OO 数据库系统,按 TPC-C 规范中的关系模型去实现测试是不合适的,因此,必须用 OO 思想重新理解 TPC-C 规范,用 OO 方法来定义 TPC-C 测试数据库的模式和事务。

在定义 TPC-C 的 OO 数据库时,我们的原则是,必须正确描述 TPC-C 定义的数据内容和保持原来的语义和完整性约束。与关系方法相比,有以下几个方面的不同:

1) 用类加上其外延来表示一个数据表。表的结构用类(class)表示,表的字段对应于类的属性,表的内容用外延(Extent)表示,表的元组对应于一个对象。这样为9张表分别定义了9个类。

2) 对于表之间的联系,不像关系模型用外关键字表示,而是用联系(Relationship)属性表示。

3) 将对表的操作,如插入,删除,搜索等提炼为类的方法,所有对类操作都由方法来完成,充分体现面向对象的封装性和方法的信息隐藏等优点。

4) 为提高信息的抽象程度,充分利用对象继承性所带来的结构清晰、代码量小的优点,在9个类上又进一步进行数据抽象,定义了三个虚基类,将一些相同的属性定义在基类中。

图4给出了完整的 TPC-C 的 OO 模式。其描述方法采用的 OMT 形式[4],限于篇幅,省略了对象类的属性和方法的详细定义。

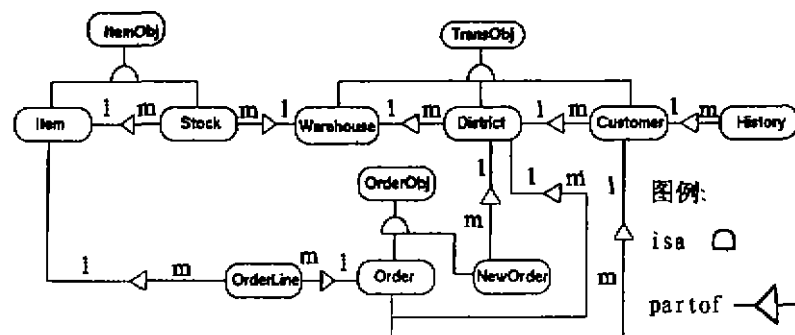


图4 TPC-C的ODMG模式

按照图4的定义,我们应可以用与 C++ 联编的 ODMG 语言进行编程。为举例起见,下面给出了 Customer 类模式定义:

```
class TransObj, public d Object{
public:
// Attributes
d ULong id;
d ULong ytd;
};
extern const char dCustomer[], hCustomer[], oCustomer[];
```

• 60 •

```
class Customer, public TransObj{
public:
// Attributes
d String firstName;
d String data;
// Relationships
d Rel Ref(District, dCustomer) cDistrict;
d Rel Set(History, hCustomer) cHistory;
d Rel Set(Order, oCustomer) cOrder;
// Operations
Customer(ShortId tcustId, District* dist);
static T RTCODE SearchByld(int, d Ref(District)&,
d Ref(Customer)&);
void ModifyDeliveryCount();
// Extent
static d Ref(d Set(d Ref(Customer))) customers;
static const char* const extent name;
};
```

其中 TransObj 是 Customer 的超类, Customer 继承 TransObj 的所有公共特性的方法, cDistrict 是一个联系属性, 指向一个顾客所属的分店, cHistory 是一个联系集合, 指向一个顾客之下的所有历史, warehouses 为一静态集合类型, 定义了 Warehouse 的外延, extent name 定义了外延的名字。SearchByld() 为一个方法, 完成按 ID 检索一个顾客。

结束语 根据本文所述的 TPC-C 的 OO 设计, 我们已经实现了一套符合 ODMG/93 的 TPC-C 测试程序, 并完成了对一个 OO 数据库原型系统的测试。测出了该系统在 Pentium/133 主机和 solaris/x86 操作系统上的运行性能。通过对测试后结果的分析和评价, 发现了影响系统的几个关键因素, 主要有:

持久性对象引用的实现方法, 外延的索引结构、死锁的检测方法、事务的重启动等待时间等。这些因素对于该系统的改进有重要的作用。我们的下一步的工作将进行分布式 TPC-C 测试的设计和实现。

参考文献

- [1] Transaction Processing Performance Council (TPC), TPC Benchmark C Standard Specification, Revision 1.0, 13 August 1992
- [2] The Object Database Standard, ODMG-93, edited by R. Cattel, Morgan Kaufmann Pub., 1996
- [3] M. J. Carey et al., The OO? Benchmark, ftp://ftp.cs.wisc.edu/pub/techreports
- [4] [美] P. Coad, E. Yourdon, 面向对象的设计, 邵维忠等译, 北京大学出版社, 1992