ORM中视图的扩展应用

吴 纲[1] 刘 凡[2]
(1.武汉职业技术学院 计算机学院,湖北武汉 430074;
2.湖北省纪委办公厅综合信息处,湖北武汉 430071)

摘 要 目前应用软件开发大多使用面向对象设计,采用如Java、C#等面向对象的语言来实现,而数据库一般还是RDBMS(关系数据库管理系统)关系数据库。由于对象模型和关系模型存在阻抗不匹配,因此需要在两个模型间进行映射与转换,即ORM(Object Relation Mapping,对象关系映射)。视图是关系数据库中的一种抽象机制,主流关系数据库都增加了一些新的视图特性,若能灵活应用于ORM中可以发挥很大的作用。本文结合某移动公司运营分析支撑系统的开发经验,以SQL Server为例,探讨在ORM中如何利用视图机制和这些新的视图特性。

关键词 ORM;视图;索引视图;分区视图
中图分类号 TP31 **文献标志码** A **文章编号** 1671-8100(2009)01-0039-06

1 ORM与视图简述

1.1 ORM

几乎所有的程序都存在对象/关系数据库。在业务逻辑层和用户界面层中往往是面向对象的,当对象信息发生变化的时候,需要把对象的信息保存在关系数据库中。由于RDBMS是以二维表为基本管理单元的,对象模型最终是由二维表及表间关系来描述的,需要从对象模型向数据库概念模型的映射。ORM在对象模型和关系数据库模型之间产生一个映射,使前台的面向对象方法中使用的各种对象映射到后台的关系数据库中。ORM主要有三种类型:

1)嵌入式类型。该类型直接将SQL语句嵌入到应用对象类中,这种方法扩展性比较差,主要用于比较小型的应用。

2)数据层类型。该类型中每个业务类对应一个数据存取类,SQL语句集中在数据类中,这种方法需要编写更多代码,不过很多代码自动生成工具如CodeSmith等可以辅助完成编写工作。

3)持久层类型。该类型建立一个持久层,自动完成业务对象到数据库的映射(一般通过反射机制或者XML配置文件实现)。持久层设计比较复杂,但应用程序的对象逻辑结构比较清晰,维护方便[1]。

现在通常的做法是利用诸如Hiberate等持久层框架完成面向对象的设计,性能扩展和优化主要是在应用服务器对对象一级的缓存来实现,甚至完全抛弃DB(数据库)设计这个层次的方法[2]。这种方法不能充分利用关系数据库的性能优势,在实际运用中往往需要优化关系数据库,尤其体现在与OLAP(On-Line Analysis Processing,联机分析处理)有关的应用方面。因此应该根据实际情况,采用比较灵活、可控的轻量级ORM设计。

1.2 视图

视图是数据库中一种抽象机制(逻辑数据独立性),即数据库外模式。标准视图作为数据库中的一种实体,实际上存在的只是它的脚本。视图的所有用处都是围绕一个基本概念——给用户呈现与物理结构或物理数据无关的数据格式。视图的用途有总结数据、过滤数据、数据库安全、提高数据库和应用程序性能以及数据分割等[3]。

UML图和ER图的映射主要是数据层面,对于方法层面,主要在应用对象中实现,一些方法可以看作是对主要查询或者存储过程的封装,也就

收稿日期:2008-12-06
作者简介:吴 纲,男,武汉职业技术学院教师,硕士,主要从事计算机应用技术方面的教学与科研工作。

是常用的查询 SQL 语句都可以映射到某个方法中。一些优秀的 ORM 设计，如 iBATIS 用统一的配置文件来管理查询语句，这样就更加方便设计和统一管理。通过视图，可以很灵活地映射很多标准的面向对象机制，这是 ORM 设计中一种值得注意的机制。

2 逻辑设计

2.1 传统的对象/关系模型映射规则

对象模型向数据库概念模型的映射就是向数据库表的变换过程，经典的变换规则可以简单归纳如下[4][5]：

1)一个对象类可以映射为一个以上的库表，当类间有一对多的关系时，一个表也可以对应多个类。

2)关系(一对一、一对多、多对多以及三项关系)的映射可能有多种情况，但一般映射为一个表，也可以在对象类表间定义相应的外键。对于条件关系的映射，一个表至少应有 3 个属性。

3)单一继承的泛化关系可以对超类、子类分别映射，也可以不定义父类表而让子类表拥有父类属性；反之，也可以不定义子类表而让父类表拥有全部子类属性。

4)对多重继承的超类和子类分别映射表，对多次多重继承的泛化关系也映射一个表。

5)对映射后的库表进行冗余控制调整，使其达到合理的关系范式。?

2.2 视图模拟泛化关系

泛化关系多是用表来模拟，实际上如果采用视图可以更加灵活。例如在移动资源管理系统中，需要处理多种厂家设备的性能数据，每个厂家的数据有很多特殊字段，偶尔也需要查询，但更多关注的是这些厂家的数据中一些公共字段，设计时很显然就可以对这些公共数据建立一个对象类，比如“pmdata_icell”，同时对每个厂商建立一个子类，如“pmdata_icell_ALC”、“pmdata_icell_ZTE”等，设计过程中很自然地可以在数据库中创建两个对应的实体表，然后创建一个视图“pmdata_icell”作为它们的基类，类图如图 1 所示。

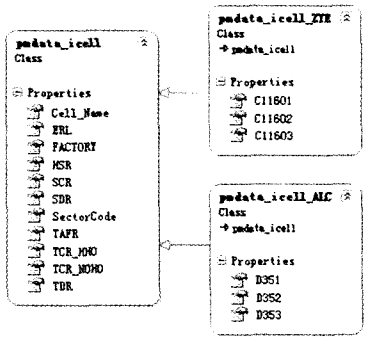

图 1 性能参数模型类图

对应的视图定义如下：

```
Create VIEW dbo. pmdata_icell AS
SELECT [Cell_Name], SECTORCODE, ERL, SCR, SDR,[TCR_NOHO],[TCR_HO], TAFR, TDR, HSR,
[ERL_TCH], SAFR,[TAFR_HO],D363, D356,'阿尔卡特' as FACTORY
FROM[pmdata_icell_ALC]
UNION ALL
SELECT [Cell_Name], SECTORCODE, ERL, SCR, SDR,[TCR_NOHO], [TCR_HO], TAFR, TDR, HSR,
[ERL_TCH], SAFR, [TAFR_HO], C11616,C11607,'中兴' as FACTORY
FROM [pmdata_icell_ZTE]
```

视图同样也可以模拟子类，例如载波告警信息主要是从当前性能数据中获取，但是其中载波数信息还要参照系统参数的信息计算得出，用类来设计就是载波告警性能的子类，同时它和系统参数类有关联关系，从 SQL 语言角度来看就是一个多表查询视图，类图如图 2 所示。

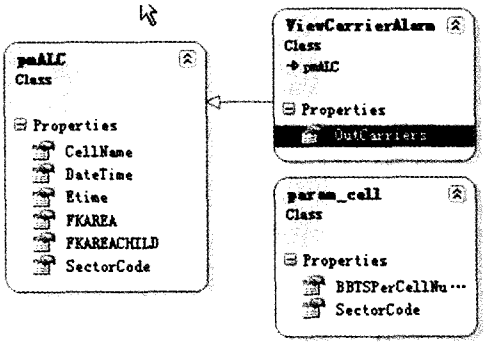

图 2 载波告警模型类图

相应的视图定义如下：

```
Create VIEW ViewCarrierAlarm AS
```

```
SELECT pmALC. SECTORCODE as sectorcode, pmALC. Cell_Name,
pmALC. FKAREA, pmALC. FKAREACHILD,
param_cell. BBTsPerCellNumber AS Carriers,
CEILING(pmALC. D356 / 8) AS GoodCarriers,
param_cell. BBTsPerCellNumber - CEILING(pmALC. D356 / 8) AS outCarriers,
pmALC. datetime, pmALC. eTime
FROM pmALC INNER JOIN
param_cell ON pmALC. SECTORCODE = param_cell. SectorCode
```

需要指出的是,当视图模拟泛化关系时,适合扁平化类关系的设计,比较复杂的类设计映射起来比较复杂。

2.3 iBATIS 框架

Hiberate 和 iBatis 是现在比较流行的开源 ORM 框架。Hiberate 功能很强大,直接从数据库模式生成实体类和基本的配置文件,而且大部分情况下不需要编写 SQL 语句,会较大提升开发效率,但这也产生很多的局限性,尤其是对环境的要求较高(数据库设计、对象设计、团队的协作等)。相对而言,iBATIS 小巧灵活,可扩展性高,封装了数据访问层(事务、缓存、异常、日志等),并提供了数据访问层框架支持。利用 iBATIS 可以轻松实现代码和 SQL 的分离,只要 SQL 能够解决的问题,iBATIS 就能较容易解决,同时也使整体项目设计对某一框架的依赖性变小(因为 iBATIS 是非侵入性的)。这将极大地降低项目风险,减少解决复杂问题的时间,使项目的维护变得简单[6]。

iBATIS 采用混合机制结合了关系数据库的强大功能和面向对象设计,而且对数据的封装和层层映射更加自然,可以很方便地使用 RDBMS 的方式来处理数据,容易支持 OLAP 类型应用,也更容易发挥索引等数据库性能优化的能力。在面向对象设计层,也和数据表有了清晰的对应机制,将大多数 SQL 定义甚至方法的底层支持都放到数据访问层中,也就是 SQL Mapping,对于数据访问层设计人员,主要是熟悉关系数据库 SQL 语句就可以充分发挥 SQL 作用。

3 物理设计及性能考虑

逻辑设计最终要映射到物理设计,关系数据库提供了很多新的机制,对视图的性能等做了很大的优化和功能扩展,有力支撑视图作为一种 ORM 选择。

3.1 索引视图

关系数据库中,复杂的视图有时候不能很好地利用实体表的索引,例如 SQL Server 很多"union all"视图不能有效利用索引。SQL Server2000 开始提供了索引视图的功能,可以为视图创建索引,即对视图创建一个唯一的聚集索引。索引视图是被具体化了的视图,即它已经过计算并存储。索引视图尤其适于聚合许多行的查询,索引视图还具有使用标准索引不能获得的其它性能优点。索引视图能够在诸如预先计算聚合并将其存储在索引中,从而最大限度地减少在执行查询期间进行成本很高的计算;预先联接表并存储生成的数据集。索引视图如果使用合理会十分有用,因为它们能够显著地提高查询的性能。但是,由于聚集索引增加的性能,数据库引擎必须在视图基表的所有事务过程中维持那个索引,即不太适于经常更新的基本数据集。

例如前面所说的 ViewCarrierAlarm 视图从参数表 param_cell 取 BBTsPerCellNumber 数据,对 ViewCarrierAlarm 的查询可以利用性能 pmALC 的索引,但如果涉及到 BBTsPerCellNumber 的查询性能就不理想,因此可以在视图上定义索引。相应索引定义如下:

```
——设置一些系统参数
SET QUOTED _ IDENTIFIER, ANSI _ NULLS ON;
GO
——修改视图定义,带入 with Schemaabing
CREATE VIEW ViewCarrierAlarm WITH SCHEMABINDING AS
SELECT pmALC. SECTORCODE as sectorcode, pmALC. Cell_Name,
pmALC. FKAREA, pmALC. FKAREACHILD,
param_cell. BBTsPerCellNumber AS Carri-
```

```
ers,
    CEILING(pmALC. D356 / 8) AS GoodCarriers,
    param_cell. BBTsPerCellNumber - CEILING(pmALC. D356 / 8) AS outCarriers,
    pmALC. datetime, pmALC. eTime
    FROM pmALC INNER JOIN
    param_cell ON pmALC. SECTORCODE = param_cell. SectorCode
    GO
    ——在视图上创建索引
    CREATE UNIQUE CLUSTERED INDEX IDX_V1
    ON ViewCarrierAlarm (BBTsPerCellNumber)
```

实际测试中，尤其在数据数量巨大的情况下，使用或不使用索引视图查询某个小区载波告警情况时，性能差异非常显著。若不使用索引视图，随记录数增加性能下降得非常厉害，当超过 100000 条记录后基本不可忍受，而使用索引视图则性能下降比较平滑。查询性能如图 3 所示。

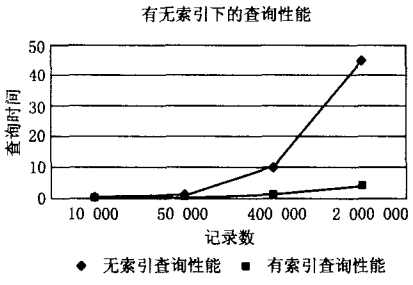

图 3 查询性能对比

需要指出的是，如果采用索引视图后还存在性能方面问题的话，可以考虑用存储过程来改进，对于一些统计计算类型的视图，应考虑将中间计算结果保留到物理表中。

3.2 大数据量处理与分区视图

现在包括电信运营分析系统在内的各种应用数据量越来越大，对数据库性能优化就必不可少，除了内部调校、表模式优化、冗余数据外，无非是下面三种手段及其组合：

1)分散(分片)：将海量数据表分割成多个表，如按 IP 地址散列、按时间切割等。一些

数据库系统内置分区视图功能，如 Oracle、SQL Server 等；

2)分布：多台机器的分布式存储，著名的如 Google 的分布式存储；

3)分层(级)：多级存储访问，如内存文件系统、内存数据库、Memcache 缓存等。

当数据表很大的时候，可以根据数据的时间局部性和空间局部性原则进行分片设计。数据库 Sharding(分片)技术，不是特定数据库软件功能，而是在具体技术细节之上的抽象处理，是用程序逻辑来实现数据分片，是水平扩展的解决方案，其主要目的是为突破单节点数据库服务器的 I/O 能力限制，解决数据库扩展性问题。一般有两种分片方法：横向分片和竖向分片。

开源数据库如 MySQL 里面广泛使用数据库分片技术，用户可以自己编写程序逻辑实现，也可使用一些 MySQL Proxy＋、HSCALE 等分片软件来实现。

商业数据库如 Oracle、SQL Server 等除此之外，还内置标准化的分片机制——分区视图。

分区视图在一个或多个服务器间水平连接一组成员表中的分区数据，使数据看起来就象来自一个表，可以通过水平数据分区或垂直数据分区分解数据表，数据的位置对应用程序是透明的。还可利用分布式分区视图来分解数据表，分布式分区视图可用于实现数据库服务器联合体，这种通过分区数据形成数据库服务器联合体的机制使用户能够扩大一组服务器，以支持大型的多层 Web 站点的处理需要[7]。

例如处理历史数据可以按照行政区域来分割，系统管理 17 个县区，对每个县区的性能数据创建一张物理表并指定约束范围，然后创建一个分区视图。分区视图定义如下：

```
    ——创建一个分区表：
    CREATE TABLE [dbo]. [pmdataHistory_wh] (
    [datetime][datetime] NOT NULL,
    [FKAREA][int] NOT NULL , CHECK ([date_key] =1),
    [sectorcode][int] NOT NULL , …
    )
    ALTER TABLE[pmdataHistory_wh] ADD PRIMARY KEY ([sectorcode],[datetime])
```

——创建其它分区表，每个地区一个，如 pmdataHistory_hg

——创建相应的分区视图

CREATE VIEW [dbo].[pmdataHistory] AS

SELECT * FROM [dbo].[pmdataHistory_wh]UNION ALL?

SELECT * FROM [dbo].[pmdataHistory_ez] UNION ALL

SELECT * FROM [dbo].[pmdataHistory. yc]UNION ALL

…

这些性能优化都是独立操作的，不需要对应用程序逻辑进行修改。随着数据量的增大，还可以很方便地扩展到分布式的分区视图。

4 结 语

ORM 可以透明地插入对象级别的缓存，是细颗粒度的缓存，当一个 web 类型的 OLTP 应用的数据量和用户量达到一个相当大的规模的时候，应用级缓存是有效的性能提升手段，因此 ORM 在大数据量和大负载的 OLTP 应用中对性能提升能起到很大作用。但不管使用面向对象方法还是传统的方法，软件设计本质都是对现实概念的抽象，这才是最重要的原则。ORM 中关于框架的使用，不要盲目追求所谓最复杂功能、最强的方案，要因地制宜，不能为了面向对象而面向对象。

主流数据库的视图功能都有了很强的扩充，与 ORM 结合可以提高灵活性，不过要注意索引视图和分区视图使用有一些约束，同时要经常利用数据库的工具，如查询剖析器等，分析和调校性能。

关系数据库是最成功的软件领域之一，在可以预见的将来还是主流，一些重量级的方案提供了很复杂的自动映射功能，试图屏蔽关系模型和对象模型的差距，仅仅在应用层进行性能优化完全替代数据层的性能优化。这种方案并不能充分利用关系数据库的功能，应该尽量采用轻量级、灵活性和可扩展性强的 ORM 框架，如 iBATIS，从而能够在应用层和数据层进行性能优化。

参 考 文 献

1 陈春玲，朱常宝，严劲. 采用 ORM 技术的软件开发方法研究[J]. 计算机与现代化，2006(6)：55－57，60.

2 肖光星，钟海云. 轻量级 ORM 持久层的研究与实现[J]. 南昌工程学院学报，2008，27(3)：55－59.

3 Ryan K. Stephens，Ronald R. Plew. 何玉洁等译. 数据库设计[M]. 北京：机械工业出版社，2001，9.

4 Scott W. Ambler. Mapping Objects to Relational Databascs [EB/OL]. http://www－900. ibm. com/developerWorks/cn/components/mapping－to－rdb/index_eng. shtml，2000－07

5 Mark L. Fussell Foundations of Object Relational Mapping [C]. White Paper，ChiMu Corp.，1997：20－22

The Extended Application of View in ORM

WU Gang[1]，Liu Fan[2]

(1. Wuhan Institute of Technology，Wuhan430074，China；
2. General Office of Hubei Provincial Commission for Discipline Inspection of the CPC，Wuhan 430071，China)

Abstract：The present software is object-oriented and developed by using Java，C＋＋ or other computer programming languages with the database remains RDBMS relational database. Due to the inconsistence of the object model and relational model，mapping and conversion between the two models is required，that is Object Relation Mapping (ORM). Because the database is an abstract mechanism and some new characteristics of view have been added to mainstream relational database，which can be very effective if used in ORM. Through analy-

sis on the development of the support system of a mobile operation, this paper takes SQL Server as an example to discuss the ways to use the view mechanism and its new characteristics with systems performance ensured at the same time.

Key words: ORM; View; Indexed View; Partition View

(责任编辑:谭银元)

(上接第35页)

2 满一新,李素玉等.船机检修工艺[M].人民交通出版社,1993.

3 武汉水运工程学院船机教研室编,船舶机械制造与修理工艺学[M].人民交通出版社,1987.

4 陈大荣,船舶柴油机设计[M].人民交通出版社,1985.

5 钱耀男,船舶柴油机[M].大连海事大学出版社,2000.

6 余宪海,船舶柴油机维修工艺[M].武汉水运工程学院,1974.

On Recondition and Improvement of Diesel Engine Cylinder Head's Starting Valve

CHEN Ping

(Zhejiang International Maritime College, Zhoushan 316021, China)

Abstract: Through analysis on troubles occurring in operating the diesel engine, the causes of the failure and troubles are concisely positioned with the repairing process as a reference in this paper, which as well introduced essential repairing craftwork for the machine's failure and proposes feasible and effective ways to improve cylinder head's structure so as to avoid such accidents.

Key words: diesel engine; cylinder head's starting valve; Craftwork of repairing

(责任编辑:谭银元)