浙江大学

硕士学位论文

数据库物理模式推荐框架研究及其在OSCAR中的实现

姓名：丁朝辉

申请学位级别：硕士

专业：计算机应用

指导教师：陈刚

20060301

摘要

在日常数据库系统设计和部署过程中，数据库物理设计是一个关键过程。数据库系统的性能很大程度上依靠数据库物理设计。但是，越来越多、越来越复杂的物理设计特征使得这个过程也日趋复杂化。

数据库系统及其应用也变得越来越复杂、多变，从而大大增加了数据库系统管理的难度及对DBA技能的要求，也使得数据库的管理成本越来越高。自动物理数据库设计工具能通过提供适当的物理设计减轻DBA的工作负担，从而减轻降低数据库系统的总成本。

本文研究了数据库物理设计特征，提出了一个通用的数据库物理模式推荐框架(General Physical Database Schema Recommender Framework)。该框架把数据库系统的物理设计分成为两个阶段：工作负载分析阶段和设计推荐阶段。这两个阶段中，三个组件协同进行工作：负载分析组件(WAC)，特征空间构造组件(FCC)，最优特征搜寻组件(OSC)。在OSCAR关系数据库系统中实现了一个索引推荐器的原型，实验结果数据充分体现了整个框架的可行性和有效性。

第一章是绪论，介绍了一下DB和DBMS的概念、本研究的背景以及必要性。

第二章介绍了物理数据库设计的一些基本概念，并在此基础上讨论了目前部分主流商业关系数据库管理系统在物理数据库自动设计方面的相关研究。

第三章介绍了通用的物理数据库自动设计(General Physical Database Schema Recommender，GPDSR)框架。

第四章介绍了我们在神舟OSCAR关系数据库系统中实现了的索引推荐器的原型系统，同时使用标准的数据库评测基准对系统进行了测试。

最后在第五章，总结了本文所论述的工作以及主要贡献和存在问题，并对今后的发展方向做了一定的介绍和建议。

关键字： 数据库管理系统；物理数据库设计；数据库优化；描述逻辑

Abstract

Physical database design is a key phase in devising and deploying a database system to improve overall system performance. However, the appearance of more and more complicated physical design features make the design procedure very complicated.

Database systems and their workloads are become more and more complex and dynamic, and the costs of management are become more and more expensive. Automated tools for physical database design can help reduce the total cost of ownership (TCO) of databases by reducing the DBA's burden in determining the appropriate physical design.

The performance of an enterprise database system can depend crucially on its physical database design. Automated tools for physical database design can help reduce the total cost of ownership (TCO) of databases by reducing the DBA's burden in determining the appropriate physical design.

In this dissertation, A GPDSR (General Physical Database Schema Recommender) framework is provided to support automatic physical database design for relation database systems. The framework divides automatic physical database design into two phases, the workload mining phase and the design advising phase, including three collaborative agents, Workload Analysis Component (WAC), Feature-Space Construction Component (FCC), and Optimal-Feature Search Component (OSC). An index advisor for OSCAR relational engine is prototyped, and experiment results show its feasibility and efficiency.

In the first chapter, we introduce the concepts of database and database management system, and the context of the research..

In the second chapter, We review related studies on automatic physical database design and their limitations.

In the third chapter, we introduce the GAPDD framework.

In the fourth chapter, we introduce a prototyped index advisor for OSCAR database engine with some advising results for database schemas and workloads from generally used database benchmarks.

In the fifth chapter, we conclude our work by pointing out the contributions and faultiness of OSIA and future work to perfect it.

Keywords: Database management system.; Physical Database Design; Database tune; Description Logics

第一章 绪论

信息化时代的来临，使以计算机技术为基础的信息科学与技术在经济和社会生活各个领域得到了极为广泛的应用。数据库技术已经成为现代信息技术的重要组成部分，是现代计算机信息系统的基础和核心。

1.1 引言

在现代社会里，数据库和数据库系统已经成为社会生活中不可缺少的一部分。我们每天都会或多或少地和数据库发生某些联系。例如：我们可能会去银行存款；可能会为一篇论文查找数字图书馆；可能会到民航售票点预购飞机票；……；所有这些活动都会涉及到对数据库的存取。

数据库和数据库技术对不断增长的计算机的应用产生着重要的影响。可以毫不夸张地说：几乎所有涉及到计算机应用的领域中，数据库都担当着非常关键的角色&。如：商业、工程、医药、教育和图书馆，等等。

- **数据库（Database，DB）**

数据库是一个相关数据的集合。这里，数据指的是可以被记录并拥有确切含义的已知事实。数据库的规模可以是任意的，而且它的复杂程度也是可变的。例如：一个私人的通讯录，可能只有上百条记录，并且每条记录的结构也很简单。然而，一个大型图书馆的卡片目录可能会有超过百万条记录，并在不同的分类下存储——如按照作者的名字、主题或出版社分类等。有的数据库会有更多的记录，数据量可达到几百 GB 甚至 TB 级[Cra06]。显然，如此庞大的数据只有通过适当的组织和管理，才能有效满足用户对这些数据的检索和更新的需要。

- **数据库管理系统（Database Management System， DBMS）**

数据库管理系统是一个帮助用户创建和管理数据库的应用程序的集合。因此，DBMS 也就是一个可以帮助完成定义、构造和操纵数据库等处理目的的统一软件系统。

  - 定义（Defining）数据库需要指定数据库中所存储数据的数据类型、数据结构和对数据的约束条件。
  - 构造（Constructing）数据库指的是在由 DBMS 控制下，在某一介质上存

储数据的过程。

- 操纵（Manipulating）数据库包括查询数据库以检索指定数据，更新数据库以反映真实世界中的改变以及从数据中产生报表等功能。

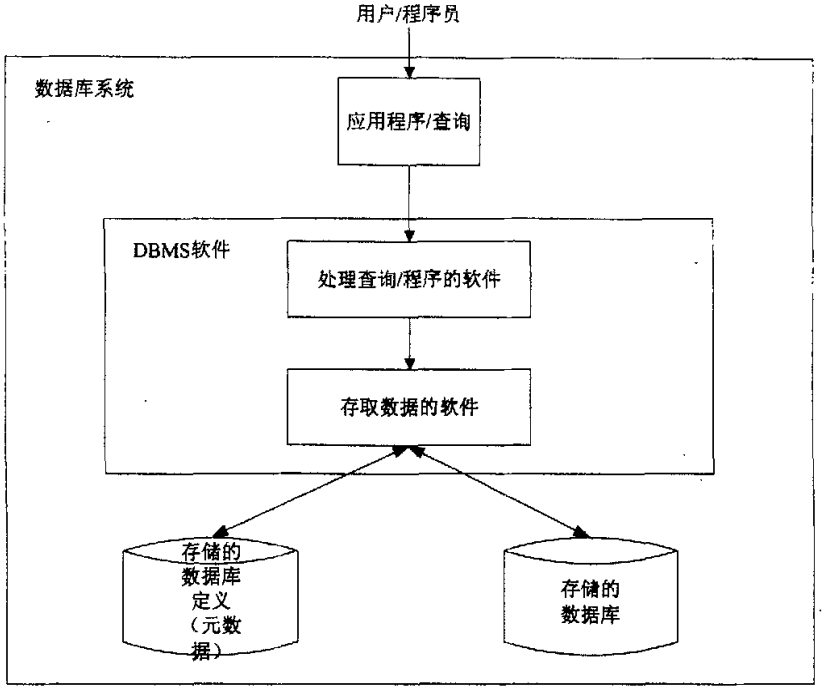

图1.1 一个简化的数据库系统环境

我们把数据库和 DBMS 软件合称为**数据库系统**，通过 DBMS 资源配置使系统资源为 DBMS 分配使用。图 1.1 表示的是一个简化的数据库系统环境[Elm03]。

数据库技术从 20 世纪 60 年代中期产生至今仅仅 30 多年的历史，就已经经历了层次和网状数据库、关系数据库、面向对象数据库三代的演变，目前已经发展成为一个庞大的数据库家族。尽管传统的层次和网状数据库仍在使用，面向对象数据库的发展也非常迅速，但关系数据库仍占主导地位，应用极其广泛。关系数据库是支持关系模型的数据库系统，它使用数学方法来处理数据库中的数据。E.R.Codd 在[Cod70]中首次提出关系模型，开创了数据库系统的新纪元。以后他连续发表了多篇论文[Cod71; Cod73]，奠定了关系数据库的理论基础。30 年来，关系数据库系统的研究取得了辉煌的成就，涌现出许多性能良好的商品化关系数据库管理系统，如著名的 DB2、Oracle、Sql Server、Ingres、Sybase、Informix 等。

1.2 项目背景

1.2.1 严峻的形势

在当今的IT世界上，芯片（包括CPU和内存条）做得越来越小，速度越来越快，可靠性越来越高，价格却越来越便宜；磁盘的存储量越来越大、存取速度越来越快、可靠性越来越高、单位字节的价格却越来越便宜；网络通讯速度越来越快、越来越稳定，系统变得越来越复杂。而唯一形成鲜明对比的是人，人并没有变得越来越聪明、越来越可靠，但是人员的费用却是越来越高。

例：存储的管理成本是物理设备的成本的两倍以上。图1.2[Cha04]表示的是分别在1984年和2000年，300万美元存储系统的成本分配。

而在一个数据库系统中，人员的费用（包括培训、运行、维护等费用）占总费用的比例更是达到了81%[Cha04]。

管理费用已经成为IT总费用的主要部分。在硬件费用不断下降的同时管理费用在不断上升，在今天，信息技术已经成为商务活动的核心，越来越多的顾客和供应商直接通过网络平台进行交易，这得益于IT基础设施的可靠性。可以说，经营业绩是和IT性能直接挂钩的。据商务分析统计，一个电子商务网站一分钟的系统“Down”机导致的损失可达数千万美元[Ora00]。而对信息技术信任度的增强和数据量的爆炸性增长是以需要越来越多的系统管理人员为代价的。但是，有经验的技术人员是有限的，供不应求的局面导致数据库管理员（DBA）的薪水呈螺旋型上升。

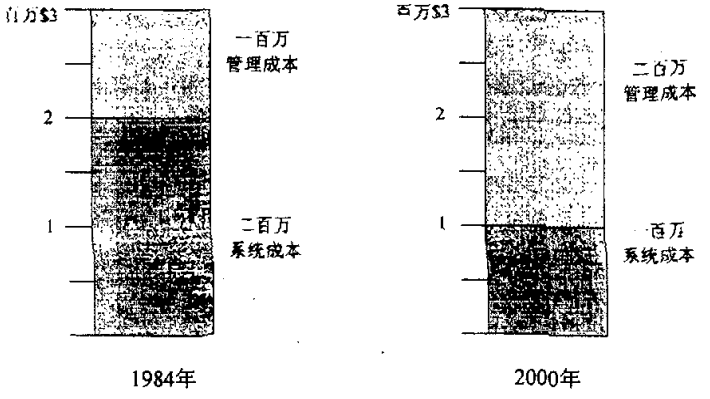

图1.2

但是，商场如战场，激烈的竞争需要降低管理操作成本。因此，IT 管理人员经常被要求做更多的事情。

管理的复杂性在不断地增长。

应用程序的复杂性在增大，程序也变得越来越大。数据库的工作负载越来越复杂，更加动态，数据量在迅速增长，TB 级的数据量已经不再是例外。

针对以上问题，DBMS 的供应商们提供了相应对策来面对挑战：

- 扩大现有特征的范围，如：新的存取结构、综合优化、复杂的硬件技术（如集群技术、大规模并行处理技术等）。
- 在服务器端增加新的特征，如：对象（Objects），可扩展标记语言(XML)，联机分析技术（OLAP），数据挖掘（data mining），ETL（数据抽取、转换、清洗、装载），等等。

而对一个现代数据库进行管理和优化需要非常高水平的专门技术。

1.2.2 数据库性能优化

数据库的性能优化一直是数据库管理中的重要环节，也是最复杂的内容之一。对于数据库管理人员而言，数据库建立起来后，主要的工作内容有 3 个：优化、恢复和备份。要实现高效地运行数据库，优化性能要占大部分的工作量（如果数据量很小，当然是可以不进行优化）。

在 20 世纪 70 年代，美国的 TonyDaugherty 提出了数据库性能优化(Performance Optimization)的概念，不仅对于数据库应用系统的研究起了重要作用，而且对于数据库性能问题的研究也起了开创性的作用。到 20 世纪 80 年代，Performance Tuning Basics 中提出“性能调整(Performance Tuning)”的概念，论及性能调整是一项通过修改系统参数、改变系统配置(硬件调整)、优化组件应用来改变系统性能的活动。

随着 DBMS 复杂性增强以及新功能的增加，使对数据库管理人员的技术需求和熟练数据库管理人员的薪水支付都在大幅度增长，导致企业人力成本支出也在迅速增加。随着关系数据库规模和复杂性的增加，系统调整和管理的复杂性相应增加。今天，一个数据库系统管理员（DBA）必须了解磁盘分区，并行查询执行，线程池和用户定义的数据类型。他的主要职责如下[Abr02]：

但是，商场如战场，激烈的竞争需要降低管理操作成本。因此，IT 管理人员经常被要求做更多的事情。

管理的复杂性在不断地增长。

应用程序的复杂性在增大，程序也变得越来越大。数据库的工作负载越来越复杂，更加动态，数据量在迅速增长，TB 级的数据量已经不再是例外。

针对以上问题，DBMS 的供应商们提供了相应对策来面对挑战：

- 扩大现有特征的范围，如：新的存取结构、综合优化、复杂的硬件技术（如集群技术、大规模并行处理技术等）。
- 在服务器端增加新的特征，如：对象（Objects），可扩展标记语言(XML)，联机分析技术（OLAP），数据挖掘（data mining），ETL（数据抽取、转换、清洗、装载），等等。

而对一个现代数据库进行管理和优化需要非常高水平的专门技术。

1.2.2 数据库性能优化

数据库的性能优化一直是数据库管理中的重要环节，也是最复杂的内容之一。对于数据库管理人员而言，数据库建立起来后，主要的工作内容有 3 个：优化、恢复和备份。要实现高效地运行数据库，优化性能要占大部分的工作量（如果数据量很小，当然是可以不进行优化）。

在 20 世纪 70 年代，美国的 TonyDaugherty 提出了数据库性能优化(Performance Optimization)的概念，不仅对于数据库应用系统的研究起了重要作用，而且对于数据库性能问题的研究也起了开创性的作用。到 20 世纪 80 年代，Performance Tuning Basics 中提出“性能调整(Performance Tuning)”的概念，论及性能调整是一项通过修改系统参数、改变系统配置(硬件调整)、优化组件应用来改变系统性能的活动。

随着 DBMS 复杂性增强以及新功能的增加，使对数据库管理人员的技术需求和熟练数据库管理人员的薪水支付都在大幅度增长，导致企业人力成本支出也在迅速增加。随着关系数据库规模和复杂性的增加，系统调整和管理的复杂性相应增加。今天，一个数据库系统管理员（DBA）必须了解磁盘分区，并行查询执行，线程池和用户定义的数据类型。他的主要职责如下[Abr02]：
行，线程池和用户定义的数据类型。他的主要职责如下[Abr02]：

- 模式定义：通过执行一系列数据定义语言，创建初始的数据库模式；
- 存储结构和访问方法的定义：如定义表的分区、创建索引等；
- 模式和物理组织结构的修改：为适应相关需要的变化或提升性能的需要，修改模式或物理组织结构；
- 为使用数据库的各个用户授予合适的访问权限；
- 对数据库进行日常维护：包括定期对数据库进行备份、在需要时进行硬件升级、监控数据库上的应用程序的运行情况等。

基于上述原因，数据库系统自调优和自管理工具的需求增加，对数据库自调优和自管理的研究也逐渐成为热点。

这类项目至少包含两个部分。首先，目前的 DBMS 有大量“调节按钮”，这允许专家从可操作的系统上获得最佳的性能。通常，生产商要花费巨大的代价来完成这些调优。对数据库管理员，最重要的工作就是在计算机硬件平台配置一定的基础上，通过调整数据库系统的物理设计，取得最优的性能。此过程中存在许多数据库设计特征，比如各种类型的索引，物化视图以及水平、垂直分区等。这些特征对数据库中的实体都是可以选择的。因此数据库管理员往往在做出选择时处于两难的境地。事实上，大多数的数据库管理员在做这样的调整时，并不非常了解这些调整的意义。只是他们以前看过很多系统的配置和工作情况，将那些使系统达到最优的调整参数记录在一张表格中。当处于新的环境时，他们在表格中找到最接近眼前配置的参数，并使用那些设置。

这就是所谓的数据库调优技术。它其实给数据库系统的用户带来极大的负担和成本开销，而且 DBMS 的调优工作并不是仅依靠使用者的能力就能完成的。数据库管理员希望数据库系统足够智能地自主选择最佳的物理设计特征，从而取得最优的性能和减少拥有数据库系统的总拥有代价（Total Cost of Ownership，TCO）[Cha04]。

- **数据库自主管理的研究内容**

数据库系统自主管理指将原来 DBA 负责的工作交给数据库管理系统本身去完成，使之能在不需要用户干涉的情况下良好地运行。数据库系统自主管理大体上可以分为如下四类：

- 自我配置（Self-Configuration）：能适应环境的动态变化。
- 自我修复（Self-Healing）：能发现、诊断自身问题，并能对问题做出反

应。

- 自我调优(Self-Optimization):能调整资源和使用情况及进行负载均衡，使得信息资源得到最充分的利用。
- 自我保护（Self-Protecting):能预测、检测、识别出存在的攻击，并保护自身免受攻击。

上述四个方面，称为 Self-CHOP [CDL04]。Surajit Chaudhuri 还将自主管理分为五个层次：Basic、Managed、Predictive、Adaptive 和 Autonomic，分别对应 Level1 至 Level 5，级别越高，自动化程度越强。

其实，把基于规则的系统和可调控的数据库联系起来是可以实现数据库自动调优的。早在上个世纪八十年代，人们就认识到了需要使用工具软件辅助数据库系统进行性能调优。那时候专家系统被用来为系统调优提供决策支持[Dom87; Hel85]。

Hellerstein J L.提出了一个完整的 ATS（Automated Tuning System）架构[Hel97]，它通过在目标系统上层构建反馈控制机制来实现系统的自调优。[Ben03]和[Mar04]提出一种数据库性能自我诊断技术，它们把影响数据库性能的各种资源，如缓冲区大小、页面刷出频率、等锁时间等，按照对数据库性能影响的程度，赋予不同的优先级并组织在一棵诊断树中。当监测到性能问题时，就沿诊断树进行搜索，找到相应的问题资源，并通过调整它的值来解决。同样，此技术也是在数据库系统外围实现的，它利用 DB2 数据库提供的存储过程、触发器等在 DB2 平台上实现了一个自诊断工具。这些技术由于针对性较强，不利于系统的移植，如从一个 DBMS 上迁移到另一个 DBMS 上时，相应的自调优工具可能就需要重新开发。

真正认识到由数据库服务器自身来实现自我管理的重要性，是近十年来的事情。数据库研究者及数据库服务器供应商们都做了积极的努力。

研究报告[Ber98]中明确提出了“Plug and Play”数据库管理系统的概念，其第一个含义就是指数据库系统的 Self-Tuning，即如何使数据库系统中不再需要有数据库系统管理员，数据库系统中不再存在各种可以人为调节的参数，它本身可以适应环境的变化。报告中指出，要实现这一目标，需要做的第一步工作就是将数据库中各种性能调节参数去掉；第二步工作是实现自动化的物理数据库设计、逻辑数据库设计和应用程序设计。

Gerhard Weikum 在[Wei02]中对 Self-Tuning 数据库做了较全面的论述。他通过 E-Service 环境下，Self-Tuning 技术的必要性及实现该技术面临的巨大挑战出发，介绍了 Comfort 项目中一些成功的 Self-Tuning 经验，提出了基于反馈控制实现 Self-Tuning 数据库的 OPR 循环过程：即不断地 Observation、Prediction 和 Reaction，并总结了自管理数据库技术的发展现状和未来方向。他指出目前一些对性能影响不大或难以理解的参数已经从数据库系统中删除，而更多的可调参数已经被赋予合适的默认值。在磁盘存储级别的调优问题已经通过诸如采用分片存储、增加硬件等方法基本解决。物理数据库设计也基本实现自动化，这也是近几年被广泛研究并解决的一个问题。

经过学术界和工业界的努力，数据库自调优技术取得了巨大的进步。目前，广大的用户其实已经在数据库调优方面积累了大量的经验，诸如：动态资源分配、物理结构选择以及某种程度上的视图实例化等。

我们认为，数据库系统的最终目标是“没有可调部分” [孟小峰 04]，即所有的调整均由 DBMS 自动完成。它可以依据缺省的规则，如响应时间和吞吐率的相对重要性做出选择；也可以依据用户的需要制定规则。因此，建立能够清楚地描述用户行为和工作负载的更完善的模型是这一领域取得进展的先决条件。除了不需要手工调整，DBMS 还需要一种能力以发现系统组件内部及组件之间的故障，辨别数据冲突，侦查应用失败，并且做出相应的处理。这些能力要求 DBMS 具有更强的适应性。

1.3 本文的组织

本文提出一个通用的物理数据库自动设计(General Physical Database Schema Recommender，GPDSR)框架。并基于神舟 OSCAR 数据库系统实现该框架的一个原型。

第一章是绪论，介绍了一下 DB 和 DBMS 的概念、本研究的背景以及必要性。

第二章介绍了物理数据库设计的一些基本概念，并在此基础上讨论了目前部分主流商业关系数据库管理系统在物理数据库自动设计方面的相关研究。

第三章介绍了通用的物理数据库自动设计(General Physical Database

Schema Recommender，GPDSR)框架。

第四章介绍了我们在神舟 OSCAR 关系数据库系统中实现了的索引推荐器的原型系统，同时使用标准的数据库评测基准对系统进行了测试。

最后在第五章，总结了本文所论述的工作以及主要贡献，并对今后的发展方向做了一定的介绍和建议。

第二章 相关工作

在这一章里，首先介绍一下数据库物理设计几种常用手段的基本概念，然后是当前几个主流数据库在数据库自动管理和调优方面的最新进展，并分析它们的特点。

2.1 数据库设计

数据库设计是建立数据库及其应用系统的技术，是信息系统开发和建设中的核心技术。数据库设计是建立数据库及其应用系统的核心和基础，它要求对于指定的应用环境，构造出较优的数据库模式，建立起数据库应用系统，并使系统能有效地存储数据，满足用户的各种应用需求（信息要求和处理要求）。一般按照规范化的设计方法，常将数据库设计分为若干阶段：

系统规划阶段主要是确定系统的名称、范围；确定系统开发的目标功能和性能；确定系统所需的资源；估计系统开发的成本；确定系统实施计划及进度；分析估算系统可能达到的效益；确定系统设计的原则和技术路线等。对分布式数据库系统，还应分析用户环境及网络条件，以选择和建立系统的网络结构。

需求分析阶段要在用户调查的基础上，通过分析，逐步明确用户对系统的需求，包括数据需求和围绕这些数据的业务处理需求。通过对组织、部门、企业等进行详细调查，在了解现行系统的概况、确定新系统功能的过程中，收集支持系统目标的基础数据及其处理方法。

概念设计阶段要产生反映企业各组织信息需求的数据库概念结构，即概念模型。概念模型必须具备丰富的语义表达能力、易于交流和理解、易于变动、易于向各种数据模型转换、易于从概念模型导出与 DBMS 有关的逻辑模型等特点。

逻辑设计阶段除了要把 E－R 图的实体和联系类型，转换成选定的 DBMS 支持的数据类型，还要设计子模式并对模式进行评价，最后为了使模式适应信息的不同表示，需要优化模式。

物理设计阶段的主要任务是对数据库中数据在物理设备上的存放结构和存取方法进行设计。数据库物理结构依赖于给定的计算机系统，而且与具体选用的 DBMS 密切相关。物理设计常常包括某些操作约束，如响应时间与存储要求等。

系统实施阶段主要分为建立实际的数据库结构；装入试验数据对应用程序进行测试；装入实际数据建立实际数据库三个步骤。

另外，在数据库的设计过程中还包括一些其他设计，如数据库的安全性、完整性、一致性和可恢复性等方面的设计，不过，这些设计总是以牺牲效率为代价

的，设计人员的任务就是要在效率和尽可能多的功能之间进行合理的权衡。

2.2 物理设计

数据库是存储在物理设备上的。逻辑数据库设计工作完成后，需要为逻辑数据模型选择适合应用环境的物理结构，即存储结构与存取方法。数据库物理设计就是为数据库文件确定存储结构和存取路径，以使各种数据库应用均能获得最佳性能的过程。由于物理结构依赖于给定的 DBMS 和硬件系统，因此设计人员必须了解所用的 DBMS 的内部特征，特别是存储结构和存取方法；了解应用环境，特别是应用的处理频率和响应时间要求；以及了解外存设备特性。数据库物理设计的任务是对给定的逻辑数据模型选取适合应用环境的物理结构，即在逻辑设计的基础上，为每个关系模式选择合适的存储结构和存取方法，使数据库的事务能够高效率地运行。许多关系数据库大量地屏蔽了内部物理结构，留给用户参与设计的余地不多。一般的 RDBMS 留给用户参与物理设计的内容大致是索引、分区等特征的设计。

物理数据库设计阶段的设计过程主要包括以下三方面工作：

(1)分析影响物理数据库设计的因素；

(2)为关系模式选择存取方法；

(3)设计关系、索引等数据库文件的物理存储结构。

2.2.1 索引（index）

数据库系统中文件的索引的工作方式非常类似于书的索引，是一种辅助存取结构，用于提高响应确定查询条件的记录的检索速度。典型的索引结构提供辅助存取路径，这些路径为记录的存取提供了不同的方法，但并不影响记录在磁盘上的物理结构。采用所有技术使得存取基于索引字段的记录变得更加有效（索引字段是用于构建索引的字段）。

最流行的索引类型是基于有序文件（单级索引）和树状数据结构（多级索引，B^+树）。索引也可以基于散列或者其他搜索数据结构来构建[Abr02]。

- **主索引（primary index）**

主索引是一个有序文件，它的每个记录是包含两个字段的定长记录。第一个字段与数据文件的排序码（也叫主码）有相同的数据类型；第二个字段是指向一个磁盘块的指针（块地址）。对于数据文件中的每一个块，在索引文件中都对应一个索引入口（或者索引记录）。

➢ **聚簇索引（clustering index）**

如果文件的记录以一个非码字段（在该字段上每个记录取值不唯一）进行排序，那么这个字段被称为聚簇字段。检索这些在聚簇字段上拥有相同值的记录时，为了加快检索速度，可以创建另一种不同类型的索引——聚簇索引。这和主索引是有区别的，后者要求数据文件的每个记录在排序字段上取值唯一。

聚簇索引也是一个每个记录包含两个字段的有序文件。为了减少插入记录带来的问题，常见的方法是对聚簇字段的每一个值预留一个整块（或者一个连续的块簇），所有取该值的记录都放在相应的一块（或者块簇）中。这样就使得插入与删除操作相对简单。

➢ **辅助索引（secondary index）**

辅助索引可以指定在文件的任何非排序字段上，同样是一个每个记录包含两个字段的有序文件。第一个字段和数据文件中的索引字段有相同的数据类型，这个索引字段是文件中的某个非排序字段；第二个字段是一个块指针或者一个记录指针。对于同一个文件可以有多个辅助索引，从而有多个索引字段。

➢ **多级索引（multilevel index）**

多级索引的思想是以因子 bfr_i 来减少需要继续搜索的索引文件的长度。bfr_i 是索引文件的块因子，它的值大于 2。于是搜索空间快速减少。bfr_i 的值被称为多级索引的扇出，记做符号 fo。搜索多级索引需要大约 $\log_{fo}b_i$ 次块访问，如果扇出大于 2，则 $\log_{fo}b_i$ 小于二分查找所需要的块访问次数。

B.和 B^+-树是众所周知的树状数据结构的两个特例。

2.2.2 物化视图（materialized view）

物化视图是包括一个查询结果的数据库对像，它是远程数据的的本地副本，或者用来生成基于数据表求和的汇总表。物化视图存储基于远程表的数据，也可以称为快照。

物化视图可以查询表，视图和其它的物化视图。

通常情况下，物化视图被称为主表（在复制期间）或明细表（在数据仓库中）。

对于复制，物化视图允许你在本地维护远程数据的副本，这些副本是只读的。如果你想修改本地副本，必须用高级复制的功能。当你想从一个表或视图中抽取数据时，你可以从物化视图中抽取。

对于数据仓库，创建的物化视图通常情况下是聚合视图，单一表聚合视图和连接视图。

物化视图提供了可伸缩的基于主键或 ROWID 的视图，并指定刷新方法和自动刷新的时间。

2.2.3 水平分区（horizontal partitioning）

将数据库分区，不仅可提高其性能，而且也易于对其维护。通过把一个大表拆分成更小的单个表，可使得只需访问一小部分数据的查询执行得更快（这是因为需要扫描的数据较少），同时还可以更快地执行维护任务（如重建索引或备份表）。实现分区操作时可以不拆分表，只需将表物理地放置在别的磁盘驱动器上即可。例如，将表放在某个物理驱动器上，再将相关的表放在与之分离的驱动器上，这样就可提高查询性能。因为当执行涉及表之间联接的查询时，多个磁头可同时读取数据。

水平分区是将一个表分段为多个表，每个表包含相同数目的列和较少的行。例如，可以将一个包含十亿行的表水平分区成 12 个表，每个小表代表特定年份内一个月的数据。任何需要特定月份数据的查询只引用相应月份的表。

将表进行分区是为了使查询引用尽可能少的表。否则，查询时须使用过多的 UNION 查询来逻辑合并表，而这会削弱查询性能。具体如何将表进行水平分区取决于如何分析数据。常用的方法是根据时段或实际使用对数据进行水平分区。例如，一个表可能包含最近五年的数据，但是只定期访问本年度的数据。在这种情况下，可考虑将数据分区成五个表，每个表只包含一年的数据。

2.2.4 垂直分区（vertical partitioning）

垂直分区将一个表分段为多个表，每个表包含较少的列。垂直分区的两种方法是规范化和行拆分。

规范化是个标准数据库进程，该进程可从表中删除冗余列并将其放到次表中，次表再按主键与外键的关系链接到主表。

行拆分将原始表垂直分成多个只包含较少列的表。拆分的表内的每个逻辑行与其它表内的相同逻辑行匹配。例如，链接每个拆分表内的第十行将重新创建原始行。

与水平分区一样，垂直分区使查询得以扫描较少的数据，因此可以提高查询性能。例如有一个包含七列的表，通常只引用该表的前四列，那么只要将该表的后三列拆分到一个单独的表中，就可获得性能收益。

应谨慎考虑垂直分区操作，因为分析多个分区内的数据需要有链接表的查询，如果分区非常多，将可能影响性能。

2.3 自动物理设计

物理设计的目的不仅要提供存储中数据的合适结构，而且要以合适的方式对性能提供保证。对于某个给定的概念模式来说，在给定的 DBMS 中存在着多种物理设计方案。直到明确了将要在数据库上运行的查询、事务和应用之后，才可能制定出有意义的物理设计方案和性能分析。

大部分 DBMS 都提供了相应的命令或追踪工具，借助这些工具，DBA 可从系统获得关于查询执行过程的信息——以何种顺序执行何种操作、使用何种辅助存取结构等。通过分析这些执行计划，可以得到产生性能问题的原因，然后进行调整（如删除或添加一条索引等）。

目前市场上主流的商业数据库管理系统都在不断推出新技术，尽可能地使 DBMS 进行自我管理，以实现最小的人力管理，朝着建立一套"自我察觉"、"自我学习"和"完全自我管理"的数据库目标前进。

2.3.1 Sql Server 中的相关内容

数据库引擎优化顾问（DTA）是 Microsoft SQL Server 2005 中的新工具，使用该工具可以优化数据库，提高查询处理的性能。数据库引擎优化顾问检查指定数据库中处理查询的方式，然后建议如何通过修改物理设计结构（例如索引、物化视图和分区）来改善查询处理性能。

它取代了 Microsoft SQL Server 2000 中的索引优化向导，并提供了许多新增功能。例如，数据库引擎优化顾问提供两个用户界面：图形用户界面 (GUI) 和 dta 命令提示实用工具。使用 GUI 可以方便快捷地查看优化会话结果，而使用 dta 实用工具则可以轻松地将数据库引擎优化顾问功能并入脚本中，从而实现自动优化。此外，数据库引擎优化顾问可以接受 XML 输入，该输入可对优化过程进行更多控制。

图 2.1 表示了 DTA 的体系结构[Agr04]。

数据库引擎优化顾问提供下列新优化功能：

- 时间限制优化。可以控制数据库引擎优化顾问用于分析工作负荷的时间。建议的质量随时间增加而提高。
- 跨多个数据库进行优化。可以优化涉及多个数据库的工作负荷。数据库引擎优化顾问可以对工作负荷中的任何数据库提供索引、物化视图或分区的建议。
- 优化更广泛的事件和触发器类。可以包括带有下列事件类的工作负荷：
  - 用户定义函数 (UDF)

- ✧ 引用临时表的批
- ✧ 触发器中的语句

➢ 优化日志。数据库引擎优化顾问将所有无法优化的事件写入优化日志，并提供消息说明事件无法优化的原因。可以在优化会话过程中查看日志，以确定

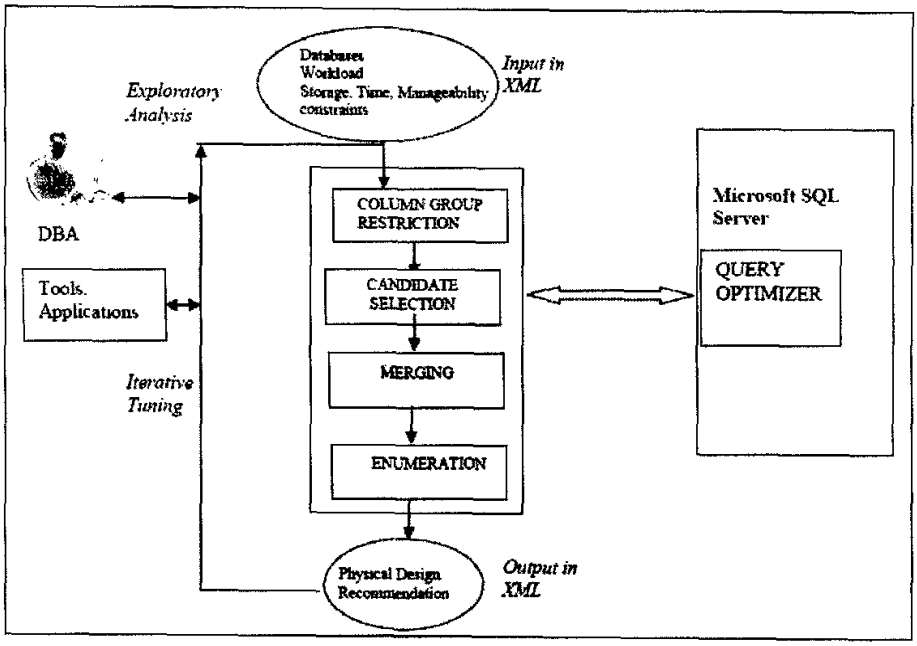

图2.1 Overview of Database Tuning Advisor

数据库引擎优化顾问是否能够优化工作负荷中的事件。

➢ 假设分析。数据库引擎优化顾问的用户指定配置功能支持假设分析。用户指定功能使用户可指定 XML 输入文件中现有和假设的物理设计结构的配置。然后可以使用数据库引擎优化顾问来评估这些物理设计结构的效果，不会在开始优化前就引起实现该配置的开销。

➢ 对优化选项的更多控制。数据库引擎优化顾问还使用户可指定更多优化选项。例如，可以指定数据库引擎优化顾问在生成建议时考虑是要添加非聚集索引还是要保留全部现有的聚集索引。

➢ XML 支持。数据库引擎优化顾问可以将 XML 文件作为输入，或者生成 XML 格式的建议。

➢ 分区建议。数据库引擎优化顾问还可适时建议分区，以改进大型表的性能和可管理性。

➢ 支持使用测试服务器减轻生产服务器的优化负载。数据库引擎优化顾问通过将多数优化负载卸载到测试服务器来优化生产服务器上的数据库。它通过使

用生产服务器硬件配置信息，而不是真正地将数据从生产服务器复制到测试服务器，来执行该操作。数据库引擎优化顾问不会将实际数据从生产服务器复制到测试服务器中。而只是复制元数据和必要的统计信息。

- db_owner 固定数据库角色的成员可以优化它们的数据库。除了 sysadmin 固定服务器角色的成员，db_owners 固定数据库角色的成员也可以用数据库引擎优化顾问优化它们的数据库。

2.3.2 Oracle 中的相关内容

Oracle 在以前的版本里对性能调优方面的工具是 Statspack。该工具是获取数据库性能统计数据的快照，并在这些快照的基础上生成报表。Statspack 的报表节省了 DBA 做性能调整方案的决策时间，但它还是把整理分析报表数据以及鉴定产生问题的根源的工作留给了 DBA。Oracle 9i 在性能调优方面还有其它一些特征，如：运行时查询动态优化、内存管理自动配置等特征。

Oracle 数据库在最新版 10g 引入了一套高级的自我管理数据库，它可以自动地对自身进行监控、调整。Oracle 数据库 10g 的自我管理基础架构包括四大组件：自动工作负载仓库、自动维护任务基础架构、服务器生成告警和顾问框架。下面我们介绍一下自动工作负载仓库（AWR）和自动数据库诊断监控程序(ADDM)[Kar05] [Ora05] [Dag04]。

- **自动工作负载仓库（AWR）**

自动工作负载仓库（AWR），顾名思义，就是每一个 Oracle 数据库 10g 的内置信息库，其包括特定数据库和其他类似信息的运行统计数据。在常规的时间间隔中，Oracle 数据库 10g.对其所有的关键数据和工作负载信息进行了快照，并将它们储存在 AWR 中。

AWR 构成了 Oracle 数据库 10g 中所有自我管理功能的基础。它相当于一个信息源，向 Oracle 数据库 10g 提供了一个透视历史的角度，对其如何被使用、以及如何针对系统的运行环境作出精准而适宜的决策进行了翔实描述。

- **自动数据库诊断监测(ADDM)**

构建于 AWR 捕捉的数据之上，Oracle 数据库 10g 包括一项自动诊断功能，名为“自动数据库诊断监测”(ADDM)。ADDM 使 Oracle 数据库 10g 可以诊断自身的性能并确定对发现的问题如何进行解决。ADDM 在每一 AWR 数据捕捉后自动运行，并对该数据进行性能检测。图 2.2 表示了 ADDM 的体系结构[Cha04]。

ADDM 检查 AWR 中所捕捉的数据，前瞻性地分析并确定系统面临的主

要问题，同时在多数情况下提供建议性的解决方案，并量化预期获得的利益。

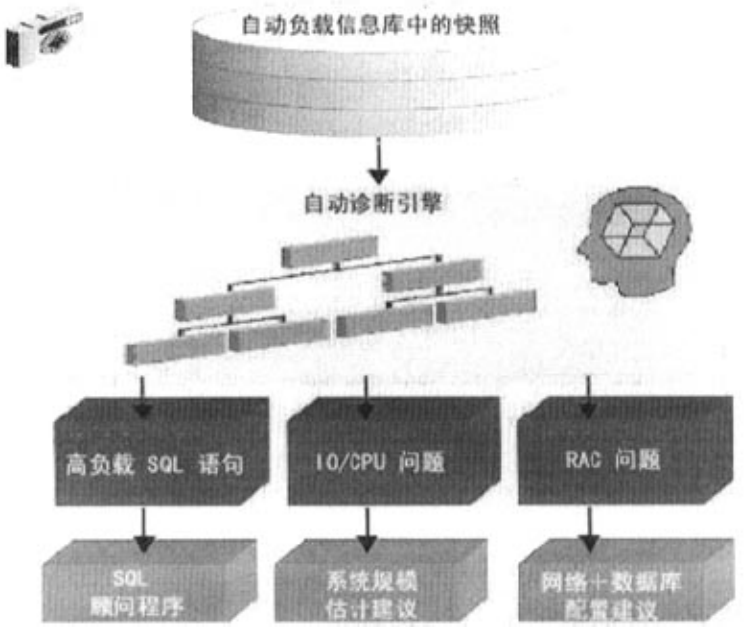

图2.2：ADDM体系结构

ADDM 通过对数据库中耗费时间最多的活动进行关注和分析，并通过一项高级的问题分类树进行详细研究。

- **Oracle 数据库 10g 诊断优点**

内置于 Oracle 数据库 10g 中的自动数据库诊断监控程序 (ADDM) 具有以下优点：

1. 每 60 分钟生成一份自动性能诊断报告
2. 问题诊断基于几十年的调整经验
3. 对问题影响进行基于时间的量化并提出建议
4. 识别根本原因，而非故障现象
5. 由于全部数据保存在自动工作负载信息库 (AWR) 中，因此显著降低了重放工作负载进行详细分析的需要。

表 2.1[Ora05] 描述了 *Oracle* 数据库 *10g* 使用前和使用后性能诊断和解决方法之比较。

Oracle 数据库 10g 之前	Oracle 数据库 10g
1. 检查系统利用率 2. 查看等待事件 3. 观察锁存器争用 4. 查看共享池上和库中高速缓冲锁存器的等 待 5. 检查 v$sysstat (difficult) 6. 查看"parse time elapsed" > "parse time cpu" and #hard parses greater than normal 6. 识别 SQL，方法是 □识别许多较难语法分析的会话，并进行追踪，或是 □用同样的 Harsh 方案（困难）检查许多语句的 v$sql 7. 分析访问的目标并检查 SQL 8. 通过观察 SQL 包含的字面含义，识别"较难的句法分析"问题	1. 检查 ADDM 建议 2.ADDM 建议使用游标共享

表 2.1：性能诊断和解决方法 Oracle 数据库 10g 使用前和使用后之比较

2.3.3 DB2 中的相关内容

IBM 公司的 DB2 在物理数据库自动设计方面也有相应的工具。该公司有一个叫"DB2 自治计算（DB2 Autonomic Computing Project）"的工程，于 1998 年推出了"索引顾问（Index Advisor）"原型。该工程先前叫做"Self-Managing And Resource Tuning (SMART) "，于 2000 年初正式启动。

现有的 DB2 自治特征（Features）有以下三种[Cha04]：

- 索引顾问（Index Advisor）
- 配置顾问（Configuration Advisor）
- 健康顾问（Health Advisor）

最新的 DB2 版本又推出了设计顾问（Design Advisor）[Dan04] [Zil04]。它是在索引顾问的基础上扩充而来的，在索引的基础上又增加了物化视图、表分区技术（水平分区或垂直分区）和多维群集存储方案的自动推荐。并实现了自动统计信息采集（Automated Statistics Collection）。

2.4 小结

当前所有对物理数据库自动设计的研究都针对特定的数据库系统，因为不同的数据库系统支持不同类型的物理设计特征。移植一个数据库系统中的物理数据库自动设计框架到另外一个数据库系统是非常困难的事情。而且目前的体系架构非常不灵活，加入新的物理数据库设计方法将使得整个架构无效。

本文中使用基于推理的技术来解决上述问题。该解决方案包括一个数据库物理模式推荐框架(General Physical Database Schema Recommender， GPDSR)和一个用于产生物理设计特征的描述逻辑集合。需要修改原有物理特征或者添加新物理特征时，物理数据库自动设计框架本身不需要改变，仅需修改或者扩展逻辑描述集合。

第三章 数据库物理模式推荐框架

索引、水平分区、垂直分区、物化视图、多维集群表[Pad03]等物理数据库特征能显著提高数据库的查询性能。但是，插入和修改记录需要维护索引；物化视图需要冗余的存储空间，并且在记录被修改时还需要同步（或者是周期性地同步）；等等。这些都需要一定的开销。 数据库物理模式推荐框架（GPDSR）就是为了最大限度地提高数据库系统的性能，并有效地控制由此带来的开销，得到合理的“性价比”。

数据库自动物理设计需求：

- 能对物理设计特征进行综合的推荐。
- 将部分管理特征融入到物理设计
- 具有良好的可伸缩性，能处理超大规模的数据库和数据量
- 能用很小的开销调优工作数据库
- 具有较好的客户定制能力；客户可以根据自己的需求进行定制或配置。

3.1. 框架概述

> **问题描述：**给定一个工作负载 W（一组 SQL 语句），磁盘空间约束 D，时间限制 T，在使用磁盘空间不超过 D 的情况下，求出一组物理设计特征（索引、物化视图等）建议，使得数据库系统在执行 W 时能获得最佳性能。

图 3.1 是数据库物理模式推荐框架，分为两个阶段：工作负载分析阶段和设计推荐阶段。

输入：

- 多个数据库的运行信息。很多应用同时访问多个数据库，能同时对多个数据库提出优化建议是很重要的。
- 工作负载（Workload）。工作负载是在数据库服务器上运行的 SQL 语句集合，可以从数据库管理系统的 SQL 语句解析（parse）模块得到。
- 需要调整的特征（Feature）集。可以优化的数据库特征有索引、分区、物化视图等，但有时 DBA 可能只想调整其中一个子集，如：某个系统不想调整物化视图，那么就必须把这个信息输入，使得推荐器能从剩下的特征里推荐出最优解。

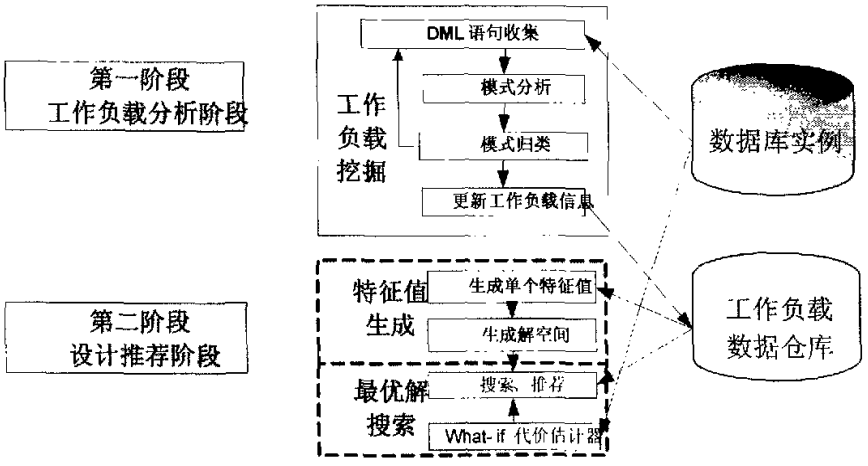

图3.1： 数据库物理模式推荐框架

输出：

- 物理数据库设计建议，包括：索引、水平分区、垂直分区、物化视图等。

除了以上输入信息以外，还需要提前自动采集到以下信息：

- 数据库的特征信息，包括模式、统计信息（如表或列的数量）以及其它已有的一些物理数据库设计特征，如索引、视图、分区等信息。
- 系统配置信息，包括处理器的估计处理能力、磁盘的技术参数、处理机结点的数量以及网络的带宽（在 shared-nothing 多处理机环境中）等。

工作负载分析阶段(阶段 1)获取负载和分析当前运行数据库实例的工作负载。此阶段中执行的 SQL 语句根据所属模式被记录和分类，产生一个模式类别的列表，其中每个模式类别包含对应的模板 SQL 语句以及其出现频率。该列表被存储于工作负载仓库中。该仓库既可被存储于当前数据库实例，也可被存储于其他机器上独立运行的数据库实例中。

设计推荐阶段(阶段 2)，根据当前工作负载推荐一个可取得最优性能的物理设计方案。此阶段中首先装载工作负载分析中取得的模式类别及其频率信息；接着生成可能提高数据库系统性能的所有物理设计的解空间；最后基于运行实例本身的 what-if 代价估计器，在解空间中搜索出使加权 (频率)总代价最小的物理设计特征。

下面基于图 3.2 中描述的工作负载，讨论运行于以上两阶段的三个相互协作组件的细节。

3.2. 工作负载分析组件

影响物理设计工具可伸缩性的一个主要因素是工作负载的规模，也就是工作负载中的语句数量。为了保证推荐的准确性，工作负载的量可能很大，那么分析工作负载的某个子集是否也能获得和分析整个工作负载差不多的效果呢？

两个采样策略可以证明采用工作负载的子集作为样本具有重大缺陷。

- 采用平均地、随机地从工作负载中采样的方法会漏掉重要的查询信息，因此可能会使用比必需的工作负载规模还要大的样本才能达到满意的物理设计效果。
- 另一个策略是采用代价最大的前 N 个工作负载作为样本，那么至少样本所花费的代价占所有工作负载总代价的百分比（预先定义好的）是达到了，由此却带来了另一个问题。工作负载中的语句有的可能是偶尔出现的（如：调用一个存储过程），这样，有可能属于一个模式的所有语句的代价均高于其它模式，最后可能大量使用了属于该模式的样本，而没有使用别的模式的样本，最后的设计效果也就可想而知了。

工作负载压缩技术

工作负载压缩技术是利用工作负载的内在模式，在每条语句“签名”的基础上将其分类，也就是说，两条 SQL 语句中，如果除了引用的常量不同以外，其它特征都一样，那么我们认为这两条语句具有相同的“签名”。该技术使用基于分组的方法从每组选取一定数量的子集。我们将该方法运用到 GPDSR 中，可以显著地减少自动设计的时间，而不会明显影响物理设计特征的推荐质量。

工作负载分析组件（Workload Analysis Component， WAC）是工作负载分析阶段的唯一组件。它周期性地收集工作负载信息，同时，也可在数据库系统遇到性能问题时被激活收集信息。WAC 被激活后，所有数据库实例收到的 SQL 语句将被收集并分析。该组件既可作为一个外部 SQL 插件的形式存在，也可作为一个内嵌子模块嵌入在数据库系统的 SQL 解析模块中。该组件分析四种数据操作(Data Manipulation Language， DML) SQL 语句：select、insert、update 和 delete。与数据库系统中的 SQL 语言解析器不同，WAC 关注于识别 SQL 语句的模式，而非理解 SQL 语句本身的意义。

如图 3.2，查询语句 SQL1 的模式特征被识别为: (OP:S(student)，TAR:(id，name)， WH:(name)，SORT:(id))；而更新语句 SQL2 被识别为: (OP:U(student)，TAR:(name)， WH:(id))。具有相同模式的 SQL 语句将被归类于同一模式类别中，如:SQL3 被认为与 SQL1 具有相同模式，因为其差别仅在于一些不影响查询执行的顺序调换和常量替换。而 SQL4 与 SQL1、SQL3 则不属于同一个模式类别，它的模式被识别为: (OP:S(student)，TAR:(*)，WH: (name，age))。

数据库模式:
CREATE TABLE student (id INT, name CHAR(16), age INT, addr VARCHAR(255))
语句:
SQL1: SELECT id, name FROM student WHERE name = 'Alice' SORT BY id; 频率: 20%
SQL2: UPDATE user SET name = 'Bob' WHERE id = 1; 频率: 10%
SQL3: SELECT name, id FROM student a WHERE 'Bob' = name SORT BY id; 频率: 20%
SQL4: SELECT * FROM student WHERE name = 'Alice' and age = 20; 频率: 25%
SQL5: SELECT id FROM student WHERE id = 1 and name = 'Alice'; 频率: 15%
SQL6: SELECT id FROM student SORT BY id, name; 频率: 10%

图3.2 工作负载样本

以上例子中的 SQL 语句很简单，其模式可被表示为简单的字符串形式。然而，在现实例子中，SQL 语句通常包含子查询、各种连接以及各类聚集函数。这些复杂的 SQL 语句将被分析，其模式将被以树的形式进行存储。收集一段时间后，依据其模式，SQL 语句被分到不同的模式类别中。每种模式类别信息中包括：模式类别树、模板 SQL 语句以及该模式类别所属 SQL 语句的总出现频率。这些信息将被记录于工作负载仓库中。

工作负载模式的存储技术

- 一般地，有一种简单的存储工作复杂模式的方法：
  在一个列表里存储所有的模式...
  从头到尾比较每一个模式...
  结果因为有太多的模式而导致处理一条工作负载的速度非常慢。
- 有一种可能更好的方法：
  在一个哈希（HASH）表里存储模式，直接用 HASH 函数映射模式（具有相同的 HASH 值映射到同一模式）。
  在 HASH 表里搜索模式...

3.3. 特征空间构造组件

特征空间构造组件（Feature-space Construction Component， FCC）是设计推荐阶段的两个组件之一。阶段 1 收集和分析工作负载信息后，FCC 模块被激活产生可能提高系统性能的物理设计特征。FCC 分析存储在工作负载仓库中的模式类别信息，而后产生一系列当前数据库系统支持的物理数据库设计特征。每种模式类别信息被处理后，所有的物理数据库设计特征被最终检查，去掉重复的设计特征，产生以后进一步推荐的备选设计特征集合。

特征生成步骤：

1. 为每一类模式（pattern）生成特征，并将它们组合；
2. 删除重复的特征；

3. 删除目前已经存在的特征；
4. 删除其它一些特征（指已经包含在别的特征里的特征）。

特征举例：

- 如图 3.2 中的 SQL1 模式类别和 SQL3 的模式类别使 FCC 产生以下设计特征: IDX(student.name)，IDX(student.id)，VP(student.id，student.name)。而 SQL4 产生的设计特征为：IDX(student.name，student.age)，VP(student.name，student.age)。这里用 IDX 表示索引，用 VP 表示垂直分区。
- 又如在查询中，所有形如“SELECT-FROM-WHERE-GROUP BY”形式的组合可以产生设计特征 MV(物化视图)。
- 所有使用 VIEW（逻辑视图）的查询都将产生设计特征 MV。因为用户可能经常需要查询这些要素，为了简便而建立了 VIEW，推荐 MV 可能是比较好的候选特征。

然而，对不同数据库系统，它们的设计特征的特性不同。要使被一种数据库系统支持的设计特征能够被其他的数据库系统使用，只有为新数据库系统重新设计 FCC。更好的方法是提供一个框架去完成 FCC 的主要工作，然后为每个数据库系统提供一个基于它的物理设计特征的知识库。这样 FCC 就可作为一个推理组件，它依据提供的物理设计特征知识库来对设计特征进行推理。

我们使用描述逻辑（DL）[Baa03] 来构建特征生成知识库。

3.3.1 T-Box 和 A-Box

T-Box 形式化地描述特征生成知识库的基本概念和关系。A-Box 将在模式实例推理中被使用。

定义 1。 T-Box (特征生成知识库的基本概念和关系)

- $Tab \subseteq DBEnt$ ：数据库中的表
- $Cols \subseteq DBEnt$ ：数据库中的属性列表
- $Idx \subseteq DBFeat$ ：数据库中可能的索引，一种物理设计特征
- $Ptn = PtnSel \cup PtnUpd \cup PtnIns \cup PtnDel$ ：可能的工作负载模式，包括四种 DML 模式
- hasCols(x， y)：Cols x 包含 Cols y （不考虑顺序）
- incCols(x， y)：Cols x 包含 Cols y （考虑顺序）
- tabCols(x， y)：Tab x 有 Cols y
- idxTab(x， y)， idxCols(x， z)：Idx x 在 Tab y， Cols z 上
- ptnTab(x， y)：Tab y 上的 Ptn x

- ptnTarget(x， y) ： Ptn x has target list Cols y
- ptnWhere(x， y) ： Ptn x 在 Cols y 上有 where 子句
- ptnSort(x， y) ： Ptn x 在 Cols y 有 sort 子句
- genFeat(x， y) ： Ptn x 产生特征 DBFeat y

定义 2。 **A-Box (OSCAR 数据库索引特征生成规则)**

- 规则 **1. Where 子句索引生成规则** ：

$\forall$ c:Cols， <Ptn $\cap$ $\exists$ ptnTab.(tabCols.c) $\cap$ $\exists$ ptnWhere.(c)， Idx $\cap$ $\forall$ idxTab.(tabCols.c) $\cap$ $\forall$ idxCols.(hasCols.c) > : ***genFeat***;

- 规则 **2. Sort-by 子句索引生成规则** ：

$\forall$ c:Cols， <PtnSel $\cap$ $\exists$ ptnTab.(tabCols.c) $\cap$ $\exists$ ptnSort.(c)， Idx $\cap$ $\forall$ idxTab.(tabCols.c) $\cap$ $\forall$ idxCols.(incCols.c) > : ***genFeat***;

值得注意的是：A-Box 中的两条规则不同，因为：

- 规则 1 是针对 select/update/delete 的 where 子句；另一个是针对 select 中的 sort-by 子句。

- 根据 SQL 语句的 where 子句生成索引的时候，索引中属性的顺序是无关的；然而根据 SQL 语句的 sort-by 子句生成索引的时候，索引中属性的顺序有关。比如，图 3.2 中的 SQL5，可以产生两个索引：

  CREATE INDEX I_1 ON student (id， name);

  CREATE INDEX I_2 ON student (name， id);

它们对优化器选择执行路径的影响效果一样。但是对 SQL6，*I_1* 减少搜索的代价，而 *I_2* 没有用。

3.3.2 特征生成推理

要提供特征生成知识库，需要推理算法的步骤如下：

算法 1：特征生成推理算法：

- 函数：产生一个特征集合，这些特性有可能提高数据库系统性能
- 输入： 模式的列表
- 输出：一个特征集合
- 说明：N 表示模式的个数，pt_i 是第 i 个模式（$1 \le i \le N$）， F 是特征集合的计算结果

1. 开始执行
2. 设置 $F=\emptyset$;
3. 从 i=1 到 N 解析 pt_i 并且根据 T-Box 的定义，表示为 ptn_i
4. 使用 A-Box 中的每个规则：*genFeat*（*Ptn*（*c*），*DBFeat*（*c*））：
   a) 使用推理工具，查找集合 F' = { c | ptn_i: *satisfies Ptn(c)*，*c:COLS* };
   b) 合并 F' 到 F； 继续下一个 *genFeat* 。
5. 返回 F ，作为产生的特征集合
6. 执行结束

3.3.3 特性生成举例

以图 3.2 中工作负载为例子。针对这六条 SQL 语句，WAC 产生五类模式，然后使用 T-Box 重写这些模式，获得结果如表 3.1。

ID	模式	频率
1	*PtnSel ∩ptnTab. student ∩ptnTarget.(id，name) ∩ptnWhere.(name) ∩ptnSort.(id)*	40%
2	*PtnUpd ∩ptnTab.student ∩ptnTarget.(name) ∩ ptnWhere.(id)*	10%
3	*PtnSel ∩ptnTab.student ∩ptnTarget.(*) ∩ptnWhere.(name， age)*	25%
4	*PtnSel ∩ptnTab.student ∩ptnTarget.(id) ∩ ptnWhere.(id，name)*	15%
5	*PtnSel ∩ptnTab.student ∩ptnTarget.(id) ∩ ptnSort.(id，name)*	10%

表3.1： 工作负载模式类别

通过推理算法处理，对类别 1，当 c 是（name）的时候，模式满足 A-Box 中的规则 1；当 c 是（id）的时候，模式满足 A-Box 中的规则 2。其他四个模式也进行如此处理，得到生成特征结果如表 3.2。

通过合并，删除重复，最终的特征集合将是：

Idx ∩idxTab.student ∩idxCols.(id) ，

Idx ∩idxTab.student ∩idxCols.(name) ，

Idx ∩idxTab.student ∩idxCols.(name，age) ，

Idx ∩idxTab.student ∩idxCols.(id ，name)。

表 3.2： 各种类别生成的特征

ID	生成的特征
1	*Idx ∩idxTab.student ∩idxCols.(id)* *Idx ∩idxTab.student ∩idxCols.(name)*
2	*Idx ∩idxTab.student ∩idxCols.(id)*
3	*Idx ∩idxTab.student ∩idxCols.(name， age)* *Idx ∩idxTab.student ∩idxCols.(age， name)*
4	*Idx ∩idxTab.student ∩idxCols.(id， name)* *Idx ∩idxTab.student ∩idxCols.(name， id)*
5	*Idx ∩idxTab.student ∩idxCols.(id， name)*

3.4. 最优特征搜寻组件

最优特征搜寻组件（Optimal-feature Search Component， OSC）作为在阶段 2 中的另一个组件，它在 FCC 生成特征集合（用 F 来标识）之后被激活。OSC 搜索 F 中一个子集，使该子集在满足一定硬件平台限制下取得最佳的整体性能。OSC 中两个重要问题就是代价评价的方法和搜索使用的搜索算法。

为评价不同的物理设计方案，OSC 用当前的数据库实例评价物理设计的代价。对 F 中的每一子集 F'，OSC 要求数据库实例虚拟应用 F'，在这里虚拟应用的意图是不建立真实的数据库实体，而仅让优化器意识到 F'中物理特征的存在。成功运用了虚拟特征后，OSC 使优化器重新评价样本 SQL 语句，为工作负载仓库中每种模式类别计算执行一次的代价。被估计的总代价就是每种模式类别执行一次的代价加权求和，而权重就是该模式类别在全部模式类别中出现的频率。

OSC 中最优化的搜索问题其实就是背包问题，此背包问题中背包的物体就是物理设计特征，选择物体的条件限制就是运行该数据库实例可用的物理空间。很多成熟的算法可以解决最优解空间搜索的背包问题。

3.4.1 代价估计

对 workload 进行代价估计：

C＝Cost（W，D），

其中 W 代表 workload，D 代表物理设计特征组合，C 代表采用 D 的情况下执行一次 W 所需要的总代价：

$W = a_0 \cdot Q_0 + a_1 \cdot Q_1 + \cdots + an \cdot Q_n$;

其中 Q_n 代表 workload 中的一种模式，a_n 代表模式为 Q_n 的工作负载出现的频

率（用百分比表示），$a_0 + a_1 + \cdots + an = 1$。

Cost(W, D) = a0 • Cost(Q0, D) + a1 • Cost(Q1, D) + … + an • Cost(Qn, D);

其中 Cost(Qn, D)代表采用特征组合 D 的情况下执行一次模式为 Qn 的工作负载所需要的代价。

Cost = Kc*Cc + Kd*Cd + Ks*Cs;

其中 Cc： cpu 的利用率，Cd：磁盘的访问（IO）次数， Cs：内存页面数 Kc， Kd， Ks 指相关系数。

Cost(Qn, D)由 CPU、磁盘、内存三方面的开销组成，实际的代价可由数据库引擎提供。

3.4.2 搜索算法

➢ 0 / 1 背包问题[郑宗汉 05]

背包问题是个典型的 N—P 问题，在实际的工程中有着广泛的应用。

具体描述是：已知尺寸大小分别为 W_1，W_2，…，W_n 的若干物品和容量为 C 的背包，物品的价值分别为 P_1，P_2，…，P_n 。此处，W_1，W_2 ，…，W_n 和 C 都为整数。要求找出这 n 个物品的一个子集，使其尽可能使选入背包的物品的价值最大，即

最大化：$\sum_{i=1}^{n} P_i X_i$ 且满足 $\sum_{i=1}^{n} W_i X_i \leqslant C$

这里 $X_i \in \{0, 1\}$，$1 \leqslant i \leqslant n$ ； $X_i = 1$ 表示物品 i 被选入背包，$X_i = 0$ 表示未选如入。比如当 =3 时，W =[2，4，6]，P=[3，10，12]，C=11，即背包中装入物品的最大价值为 22。

➢ 贪婪算法

解决 0/1 背包问题的贪婪算法有好几种策略，如价值贪婪准则、重量贪婪准则、价值密度（P_i/W_i）贪婪准则。这种选择准则为：从剩余物品中选择可装入包的 P_i/W_i 值最大的物品，这也是组件中所选择的贪婪策略，因为它是一个直觉上近似的解。

算法过程描述如下：

1）对于给定的物品，分别求出价值密度(价值重量比)

$r_i = P_i/W_i$，i：1，…，n。

2)忽略先前的物品排序，按照价值密度，进行非升序排列：

$P_1/W_1 \geqslant P_2/W_2 \geqslant \ldots \geqslant P_n/W_n$

3)重复以下步骤，直到不满足条件为止：对于当前的物品，如果重量小于包中剩余的容量，则放入，并置物品标志为 1(表明被选中)，否则就停止。

在我们的问题中，采用贪婪算法解决问题的步骤如下：

（1）开始时将每个特征作为一个特征组合，计算每个特征组合对 workload 的代价，选出最小代价的特征组合作为当前最小代价特征组合；

（2）将当前最小代价特征组合和每个剩下的特征组成新的特征组合，计算新的特征组合对 workload 的代价，选出最小代价的特征组合；

（3）如果此次的代价比上一次的最小代价小，那么将此次最小代价特征组合作为当前最小代价特征组合，然后返回步骤（2）开始新一轮的组合、计算代价；

（4）返回当前最小代价特征组合

我们处理的工作负载类型包括 select、update、insert、delete，当大量引入索引或物化视图时，查询的性能优化了，但 update 的性能却下降了，因为它额外增加了索引维护和物化视图刷新的开销，这些正面的、负面的影响都应该在代价估计时予以考虑。

3.5 在测试服务器上完成推荐过程

在实际数据库系统日常运行中，工作负载的量是巨大的，如果直接在生产服务器上完成物理特征的推荐过程，将会影响服务器上数据库实例的正常运行。而在实际工作中，一般 DBA 都会拥有一台测试服务器。如果推荐过程能在测试机上完成，然后在生产机上实施，那么将会带来很大便利。

考虑用以下方案完成这个设想：

1. 从生产服务器上将需要调整的数据库的元数据复制到测试服务器，但不要从表中复制真正的数据。我们不但要导出空表的信息、索引，而且还包括所有的视图、存储过程、触发器，等等。这些都是必需的，因为工作负载中可能会引用这些对象。从数据库系统中导出元数据的脚本信息，目前的 DBMS 都提供这个功能。由于该操作只需要访问系统目录条目，而不依赖数据库的数据规模，这通常是一个很快的操作。

2. 在测试服务器上处理工作负载。为了在测试服务器上获得和生产服务器一样的效果，我们需要数据库服务器提供两个功能：

- 在调整过程中，为了能更精确地优化性能，在测试服务器上必须能访问当前的统计数据。可是统计数据的产生需要生产服务器上的实际数据。这时，应能从生产服务器上将统计数据导入测试服务器。
- 数据库引擎能将生产服务器上的硬件参数在测试服务器上模拟。

3. 将测试服务器上得到的物理设计推荐特征应用到生产服务器上。

图 3.3 表示了本方案的示意图。

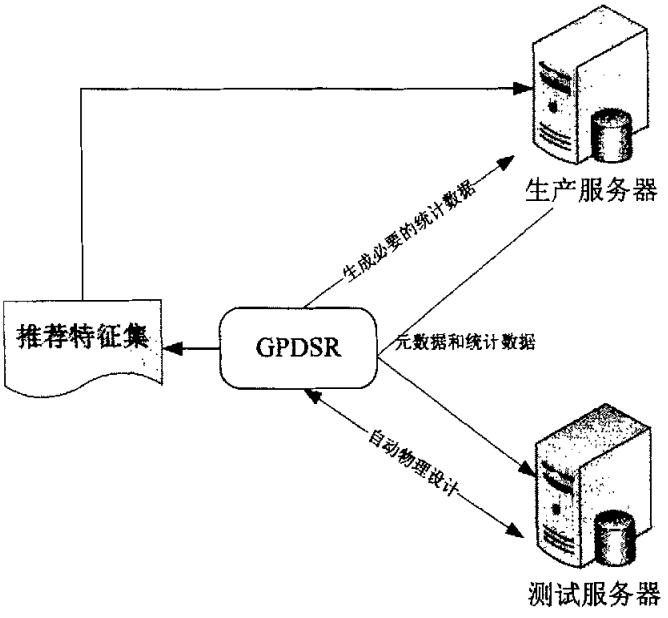

图3.3 生产/测试服务器模式示意图

第四章 OSCAR 索引顾问的设计和实现

数据库管理系统是重要的基础核心软件技术，是国家战略必争的高新技术。为了促进我国自主软件产业的发展、维护国家安全，国家八六三计划在"十五"期间设立了数据库管理系统及其应用重大专项，力求突破 DBMS 核心技术，研发具有自主知识产权的数据库管理系统；在制造业信息化、电子政务、国家信息安全、电子教育等应用中成功应用自己的 DBMS，促进国产数据库实现做得出、用得上、卖得掉的目标。

OSCAR 索引顾问（OSCAR Index Advisor，简称 OSIA）是 GPDSR 在国产大型关系数据库管理系统 OSCAR 上的一个实现。

4.1 神舟 OSCAR 数据库系统简介

神舟 OSCAR 数据库[Osc03]，是拥有完全自主知识产权的企业级大型、通用对象关系型数据库管理系统。它是北京神舟航天软件技术有限公司与浙江大学等高等院校合作，在科技部重大软件专项 "大型通用数据库管理系统及其应用" 和航天科技集团的大力支持下研制成功的，也是目前科技部经费支持力度最大的国产数据库产品。其目标是以市场需求为驱动，研制一个在功能、性能、实用性、稳定性、安全性以及可扩展性等方面能够满足电子政务、企业信息化、电子商务以及国防工业等敏感部门信息化建设需求的大型通用数据库产品，并在这些领域中逐步替换国外数据库系统。神舟 OSCAR 数据库系统性能稳定、功能完善，可广泛应用于各类企事业单位、政府机关，尤其是国防、航天、航空等事关国家政治、军事、经济安全的各要害单位的信息化建设。神舟数据库系统的成功研制和应用，一方面将可能彻底解决由于广泛应用国外数据库产品可能带来的信息安全问题，另一方面也将在系统软件领域极大地推动我国民族软件产业的发展。

4.1.1 特点分析

神舟 OSCAR 数据库在设计中采用了一些先进的相关技术，并吸收了一些当前流行数据库的常用技术，具备以下特点：

通用的数据库管理系统

神舟 OSCAR 数据库管理系统支持 Windows 、 Linux 以及 Solaris 等多种主流操作系统平台，应用 Java 技术定制各种数据库管理工具，具有很好的跨平台支持能力；神舟 OSCAR 数据库管理系统采用成熟的关系数据模型作为核心

的数据模型，支持通用数据查询语言 SQL ，因此可以在各行业中广泛应用。

支持多种计算模式

神舟 OSCAR 支持包括集中式结构、客户 / 服务器结构、 Web 浏览器 /Web 应用服务器 / 数据库服务器多层结构等多种应用体系结构，能满足 Internet/Intranet 环境下的各种应用要求。

符合国际标准的数据查询语言

神舟 OSCAR 所提供的数据查询语言 SQL ，完全符合 SQL92 入门级标准，并部分支持 对 SQL 99 中定义的抽象数据类型 ADT 的扩展支持。在系统核心的查询处理过程中，直接执行对复杂工程数据对象和关系数据的融合处理 。

海量数据管理能力

神舟 OSCAR 实现了可扩展的逻辑和物理双层存储结构管理，因此具有管理海量数据存储的能力。 神舟 OSCAR 支持的数据库最大容量达到 TB 级别，支持的单表最大容量也达到 TB 级别，同时支持的单个大对象的最大容量达到 4GB ，并实现了对数据库内表数目的无限制和表中记录数目的无限制支持。

7×24 不间断的稳定运行能力

通过完善的备份恢复机制以及对系统资源的自适应和自管理，神舟 OSCAR 可以长期稳定、高效的运行，并通过各种管理工具实现在线的数据修复能力，以保证数据的一致性和正确性。

高效的数据处理能力

神舟 OSCAR 实现完整的查询优化策略，支持高效的查询执行方案，具备稳定高效的数据处理能力。同时支持大对象数据管理；支持图形、图像、声音、文字、视频等多媒体数据管理；实现复杂对象数据和关系数据的统一管理；并支持用户自定义数据类型的扩展。

多层次的数据库安全机制

神舟 OSCAR 实现了可靠的用户身份认证，支持自主存取控制和安全的网络连接，并提供对 DBA 、普通用户、数据对象和特定数据操作的各种安全审计。神舟 OSCAR 还对数据的存放和传输提供了多种加密机制以满足不同应用的要求。

完备的数据备份恢复机制

神舟 OSCAR 提供完备的日志和故障恢复机制，支持容错机制，通过实现数据库及日志数据的完整备份、增量备份和联机备份，确保系统在出现系统崩溃、服务器掉电、存储介质错误等各种软硬件故障的时候实现数据恢复。

完整的应用开发接口

神舟 OSCAR 支持最新的 ODBC 、 JDBC 等标准应用开发接口，并针对

主要应用开发工具提供高性能的直接数据接口；支持嵌入式 SQL 、可编程存储过程和触发器。

丰富的数据库管理工具

神舟 OSCAR 提供各种基于 GUI 的交互式智能管理工具，包括系统安装和卸载、 DBA 工具、交互式 SQL 、性能监测与调整以及作业自动调度等，并提供与 Oracle 、 SQL Server 等主要大型商用数据库管理系统以及 TXT 、ODBC 等标准格式之间的数据迁移工具。

4.1.2 OSCAR 数据库系统的体系结构

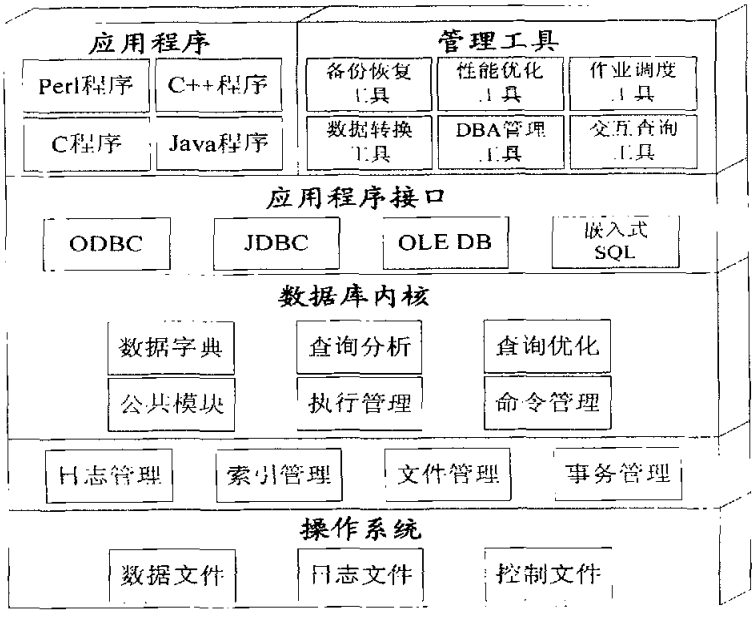

图4.1 OSCAR数据库系统的层次结构

OSCAR 数据库管理系统采用分层次的模块化设计方法，具体分为管理工具、应用程序接口、数据库内核三个主要的层次。OSCAR 数据库管理系统的层次结构如图 4.1 所示。

管理工具提供备份恢复、性能优化、作业调度、数据转换、DBA 管理、交互查询等多种工具。这些工具主要提供给数据库管理和操作人员，使他们能够通过图形化的工具方便地管理 OSCAR 系统。应用程序接口主要提供给程序员使用。包括通用的 ODBC，JDBC 接口以及嵌入式 SQL 等。对于想使用 OSCAR 数据库系统作后台数据库来开发数据库应用程序的程序员，OSCAR 的应用程序接口和其它主流数据库管理系统能够提供的一样完备。在数据库内核部分，其上层包括数据字典、

查询分析、查询优化、公共模块、执行管理和命令管理几个部分；下层包括日志管理、索引管理、文件管理和事务管理几个部分。其中和日志管理密切相关的有文件管理和事务管理两个模块。文件管理部分为日志管理模块提供了对物理文件的读写接口，并且负责管理日志组和日志文件成员。事务管理主要利用了日志管理模块，通过日志管理模块来组织日志，写入日志以及读取日志，并且通过日志记录来进行故障恢复。 另外还有与 OSCAR 系统运行息息相关的三种文件，数据文件、日志文件以及控制文件。日志文件是日志模块负责维护的，用于保存事务日志，是故障恢复的依据。数据文件是存放数据库中数据的，是数据记录存放的地点。控制文件则保存了一些控制信息，用于维护数据库的启动参数和其它控制信息等。

4.2 OSCAR 索引顾问（OSIA）的体系结构

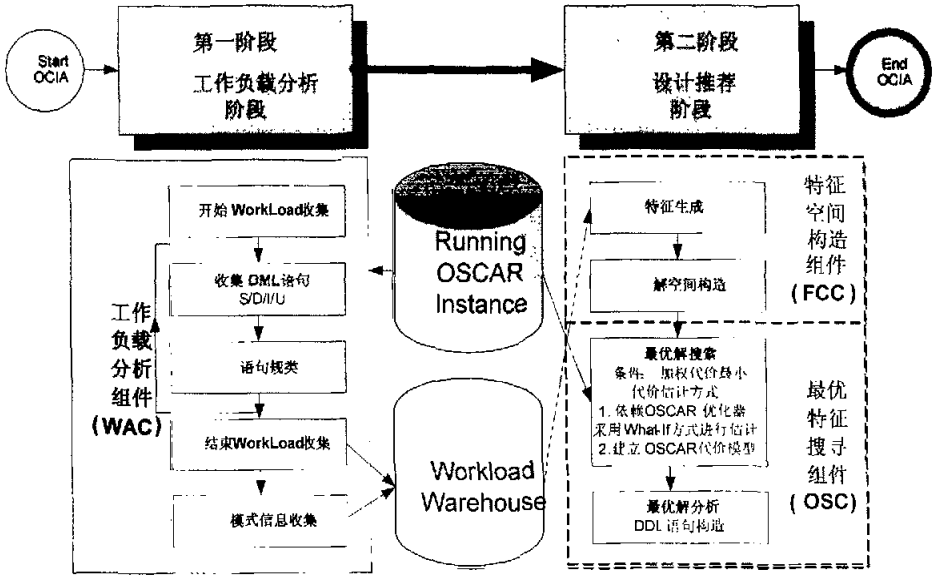

图4.2 OSIA体系结构

如图 4.2 所示，OSIA 的整个工作流程分为两个阶段：

工作负载分析阶段

特征设计推荐阶段

在工作负载分析阶段，工作负载分析组件（WAC）在运行实例上收集该时间段内所有的 DML 语句（包括所有的查询、插入、删除以及更新操作，对应 SQL 语句 SELECT/INSERT/ DELETE/UPDATE）。将这些 SQL 语句中类似的语句进行合并，如 DELETE FROM TEST WHERE A=1 和 DELETE FROM TEST WHERE

A=2，这两条语句应当认为是同一类语句。收集时，将各类 SQL 语句的特征语句，查询分析树等记录到 Workload Warehouse 中，并在收集结束后对其进行简单分析，抽取出一些关键信息，如该类语句的频率、影响的表和属性的列表等。收集完以后，OCIA 根据获取的表和属性等信息，选择性的获取运行实例的模式信息和统计信息，并也将其记录在 Workload Warehouse 中。也就是说，Workload Warehouse 中保存了两类信息：

- 一类是静态的模式信息，包括有哪些需要关注的表，表的定义，现有的索引，表和属性的各类统计信息，等等；
- 另一类是动态的负载信息，各类 SQL 语句，相关表和索引，以及该类 SQL 语句出现的频率。

在特征设计推荐阶段：

- 特征空间构造组件（FCC）根据 Workload Warehouse 中的信息进行针对特征的可行设计生成；
- 最优特征搜寻组件（OSC）在 FCC 生成的所有可行设计组成的解空间内进行最优解搜索，得到最优的一个或者一组最优设计，并作为最终解反馈给管理员。

FCC 分析 Workload 仓库中记录的每一条 SQL 语句，根据模式信息和 OSCAR 在索引使用上的一些特性分别生成可行的索引设计，由这一系列的索引设计构造出 OSC 的输入解空间。OSC 在这个解空间内进行最优解搜索：判断最优时，对于当前解，分析在解所对应的索引模式的环境中，每个查询的代价；并在此基础上，使用每个查询的频率进行加权，得到总代价；最优解就是使得这个总代价最低的代价。OSC 中的代价估计可以采取两种策略，一种是基于 OSCAR 的优化器的 What-If 代价估计，即对每一种建索引方式，在数据库中建立 FAKE 的索引，并要求优化器根据已有索引和 FAKE 索引同时进行代价估计；另一种是采取自行建立的 OSCAR 数据库代价模型进行代价估计。由于自行建立完全独立的代价模型难度太大且没有必要，所以，目前拟采取两种方式结合的办法，最终希望在对 OSCAR 的代价估算机制进行重新设计和编写以后，完全基于 OSCAR 的优化器进行代价估计。

4.3 组成模块

如图 4.3，OSIA 嵌入 OSCAR 关系数据库引擎的部分由以下几个模块组成：

- ✓ STAM 模块
  - 名称：总控模块
  - 作用：接收 DBA 命令，进行处理，并触发对 OSIA 模块所有子模块

的调用，作用类似于 TCOP

- ✓ WLA 模块
  - 名称：WorkLoad Analysis，工作负载分析模块
  - 作用：控制工作负载分析，开始、关闭工作负载分析，对于工作负载进行分析、归类，并将分析获得的信息使用 WWM 保存到工作负载仓库中
- ✓ SCA 模块
  - 名称：SCheme Analysis，模式分析模块
  - 作用：分析工作负载中涉及的所有数据库模式的信息，包含其名称、属性、长度，所占空间等
- ✓ WWM 模块
  - 名称：Workload Warehouse Management，工作负载仓库管理模块
  - 作用：负责工作负载、模式信息的保存、查询、删除等

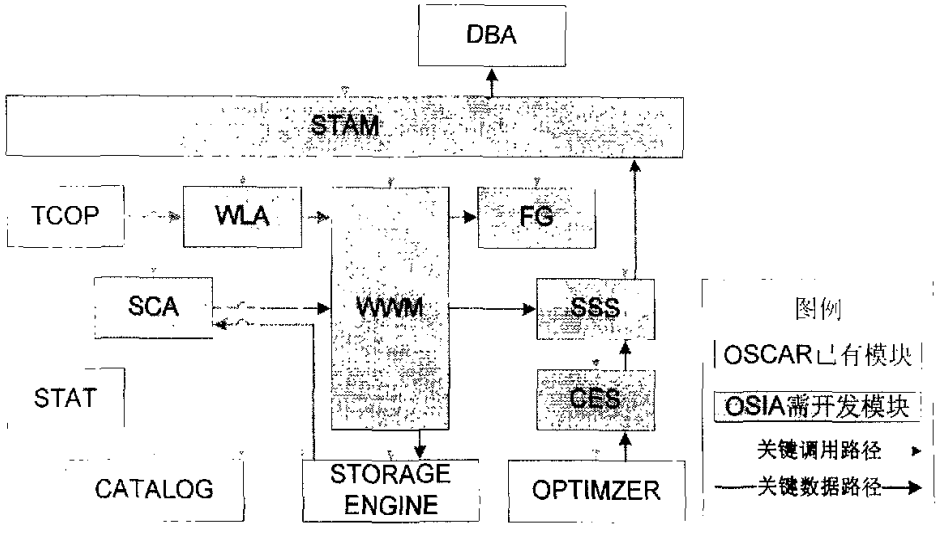

图 4.3 OSCAR OSIA 模块关系图

- ✓ FG 模块
  - 名称：Feature Generation，特征生成模块
  - 作用：根据工作负载中的查询语句，根据 OSCAR 数据库引擎的特性，产生一系列的可能优化性能的索引
- ✓ SSS 模块
  - 名称：Search Space Search，解空间搜索模块
  - 作用：控制在解空间中搜索最佳解的过程
- ✓ CES 模块

- 名称：Cost EStimation，代价估计模块
- 作用：对查询代价进行估计

模块间关系以及与 OSCAR 关系数据库引擎其他相关模块间的关系见图 4.3。

4.4 工作负载分析组件

工作负载分析组件主要是对工作负载进行分析，归类，向特征空间构造组件（FCC）提供关于工作负载的语句模式的统计信息（例如对某个表的某个属性的查询频率，更新频率等），从而帮助 OSIA 求得对于此工作负载能够优化查询性能的 index 建议。

4.4.1 主要数据结构

- **模块数据结构的定义：**

在定义模式树的时候，我们要参考一下 OSCAR 中查询树这个数据结构的定义，从中提取我们需要的数据域，作为模式树的数据结构的元素，同时，我们也要在模式树数据结构中加入一些其他的数据元素。

结构体 Pattern 的定义，用来存储语句的模式。

```
struct Pattern
{
        NodeTag         type;
        CmdType         commandType;
        Oid             relid;
        PtnFromExpr_t   *jointree;
        List            *targetList;
        Count_t         nCount;
}
```

说明：

- type 为 T_Pattern_t;
- commandType 存放命令的类型，select 对应 1，update 对应 3，insert 对应 4，delete 对应 5;
- relid 为 insert，delete，和 update 的目标表，对于 select 语句暂时为 0;
- jointree 为结构体 PtnFromExpr_t，存放 SQL 语句的“from ... where ...”部分;
- targetList 为 insert 或 update 语句的目标属性链表，是结构体 PtnTargetAttr_t

的链表；

- nCount，计数，记录相同模式的语句数目

- **数据仓库定义：**

OSCAR 数据库中，我们模块使用的数据仓库是使用一个动态 HASH 表来实现，该 HASH 表中的元素就是模式树指针。使用 HASH 可以取得比链表更好的性能。

4.4.2 工作负载分析组件工作流程

- **输入：查询树。**
- **输出：模式树。把当前的模式树反映到模式信息数据仓库中。**

WAC 从 OSCAR 数据库管理系统的 parse 和 analyze 模块取得一个查询树，这个查询树对应一个 SQL 语句的内部表示。它能够把该查询完整准确地表示出来。查询树里面有很多关于该语句的内部表述信息，比如查询对应的表，查询的目标，是否对结果进行排序等等，但 WAC 处理的信息却并不是这些信息的全部，它处理部分对我们划分模式类型有用的信息。所以，WAC 的输入是可以缩小到一定的范围的，毕竟整个查询树太大了。

- **处理过程：**

当获得输入后，WAC 根据对它有用的查询树中的信息，根据不同的语句类型做一次查询树的中序遍历，把查询树中对划分模式类型有用的信息提取，然后重新生成模式树的节点，直到最终生成整个模式树。当针对当前的 SQL 语句模式树产生之后，我们不能把它直接加到用来存储系统整个模式信息的数据仓库中(实际上是一个动态 hash 表)。因为该模式可能是以前已经收集进来了。我们先要使用 hash 函数计算出当前模式树中的 hashcode，然后根据 hashcode 取得在当前数据仓库中的模式树，这个时候有两种可能：

- 如果模式树不存在，说明以前没有存储过该模式树，我们可以把它加到模式仓库中。然后把表示模式出现次数的数据域变成 0，再把对应于该模式的样本 SQL 语句记录在模式树的数据结构中。
- 如果模式树已经存在了，我们没必要再次存储该模式树，只需要把原来模式树中用来表示模式出现次数的数据域自加 1 就可以了。

工作流程如下：

1. 开始执行
2. 对查询树中的 N 个有用节点 i=0 ，for i to N .
3. 提取信息的同时，动态开辟生成模式树节点，直到所有模式树节点生成
4. 调用 hash 函数计算产生的模式树的 hashcode

5. hash 表中取得模式树，

   如果模式树不存在：

   添加模式树进 hash 表。

   如果存在

   提取模式树，出现次数自加 1。

6. 更新数据仓库 hash 表。

4.5 特征空间构造组件

特征空间构造组件（FCC）主要是对工作负载分析组件输出的模式树进行分析，生成候选特征（索引）集，作为最优特征搜寻组件的输入。

4.5.1 主要数据结构

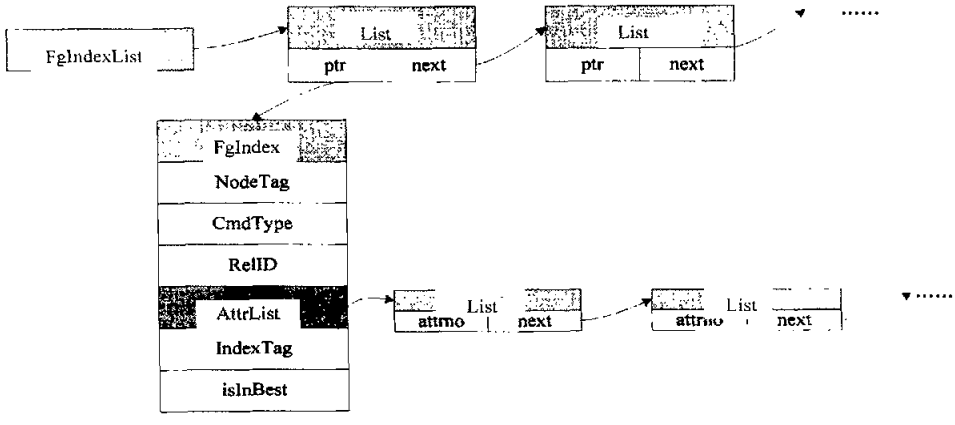

图4.4 FgIndexList数据结构

```
struct FgIndex
{
        NodeTag     type;
        CmdType     commandType;
        Oid         relid;
        List*       attrlist;
        IndexTag    idxtype;
        BOOL        IsInBest;
}
```

- 说明： FgIndex 是模块内用的结构，用于存放生成的候选索引。最后组合成列表 FgIndexList，作为模块返回值，

- NodeTag：表示结构类型
- CmmandType：存放命令的类型，select 对应 1，update 对应 3，insert 对应 4，delete 对应 5；
- RelID：表示操作对应的表
- AttrList：表示表中的属性列（可能要建立多维属性，如果是一维的，那么该列表中只有一个值）。
- IndexTag:表示索引的类型。目前，oscar 还只支持 Btree。可用于以后的扩展。
- IsInBest:判断是否在最优集合中

4.5.2 特征空间构造组件工作流程

特征生成算法：

- 功能：产生一个候选特征（索引）集合
- 输入：模式列表
- 输出：候选特征（索引）集合

说明：N 表示模式的个数，pt_i 是第 i 个模式（$1 \leqslant i \leqslant N$），$F$ 是特征集合的计算结果

1. 开始执行
2. 设置 F 为空；N 为输入模式的个数；
3. 从 $i=1$ 到 N 解析 pt_i
    a) 生成索引列表；
    b) 对每个索引列表中的项：
        - 判断该候选项是否无效；如果是的继续下一个项
        - 判断该候选项是否已经在 F 中；如果是的继续下一个项
        - 将添加到集合 F 中
4. 返回 F，作为候选特征（索引）集合
5. 执行结束

具体说明：此模块（FCC）的输入是 WAC 的输出，为了提高计算速度，WAC 输出的模式列表中，已经不存在重复的情况了。此模块的输出，将是 OSC 的输入，OSC 模块的复杂度跟候选特征的个数相关。为了提高其速度，OSC 也要对这些索引进行筛选，比如说去掉数据库中已经存在的索引等。在 FCC 模块中，也进行了部分处理，比如说去掉无效的候选索引，这里的无效指的是不是有效用户表的候

选索引。

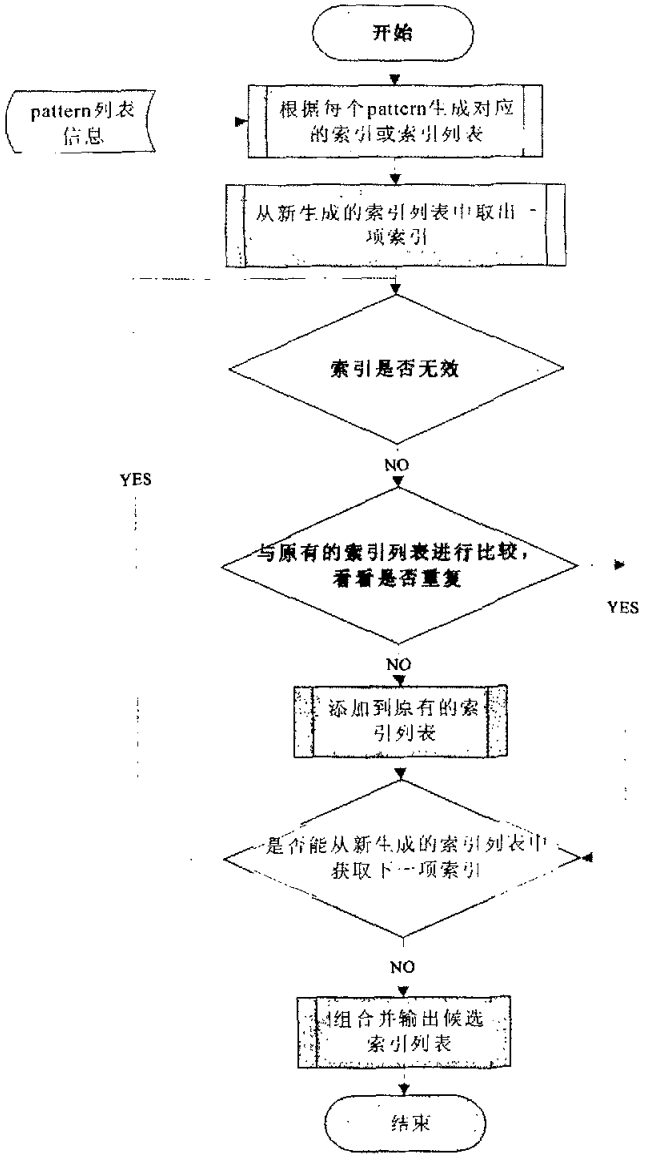

图4.5 FCC流程图

函数复杂度分析：设 M 是最后输出集合的大小，复杂度则为 O（$N * M$）

我们可以对该函数进行改进，在这里，返回的是用集合表示的，如果我们用 HASH 表来存储中间结果，设 m 是 HASH 表中最大一项的个数，复杂度则为 O($N * m$) ($m << M$)

4.5.3 特征分析

在本组件中，候选特征（索引）的表示是有两部分表示的：表 ID 和属性下标，形式如<TableId, AttrNo+>；其意思是在表 TableId 上建立索引（AttrNo+）；可以是某个属性上建索引也可以是某几个属性上建索引。

- **对 DML 语句进行分析**

➢ Insert：首先对 key（唯一的），进行搜索。这个 key 一般是已经建立索引的。（当然也有例外的，则将此 key 列入候选的 Feature）。如果没有 key，那么就将记录加入表中，然后更新原来该表的索引树！（可能在某些属性上已经建立了索引，这个跟 FCC 无关）。

如：CREATE TABLE CS_COURSE (

CNO CHAR(4), 课程号

CNAME VARCHAR(64), 课程名

CCREDIT NUMERIC(3,1), 学分

PRIMARY KEY(CNO));

Insert into CS_COURSE values（'C001'，'C 程序设计'， 3.0 ）；

那么 CNO 将被加到候选的 Feature（index）列表中。

➢ Delete：将 delete 中属性列（where 语句中的）作为候选的 Feature。（如果是条件删除的话）。

delete from CS_COURSE where CNAME = 'C 程序设计'；

那么 CNAME 将被加到候选的 Feature（index）列表中。

➢ Update：将 Update 中属性列（where 语句中的）作为候选的 Feature。（如果是条件更新的话）。

update CS_COURSE set CCREDIT = 4.0 where CNAME = 'database'；

那么 CNAME 将被加到候选的 Feature（index）列表中。

如果 CCREDIT 已经被建立了索引树，那么 update 记录的同时，也得 update Btree。（保持同步，不过这个不在 FCC 考虑的范围）。

➢ Select：这个是各种 MDL 语句中，最复杂的。上述的一些语句，主要是在 where 语句中，而 select 语句不仅涉及 where 语句还有其他的，如 select 的属性列。故这个将被详细考虑和讨论。其他的语句（insert/delete/update）的其他种种情况，都可以在 select 的讨论中找到解决之道。

在 MDL 语句中，select 语句也是最复杂的，它同时也可以作为子查询，放在 MDL 语句（select/insert/update/delete）中。这些子查询跟 select 查询的处理，是类似的。

- select 语句的分析

列举以下各种情况：

- Select Into 语句，语法是 SELECT INTO *target query_expression*；将结果插入到列表中。Index 的候选，与 Select Into 无关，相关的是 *query_expression*（可能含有 from 或者 where 等语句）。
- Select ：for update，该子句的语法是 FOR UPDATE [OF {table_name [, ...]|column_name [,...]}]；用于指定对扫描经过的行加排它锁。与 index 的候选无关。
- Select : from，该子句为 SELECT ，UPDATE 或 DELETE 语句声明数据源，语法是 FROM table_reference [, ...]；这里没有属性列，但是当有几个 table 的时候，其中的列可能会用于一般的联接，或是用于 Merge Join。那么这些选中的列，将被添加到 index 的候选中。
- Select : group by，该子句将查询结果加以分组，语法是 GROUP BY expr [, ...]；如果 expr 是列名或是与某（些）列相关的操作（比如 a^2，a 是列名），那么这些列将被添加到 index 的候选中。
- Select : having，该子句对分组的结果进行条件判断。与 where 子句形式相同。其中涉及的列，将被列入 index 的候选中。
- Select : limit，该子句限制结果集的范围，语法是 LIMIT { count | ALL } [OFFSET offset]；与 index 无关。
- Select : order by，该子句对查询结果进行排序，其语法是 ORDER BY expr [ASC | DESC | USING operator] [, ...]；expr 用于排序的表达式，通常是一个字段名，但也可以是一个复杂的表达式。那么这些相关的字段名将被添加到 index 的候选中。
- Select ：stop，该子句为集合操作，语法是 [UNION|INTERSECT|EXCEPT][ALL] {select_stmt | (select_stmt)}；与 index 无关。
- Select : where，该子句为 SELECT, UPDATE 或 DELETE 语句指定查询条件，语法 WHERE condition；条件中涉及到的列，将被添加到 index 的候选中。Where 子句有简单和复杂之分。简单的如：SELECT A FROM TEST WHER A=1 AND B= 'aaa'；那么(A)，(B)，(A，B)这几个将被添加到 index 的候选中。复杂的，如：SELECT A FROM TEST WHERE A in (select A from ……)；那么(A) 将被添加到 index 的候选中，同时还要对(select A from ……)子查询进行分析。

从上述分析可见，在 DML 语句中，主要涉及 FCC（index 的候选）的是

select/update/delete。而 update/delete 涉及的子句有 from 和 where。这两者都在 select 中有出现。而 select 除了 from 和 where 外，还要 group by，having，order by。而 order by 和 group by 涉及的列，用 index 的话，对性能提高很明显的。Having 类似于 where。故分析的重点就在 from 和 where 了。

From 子句，主要是 table 的联接。与 index 相关的是一般连接操作用到的列，Merge join 用于排序的键。

Where 子句，与 index 相关的是条件中涉及到的列。

4.6 最优特征搜寻组件

最优特征搜寻组件(OSC)目的是获取代价最小的虚拟索引组合。采用贪婪算法，将 FCC 模块输出的推荐虚拟索引经剔除、筛选、组合后得到最优的虚拟索引组合。

OSC 在由 FCC 输出的虚拟索引组合内进行最优解搜索：判断最优时，对于当前解，分析在解所对应的索引模式的环境中，每个查询的代价；并在此基础上，使用每个查询的频率进行加权，得到总代价；最优解就是使得这个总代价最低的代价，最后将最优解所对应的虚拟索引组合推荐给 DBA。

4.6.1 主要数据结构

1. SsNode

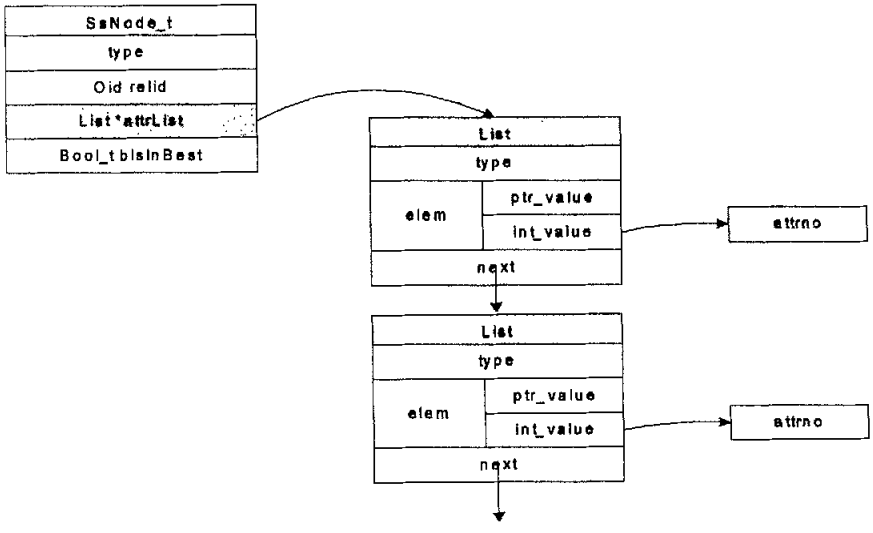

图4.6 SsNode数据结构

本组件接收 FCC 模块的输出——包含推荐索引的链表，通过本模块处理后，出 SsNode 记录这些索引的信息。

```
struct SsNode
{
        NodeTag     type;
        Oid         relid;
        List        *attrList;
        Bool_t      bIsInBest;
}
```

说明：

- Relid：所属表的 Oid
- attrList：待建索引的属性号
- bIsInBest：是否已经在最小代价链表里

2. SsFakeIndex

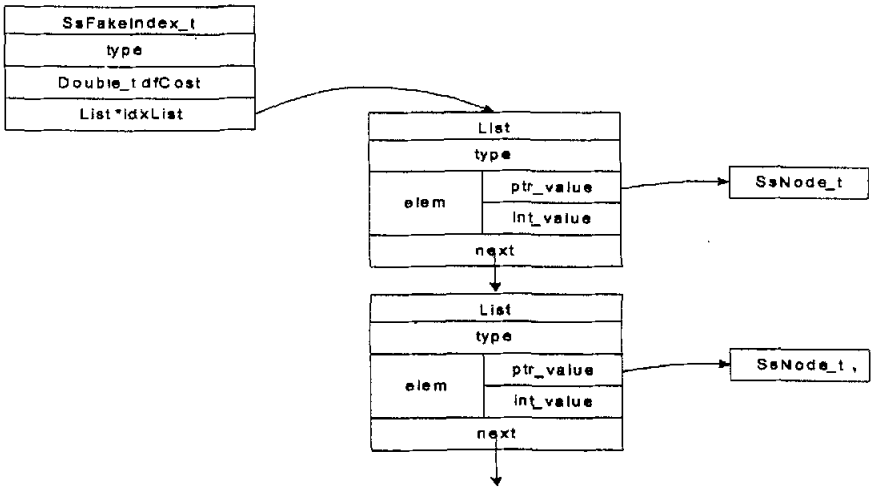

图4.7 SsFakeIndex数据结构

记录最低代价的组合 fake 索引的节点信息。

```
struct SsFakeIndex
{
        NodeTag     type;
        Double_t    dfCost;
        List        *idxList;
}
```

说明：

- dfCost：组合 fake 索引的总代价
- idxList：组合 fake 索引的链表

4.6.2 最优特征搜寻组件工作流程

怎么对推荐的索引集进行代价估算呢？

如果在数据库中建立索引，那太大、太慢了，对当前数据库实例会带来很大影响。

在 OSIA 中，我们的办法是在数据库里创建 FAKE 索引——骗过优化器，就是除了让优化器以为索引的存在以外什么也不干。

- 功能：利用贪婪算法搜索使得整个 workload 代价最小的虚拟索引组合
- 输入：

  1）FCC 模块初步推荐的虚拟索引集合

  2）WAC 模块收集的各工作负载 pattern 的频率
- 输出：代价最小的虚拟索引组合

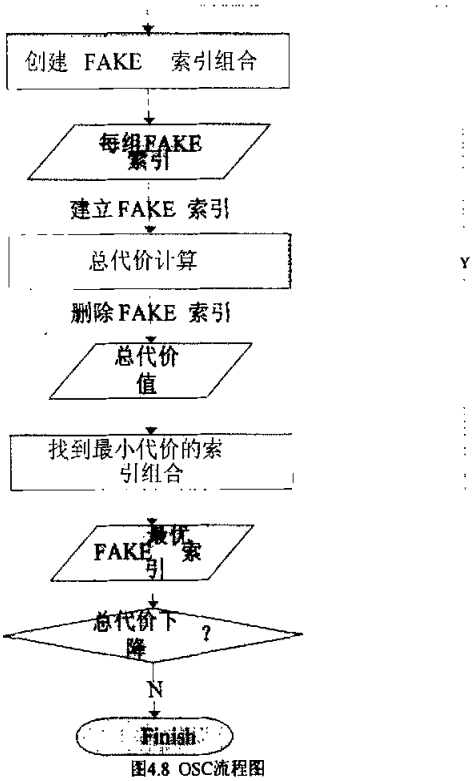

图4.8 OSC流程图

具体说明：

1．剔除：将 FCC 模块输出的虚拟索引和已经存在的索引对比，将已存在的剔除，得到一个新的虚拟索引集合；

2．筛选、组合：开始时将每个虚拟索引作为一个索引组合，计算每个索引组合对 workload 的代价，选出最小代价的索引组合作为当前最小代价索引组合；然后将这个索引和剩下的每个索引组成新的索引组合，计算新的索引组合对 workload 的代价，选出最小代价的索引组合；如果此次的代价比上一次的最小代价小，那么将此次最小代价索引组合作为当前最小代价索引组合，然后开始新一轮的组合、计算代价；否则，返回当前最小代价索引组合。

4.7. 实验分析

现在，已经建立了OSIA原型，作为OSCAR数据库引擎的索引推荐器。用TPC-C[Tpc05]对OSIA进行测试。

1，装载数据，然后运行测试程序和索引推荐器30分钟，获得推荐的特征（索引）。

2，应用这些特征（索引），重新运行测试程序30分钟。实验结果如下：

TPC-C测试：5个数据仓库，只有主键和外键，没有其他的索引。推荐的索引如下：（表示形式是 表名（列名1，列名2，…））

- customer(c_last，c_d_id，c_w_id);
- orders(o_d_id，o_w_id，o_c_id);
- order_line(ol_o_id，ol_d_id，ol_w_id，ol_i_id);
- stock(s_i_id，s_w_id，s_quantity);

选择一些重要的 TPC-C 参数，对索引创建前后进行对照。结果如表 4.1。结果说明了 GPDSR 框架的可行性和有效性。

	TPMC	90%响应时间（越小越好）		
	(越大越好)	New Order	Stock Level	Delivery
索引创建前	26.6	62.98*s*	46.31*s*	85.43*s*
索引创建后	85.8	0.21*s*	2.32*s*	0.56*s*

表 4.1：TPC-C 测试结果

4.8 小结

采用 GPDSR 框架以及基于该框架实现的智能索引推荐器 OSIA 具有以下的先进性和创新性：

- 工作负载分析通过先进的 SQL 语句的语义模式树提取和匹配技术实现，支持复杂 SQL 语句的关键语义模式抽取，并对不影响数据库设计的因素进行过滤，从而达到将工作负载中的每一条 SQL 语句进行归类和分析的目的。传统的工作负载分析一般通过简单的常数替换过滤后的字符串比较来实现 SQL 语句的归类；这种归类方式灵活性和精确性都较差，甚至认为 SELECT * FROM TABLE WHERE A + 1 = B 和 SELECT * FROM TABLE WHERE 1 + A = B 都是不同的语句；OSCAR 采用的归类技术则完全基于对 SQL 语句的语义进行详细的分析和抽取，使用抽取得到的模式进行匹配，大大提高了负载分析的灵活性和精确性。
- 自适应数据库设计框架通过使得特征构造组件和工作负载分析组件、最优特征搜寻组件的松散耦合，实现了灵活地根据给定的物理设计特征库和工作负载分析组件得到的工作负载，推荐出一系列可能改进性能的物理设计的功能。最优特征搜寻组件可选的支持各种不同的解搜索方式，包括穷举搜索、贪婪搜索、遗传算法搜索等，默认使用贪婪搜索算法实现，该算法能满足大多数数据库应用的自动物理设计要求。
- 两个阶段（负载分析阶段和推荐搜索阶段）之间的松散耦合（仅通过运行的 OSCAR 实例和负载模式数据仓库进行耦合）的方式，使得自动物理设计过程的大量计算（最优解搜索）可以在独立于数据库实例运行主机的机器上进行。该方式使得自动设计过程对于正在运行实例的性能影响较小，且可以异步地进行。
- 该框架使用 SQL 语言函数的形式提供调用接口，供管理界面和二次开发程序进行调用，使用灵活方便。

OSCAR 数据库管理系统实现了部分自适应配置、诊断、调优和维护功能，极大程度地降低了管理员的维护时间和成本，也降低了 OSCAR 数据库管理员的技术门槛。

OSCAR 数据库的自适应配置、诊断、调优和维护功能包含以下方面：

- 快速轻量级的数据库安装配置功能，提供提问式和归类式地配置向导生成 OSCAR 配置文件。其中提问式配置向导，根据管理员对部分关键问题的回答，分析管理员的配置意图，生成配置文件；而归类式配置向导通过将以前以配置英文名称排序的配置文件，按照分类的方式将最常用的部分配置组织起来供管理员选择配置。通过此两类配置向导，达到使得管理员的配置工作更加简单，而配置参数更加简化的目的。
- 内建的自诊断、自调优和自维护技术，实现数据库引擎的自我管理，包括以下关键技术点：

- 性能相关的统计信息和查询负载信息的收集和分析技术；
- 根据查询计划、估计代价以及实际查询性能，对相关的用于代价估计的行数、页数、总大小、平均大小等统计信息进行自动生成和更新的技术；
- 自动的系统高速缓存和存储空间管理，根据不同数据访问模式，自动调整局部或者全局数据文件页面缓存和调度策略，自动调整日志读写缓存策略，并提供自动的数据文件存储空间回收、重复利用以及日志文件循环利用技术；
- 根据 OSCAR 数据库内部实现，建立 OSCAR 性能问题诊断知识库，并对各个可能影响性能的参数进行跟踪，根据根系多次数据库性能参数快照以及其间的关系，分析当前存在的性能问题，并给出解决办法。

- 实现一组智能化的数据库管理工具，包括：
  - OSCAR 索引推荐器（OSCAR Index Advisor）为管理员提供对索引推荐功能进行管理的用户界面；
  - OSCAR 性能诊断器（OSCAR System Doctor）为管理员提供对 OSCAR 进行性能诊断的用户界面；
  - OSCAR 参数配置器（OSCAR Configuration Wizard）为管理员提供参数配置向导的用户界面。

存在问题：

- **代价计算**

目前代价计算采用了 OSCAR 中优化模块的代价估算函数，由于在建虚拟索引时不能获取所在表的具体信息，加上优化模块代价估算函数本身的误差，使得计算出来的代价不一定准确。将来可以设计一个全新的代价估算模型，建立新的代价估算函数，提高代价计算的准确度。

- **搜索算法**

目前采用的贪婪算法经实际测试后，发现此算法有时候不能得到最优虚拟索引组合，有些时候不仅仅得到最优解，还有一些对整体影响不大的虚拟索引组合。这个和贪婪算法在第一轮筛选出的虚拟索引有关，如果此次虚拟索引在最优解里，那么此次的结果比较准确；如果此次虚拟索引不在最优解里，那么会导致此后的一系列筛选都不怎么好，最后出来的结果也不准确。将来可以尝试采用别的背包问题算法来较好地解决这个问题。

第五章 总结与展望

5.1 数据库管理系统的发展趋势

我们的数据库系统变得越来越复杂。这个趋势是被不断推出的版本中所有新的特性和功能所带来的，还有如下的事实：大多数的企业都实现了若干种不同结构的数据库。今天现代的数据库管理系统提供了 70 年代的数据库前辈们所无法想象的功能。数据库管理系统不断吸取新的特性。例如，今天的数据库管理系统产品通常会提供分析/在线分析处理和 ETL 的特性。在 90 年代，这些都是通用的产品，但是今天他们都是通用的特性。所以，数据库管理系统变得越来越复杂。此外，我们还添加了新的数据类型(BLOB , CLOB 等)、代码(用户自定义函数、触发器、存储过程)，以及对 XML 的支持。这些都让今天的数据库管理系统功能更加丰富，同时，也更复杂。数据库管理系统要达到很高的性能，这是一项困难的任务。在 DBMS 中，各种硬件资源和软件资源之间存在着竞争。为了使 DBMS 获得高性能，在资源分配过程中必须进行精心的平衡。

计算机硬件平台的迅猛发展，用户需求的急剧增加，推动着数据库技术的进步，出现了一批新的数据库技术，如 Web 数据库技术、并行数据库技术、数据仓库与联机分析技术、数据挖掘与商务智能技术、内容管理技术、海量数据管理技术等。

目前数据库技术领域主流发展趋势是[孟小峰 04]：

- 信息集成系统的方法可以分为:数据仓库方法和 Wrapper/Mediator 方法.
- 数据流管理
- 传感器数据库技术
- XML 数据管理
- 网格数据管理
- DBMS 的自适应管理
- 移动数据管理
- 微小型数据库技术又可以进一步分为:超微 DBMS(pico-DBMS)、微小 DBMS(micro-DBMS)和嵌入式 DBMS 3 种.
- 数据库界面的研究

5.2 本研究的后续工作方向探索

5.2.1 本文小结

数据库系统的性能调优一直以来都是一个被广泛关注的主题。在信息技术日益发达的今天，科学技术各个领域间相互渗透，数据库技术也有了更加广阔的发展空间，也使得数据库系统性能调优变得越来越具有挑战性。在严峻的市场环境催动下，近年来，数据库自我调优技术不管是在理论上还是在产品实现上都迅速发展起来。

本文研究了物理数据库自动设计方面的内容。数据库系统中物理特征自动设计是数据库自我调优技术的一个重要方面。首先对目前主流商业数据库 SQL SERVER、ORACLE、DB2 在物理数据库自动设计方面的最新进展作了综述，然后提出一个通用数据库物理模式推荐框架（GPDSR），介绍了三个协同工作的组件——工作负载分析组件、特征空间构造组件、最优特征搜寻组件的原理。最后是该框架在国产自主知识产权大型通用数据库神舟 OSCAR 数据库系统上的一个实现原型。

目前，物理设计特性库包含了 OSCAR 数据库中对索引进行设计的信息，从而使得整个框架能够用于索引自动设计；然而，只要增加物理设计特性库中的特性设计描述信息，该框架就能被扩展支持其他物理设计功能（如：物化视图、垂直分区以及水平分区等）；甚至，通过增加其他数据库管理系统的物理设计描述信息，并对工作负载分析组件和最优特征搜寻组件进行少量的修改，该框架能用于其他数据库管理系统的自适应设计中（如：MySQL、PostgreSQL 等）。这样的设计使得在我们为 OSCAR 数据管理引擎新增加一种物理特性时，能够使用较简单的方式实现对该物理特性的智能推荐。

5.2.2 今后发展方向

数据库管理系统是重要的基础核心软件技术，是国家战略必争的高新技术。研发我国自主知识产权的大型通用数据库管理系统，促进我国自主软件产业的发展、维护国家安全意义重大。

本研究作为“八六三”计划重大专项 OSCAR 数据库系统项目的配套项目，在开发过程中是和 OSCAR 的开发紧密结合的。

由于 OSCAR 数据库当前支持的 Feature 所限，目前只是在 OSCAR 上实现了一个索引推荐器——OSCAR 索引顾问（OSIA）的原型。

其它各种特征（如物化视图、表分割等）的自动设计将在以后的开发过程中，

和 OSCAR 一起不断完善。

数据库系统被期盼能自我管理、自我修复和永不崩溃[Jim04]。我们研究者和开发者都在朝着这个目标而努力。我们相信，达到实质意义上的“无可调部分”，即开发出无须 DBA 的 DBMS 是可能的[孟小峰 04]。

参考文献

[Abr02] Abraham Silberschatz, Henry F.korth, S.Sudarshan, "Database System Concepts",Fourth Edition.McGraw-Hill Companies,Inc, 2002.

[Agr04] S. AGRAWAL., S. CHAUDHURI., L. KOLLÁR., A.P. MARATHE., V.R. NARASAYYA., and M. SYAMALA., "Database Tuning Advisor for Microsoft SQL Server 2005"., In: Proceedings of the 30th International VLDB 2004 Conference., Toronto., Canada., 2004., pp. 1110-1121.

[Baa03] F. BAADER., D. CALVANESE., D. MCGUINNESS., D. NARDI., and P.P. SCHNEIDER., THE DESCRIPTION LOGIC HANDBOOK: Theory., implementation and applications., Cambridge University Press., Cambridge., 2003.

[Ber98] Bernstein, Brodie, Ceri et al. "The Asilomar Report on Database Research". SIGMOD Record. 1998, 27: 74-80.

[Ben03] Benoit., "Automatic Diagnosis of Performances in Datagbase Management Systems"., Queen's University, Kingston, Ontario, Canada,2003.

[Cha04] S. CHAUDHURI., B. DAGEVILLE., and G. LOHMAN., "Tutorial: Self-Managing Technology in Database Management Systems"., In: Proceedings of the 30th International VLDB 2004 Conference., Toronto., Canada., 2004.

[CDL04] Chaudhuri, Dageville and Lohman., "Self-Managing Technologies in Database Management Systems". In: International Conference on Very Large Database Systems,2004.

[Cod70] Codd ., "A Relational Model of Data for Large Shared Data Banks. Commun. ACM. 1970, 13: 377-387.

[Cod71] Codd "Normalized Data Structure: A Brief Tutorial." In: SIGFIDET Workshop,1971. pp. 1-17.

Codd "Relational Completeness of Data Base Sublanguages. "IBM Research Report RJ 987, San Jose, California. 1971.

[Cod73] Codd "Understanding Relations". FDT - Bulletin of ACM SIGFIDET. 1973, 5: 63-64.

[Cra06] Craig Mullins., "新技术引发'数据爆炸'", http://searchdatabase.techtarget.com.cn/183/2314183.shtml,2006

[Dag04] B. DAGEVILLE., D. DAS., K. DIAS., K. YAGOUB., M. ZAÏT., and M. ZIAUDDIN., "Automatic SQL Tuning in Oracle 10g"., In: Proceedings of the 30th

International VLDB 2004 Conference., Toronto., Canada., 2004., pp. 1098-1109.

[Dan04] Daniel C. Zilio, et al., "Recommending Materialized Views and Indexes with the IBM DB2 Design Advisor.", In: First International Conference on Autonomic Computing., 2004., pp. 180-188.

[Dom87] Domanski "A Prolog-based Expert System for Tuning MVS/XA". In: the 1987 Computer Measurement Group,1987. pp. 160-166.

[Elm03] Elmasri, R., and Navathe, S.B., "Fundamentals of Database Systems", 4th Edition, Massachusetts, Addison Wesley, 2003.

[Hel85] Hellerstein and VanWoerkem YSCOPE: A Shell for Building Expert Systems for Solving Computer-Performance Problems. In: the 1985 Computer Measurement Group,1985.

[Hel97] Hellerstein., "Automated Tuning Systems: Beyond Decision Support". In: the 1997 Computer Measurement Group,1997.

[Jim04] Jim Gray., "The Next Database Revolution.", In: SIGMOD 2004, Paris, France.,2004

[Kar05] Karl Dias, Mark Ramacher, Uri Shaft, Venkateshwaran Venkataramani, Graham Wood, "Automatic Performance Diagnosis and Tuning in Oracle".,In: Proceedings of the 2005 CIDR Conference., 2005.

[Mar04] Martin, Powley and Benoit., "Using Reflection to Introduce Self-Tuning Technology into DBMSs". In: Internation Database Engineering and Application Symposium,2004.

[Ora00] Oracle Oracle 8i with Oracle Fail Safe 3.0. White Paper, Oracle Corporation. 2000.

[Ora05] Oracle Corporation: "Oracle Database 10g Release 2: The Self-Managing Database ".,Oracle White Paper,2005.

[Osc03] OSCAR5.0项目开发组,OSCAR产品白皮书，内部资料，2003.

[Pad03] S. PADMANABHAN., B. BHATTACHARJEE., T. MALKEMUS., L.CRANSTON., and M. HURAS., "Multi-Dimensional Clustering: A New Data Layout Scheme in DB2"., In: Proceedings of the 2003 International SIGMOD Conference., San Diego., USA.,2003., pp. 637-641.

[Tpc05] TPC Benchmark C., http://www.tpc.org/tpcc/spec/tpcc_current.pdf., available on Nov. 29., 2005.

[Wei02] Weikum, Moenkeberg, Hasse et al. ,"Self-tuning Database Technology and Information Services: from Wishful Thinking to Viable Engineering". In: Proceedings

of the 28th International Conference on Very Large Data Bases,2002. pp. 20-31, HongKong, China.

[Zil04] D.C. ZILIO., J. RAO., S. LIGHTSTONE., G.M. LOHMAN., A. STORM., C. GARCIA-ARELLANO., and S. FADDEN., "DB2 Design Advisor: Integrated Automatic Physical Database Design"., In: Proceedings of the 30th International VLDB 2004 Conference., Toronto., Canada., 2004., pp. 1087-1097.

[孟小峰04] 孟小峰, 周龙骧, 王 珊, "数据库技术发展趋势"., 软 件 学 报Vol.15, No.12,2004., pp.1822-1836.

[郑宗汉05] 郑宗汉，郑晓明. 算法设计与分析., 清华大学出版社.，2005.

攻读硕士期间发表论文

发表论文：

基于推理的数据库物理模式推荐，计算机应用研究。（已录用）

致谢

在我的硕士学位论文完成之际，对给予我悉心指导的董金祥教授、陈刚教授、胡天磊博士以及课题组和实验室的所有成员表示深深的感谢。

感谢董金祥教授，董老师平易近人，他渊博的知识、严谨的治学态度、悔人不倦的风范是我学习的楷模。

感谢陈刚教授，陈老师积极进取，不畏艰难，勤奋认真的科研精神和乐观开朗的生活态度是我非常敬佩的。在论文进行的各个阶段，他一直关心着工作的进展情况，并进行精心的指导。

感谢胡天磊博士，在论文的研究过程中，他一直给我无私的帮助，在我遇到疑问时给出耐心的解答。

感谢洪银杰、张晓龙同学，在论文的进展中，总是得到他们的帮助。

感谢课题组所有人员，论文的完成和他们的工作是分不开的。

最后我要把我的论文献给我的家人，向她们致以深深的谢意。

丁朝辉

二〇〇六年三月于求是园

数据库物理模式推荐框架研究及其在OSCAR中的实现

作者： 丁朝辉
学位授予单位： 浙江大学

参考文献(31条)

1. 参考文献

2. Abraham Silberschatz. Henry F korth. S Sudarshan Database System Concepts 2002

3. S AGRAWAL. S CHAUDHURI. L KOLLAR. A. P MARATHE, V. R. NARASAYYA, M. SYAMALA Database Tuning Advisor for Microsoft SQL Server 2005 2004

4. F BAADER. D CALVANESE. D MCGUINNESS. D. NARDI, P P SCHNEIDER THE DESCRIPTION LOGIC HANDBOOK:Theory, implementation and applications 2003

5. Bemstein Brodie Ceri The Asilomar Report on Database Research 1998

6. Benoit Automatic Diagnosis of Performances in Datagbase Management Systems 2003

7. S CHAUDHURI. B DAGEVILLE. G LOHMAN Tutorial:Self-Managing Technology in Database Management Systems 2004

8. Chaudhuri Dageville. Lohman Self-Managing Technologies in Database Management Systems 2004

9. Codd A Relational Model of Data for Large Shared Data Banks 1970

10. Codd Normalized Data Structure:A Brief Tutorial 1971

11. Codd Relational Completeness of Data Base Sublanguages[IBM Research Report RJ 987, San Jose, California] 1971

12. Codd Understanding Relations 1973

13. Craig Mullins 新技术引发'数据爆炸' 2006

14. B DAGEVILLE. D DAS. K DIAS. K. YAGOUB, M. ZAIT, M. ZIAUDDIN Automatic SQL Tuning in Oracle 10g 2004

15. Daniel C Zilio Recommending Materialized Views and Indexes with the IBM DB2 Design Advisor 2004

16. Domanski A Prolog-based Expert System for Tuning MVS/XA 1987

17. Elmasri R. Navathe S B Fundamentals of Database Systems 2003

18. Hellerstein and VanWoerkem YSCOPE A Shell for Building Expert Systems for Solving Computer-Performance Problems 1985

19. Hellerstein Automated Tuning Systems:Beyond Decision Support 1997

20. Jim Gray The Next Database Revolution 2004

21. Karl Dias. Mark Ramacher. Uri Shaft. Venkateshwaran Venkataramani, Graham Wood Automatic Performance Diagnosis and Tuning in Oracle 2005

22. Martin Powley. Benoit Using Reflection to Introduce Self-Tuning Technology into DBMSs 2004

23. Oracle Oracle 8i with Oracle Fail Safe 3.0 2000

24. Oracle Corporation Oracle Database 10g Release 2:The Self-Managing Database 2005

25. OSCAR5 0项目开发组 OSCAR产品白皮书 2003

26. S PADMANABHAN. B BHATTACHARJEE. T MALKEMUS. L. CRANSTON, M. HURAS Multi-Dimensional Clustering:A New Data Layout Scheme in DB2 2003

27. TPC Benchmark C 2005

28. Weikum. Moenkeberg. Hasse Self-tuning Database Technology and Information Services:from Wishful Thinking to Viable Engineering 2002

29. D C ZILIO. J RAO. S LIGHTSTONE. G.M.LOHMAN, A.STORM, C.GARCIA-ARELLANO, S.FADDEN DB2 Design Advisor:Integrated Automatic Physical Database Design 2004

30. 孟小峰. 周龙骧. 王珊 数据库技术发展趋势[期刊论文]-软件学报 2004(12)

31. 郑宗汉. 郑晓明 算法设计与分析 2005

相似文献(10条)

1. 学位论文 王雪岩 辽宁通信物资管理系统技术方案设计与实现 2005

本系统的建设是在中国网通进行体制改革后的背景下，为了更好地管理辽宁通信各个管理点的物资库存情况，为了提高库存管理的工作效率，避免人力、物力的重复消耗，也为了完整地对库存管理过程的业务进行系统的跟踪记录而提出的采用电子化管理的设想与初步的方案。本着进、销、存管理系统引用了现代企业管理的新概念，保留了ERP的设计思想，以进销存为主线，展开业务需求，以库存为基础，采购为前提，缩短了信息流程，增强了企业适应市场的变化能力。

通信企业物资管理是一项全新的工作，具有系统性、复杂性和长期性。不仅涉及的部门多、内容多、关联环节多，工作量也大，而且也涉及原有管理观念的转变和管理职能的调整。物资库存管理是物流管理系统中的一个重要环节。

所含内容有物资计划采购管理、价格管理、供应商管理、，合同管理、物资运输管理、物资库存的管理、物资财务管理、客户关系管理、信息系统的管理、考核与监督的管理等。

通过系统建设及应用，实现全省通信物资在时间，空间上的优化配置，最大限度地创造时间效益和空间效益，提高生产部门服务水平，提高企业竞争力，提高整个物资管理系统的整体功能和效率、提高服务目标、快速，及时，准确目标、节约目标、规范化，现代化目标、调剂，调节目标。

本系统定位为：与其他管理系统有一定业务及数据关联的帐务式库存管理系统。

为实现前述各项功能和要求，系统架构设计为当前业界的主流实现方案：使用Java语言开发，基于JAVA技术的B/S三层结构，系统建立于web平台之上，所有的功能在客户端在浏览器中进行访问，业务逻辑在应用服务器上实现，各种数据存储于数据库中。

系统所有的界面表现风格和操作方法统一，便于使用者理解系统功能。

业务逻辑的组件化设计，可提高系统配置的灵活性和代码的重用性，各模块功能独立，耦合性低，易于扩充功能或在其基础上开发新的应用。

系统提供了充分的维护功能，简洁的定制接口使其能够适应随时间变迁而出现的各种应用需要，为动态变化的系统提供了稳定的基础平台，在其基础上发布各种新的功能易于实现，可以扩展、嵌入各种类型的应用，可集成目前和未来的系统，满足系统扩展的需求。

本文主要由四部分组成。第一部分介绍了辽宁通信公司的物资管理现状及系统开发的必要性、开发的原则、开发的目标和系统的技术特点。第二部分介绍了该系统的开发工具及开发环境和数据库的设计，包括编码设计、数据库的安全性分析以及系统中的数据备份。第三部分对该物资管理系统的各个模块的功能及设计方法作了详细介绍，并以收料为例详细给出了收料的处理流程和系统中实现的界面。最后一部分给出了系统的测试结论，表明该物资管理系统设计完好，已基本上满足了辽宁通信的物资管理需要。

2. 会议论文 刘玉荣. 李绪志. 踪念霞 "神舟"号飞船有效载荷监控管理系统归档数据库的设计 2002

本文以"神舟"号飞船有效载荷监控管理系统运行特点为基础,介绍了"神舟"号飞船有效载荷监控管理系统归档数据库的设计特点."神舟"号飞船有效载荷监控管理系统归档数据库具有不同于一般数据库应用系统的三层结构特点,对于归档数据的管理采用以数据文件为单元的管理方法,并通过数据库的并行服务技术保证了系统的可靠性.上述特点是"神舟"号飞船有效载荷监控管理系统能够成功支持"神舟"一号、"神舟"二号、"神舟"三号飞船飞行任务的基础,也是支持后续飞船飞行任务的重要保证.

3. 学位论文 王旭华 基于B/S结构的学生成绩管理系统的设计与实现 2007

Active Server Pages(ASP)是一套微软开发的服务器端脚本环境，使用它可以创建和运行动态、交互的 Web 服务器应用程序。本文的基础就是利用ASP技术的内置对象与 Active X 组件的对象ADO与Web数据库结合，开发出一套基于 B/S 结构的教务管理系统中子系统学生成绩管理系统。

首先，本文探讨了数据库运行的体系结构，介绍了浏览器/服务器模式的工作方式及特点；比较了主机/终端、客户机/服务器和浏览器/服务器等三种模式各自的优缺点。比较结果表明，浏览器/服务器模式继承了客户机/服务器模式的优点并克服了它的缺点，不受操作系统和硬件的制约，比较容易实现不同网络间的连接与应用升级。B/S结构是目前广泛使用的，因此，我们开发的学生成绩管理系统采用了浏览器/服务器模式。

其次，本文对Web数据库的访问方法进行了讨论。有提供中间件的方法两种：CGI和API。其中，CGI效率低，速度慢；API 虽然克服了CGI的缺点，但兼容性差和开发难度大。用 Java 技术虽然可以通过 JDBC 技术来访问数据库，但目前JDBC标准尚不完善、安全性也待提高。PHP技术稳定性好，安全性高，但由于它对不同数据库操作所使用的函数不同，当数据库发生变化时，改动非常大，并且其安装过程很繁琐。ASP技术是一种比较容易实现的功能强大且高效的数据库访问技术，因此，我们在开发学生成绩管理系统的过程中采用了ASP技术来访问学生成绩数据库。

本文重点讨论了利用ASP技术的内置对象、ActiveX服务器组件和ADO对象来实现对学生成绩数据库的访问方法，给出了设计细节和具体步骤和方法。在第六章还分析了学生成绩管理系统的实现细节。得出的结论是：用ASP+ADO访问服务器端的数据库的实质就是利用ADO对象访问数据源。但是这种方案也有它的不足之处：由于ADO是以一个本地数据访问组件，而ASP+ADO方案中，ADO在Web服务器端使用，因此该方法只能对服务器端数据库进行访问，而当数据库服务器与Web服务器不在同一计算机上时，这种方案就行不通；再者由于用户每次查询都必须连接服务器，然后等待回应，将占用较多的系统资源，产生传输需求的矛盾。

基于B/S结构的学生成绩管理系统就是利用ASP技术和 Web数据库相结合实现的，它具有查询、添加、更新、删除等功能。它的成功实现对提高高校的教务管理水平具有非常重要的意义。

4. 期刊论文 赵高丽. 杨斌. 宋军平. 蒋合领 高校考务管理系统数据库的设计与优化 -河南科技学院学报（自然科学版） 2008,36(3)

本文分析了考务管理系统在实现时数据库的设计方法,着重讨论了如何实现对考务管理系统数据库的设计与优化,从而提高了考务管理系统在考务方面的方便性、准备性、及时性.

5. 学位论文 朱国锋 基于.NET的多数据库供用电合同管理系统的设计及实现 2007

供用电合同管理系统是依照《中华人民共和国电力法》和省电力公司《供用电合同示范文本》开发的，能快捷、规范地起草合同，按照合同管理的流程进行会签、签订、执行管理等，并提供有效的数据查询、统计分析等管理功能的信息管理系统。随着国家电网和各电力公司业务和工作流程的不断规范、电力法律、法规的不断健全，对于供用电合同管理提出了更高的要求，已有的供用电合同管理系统都是基于C/S模式开发的，而且都只支持一种数据库，已不能适应形势发展的需要。迫切需要一个能兼容多种数据库，体系结构灵活，可扩展性好，便于部署和维护的，供用电合同管理系统。

随着B/S模式系统的优点的凸现及Web SERVICE的推广应用已经逐渐成熟，针对上述问题，本文提出了一种基于. NET平台的供用电合同管理系统设计方案。该系统采用B/S和C/S结构相结合的方式，结合三层结构客户端系统模式，运用ASP. NET、C#、ADO. NET、SOA架构技术，实现供用电合同的跨地区会签、签订、执行和管理，以及数据查询、统计分析、管理等功能，同时能够兼容Oracle9i和Sybase Ase 12.5两种数据库。

本文首先分析了该信息系统建设的相关背景，建设的必要性、意义和所能带来的社会、经济效益。接着对该系统开发过程中所用到的一些关键技术进行了详细的介绍。探讨了微软开发环境下比较通用的三层结构模型的划分和设计模式的应用，对系统需求分析与架构设计进行了详细的论述，对系统中最关键的数据库访问层的分析与设计进行了重点讨论。

本文对新形势下开发电力系统供用电合同管理系统提出了新的解决方案，为基于Web技术的多层结构的网络数据库的运用和开发进行了有益的探索，希望本文所进行的研究与实践可以为其他信息系统的建设提供一定的参考价值。

6.期刊论文 何桢.卢晋.刘晓亮.何曙光.HE Zhen.LU Jin.LIU Xiao-liang.HE Shu-guang 集成质量管理系统IQMS2.0的设计与开发 -工业工程2007,10(6)

回顾了国内外对集成质量管理系统的研究成果,分析了目前仍然存在的问题.在介绍集成质量管理系统的相关定义的基础上,简述了参与开发的集成质量管理系统IQMS2.0的优点,重点阐述了该系统数据库独特的设计思路、设计结构和实现方法,详细描述了该系统基于底层数据库所实现的功能模块.

7.学位论文 韩九曦 全球矿产资源信息数据库及应用管理系统 2005

在对国内外全球矿产资源数据库进行系统调研的基础上，作者确定了“全球矿产资源信息数据库及应用管理系统”(GMRIDAMS)的开发思路和框架结构。该系统的框架结构是纵向上划分出全球层次、大洲层次、国家层次和重要成矿带层次等四个层次，横向上划分出地理信息、地质信息、矿产信息、矿业信息以及地质调查工作程度信息等五种类型的信息。按照系统开发思路和框架结构的要求，作者收集和整理了世界各国十二万余条矿产地记录和全球2300多个矿业公司的基本信息以及全球层次、大洲层次和国家层次的地理信息与地质信息。在此基础上，作者采用MicrosoftAccess2002数据库管理系统，建立了全球矿产资源信息数据库，并采用VisualBasic可视化编程语言与美国ESRI公司的MapObject控件开发了包含五个类模块和三个公共模块的全球矿产资源信息数据库应用系统。

通过上述工作，论文取得了以下主要成果：1.建立了我国的全球矿产资源信息数据库。2.开发了全球矿产资源信息数据库应用系统。通过该应用系统提供的模糊查询、属性查询、区域查询、缓冲区查询等多种查询检索方式，不仅可以对全球矿产资源信息数据库系统中的各种信息进行查询和检索，而且可以以图片、文件或纸张打印的方式输出查询结果。

3.以TrueType字体构建了特殊符号库，实现了对168种矿产符号的图面表示。同时，参考国际地层委员会制订的国际地层标准，提出了全球、大洲、国家、重要成矿带层次的地层编码和图例标准。

8.期刊论文 许惠平.覃如府.叶娜.欧少佳.XU Hui-ping.QIN Ru-fu.YE Na.OU Shao-jia 中国岩石圈三维结构数据库总库管理系统 -中国地质2006,33(4)

中国岩石圈三维结构数据库总库管理系统是中国岩石圈三维结构数据库的管理和服务中心.在中国岩石圈三维结构数据库需求分析基础上,完成了系统总体设计,制订各项技术标准;开发设计一个元数据编辑器和元数据浏览器;应用统一建模语言UML设计和优化了数据库结构;采用ESRI公司Geodatabase的数据模型,利用计算机辅助设计CASE工具设计中国岩石圈三维结构数据库.在ArcObjects和MapObjects的基础上,分别建立了功能齐全、操作简便的两套组件式总库管理系统,分别适合专业和非地理信息系统专业人员.

9.期刊论文 戴雪蕾 药业营销管理系统的数据库性能优化 -科技信息2009,""(34)

结合对制药企业营销管理系统数据库的分析和试验,本文从以下几个方面探讨了基于SQL SEILVER 2000的数据库性能优化技术:[1]数据库服务器的性能优化;[2]数据库物理设计的优化;[3]数据库逻辑设计的优化.

10.会议论文 梁国坚.冉桂平 地面气象站气象观测资料数据库及管理系统 2000

应用Microsoft Visual Foxpro6.0(VFP6.0企业版),关系数据库理论,SQL数据查询语言,为地面气象站气象观测资料建立了一个C/X方式的数据库及其管理信息系统,该系统充分利用了32位微处理器强大的32位数据处理性能,多种可视化编程工具.面向对象编程,功能强、直观易用、支持客户机、服务器结构的种种优势.数据库尽可能存储该站自建站起的几乎所有气象资料,信息管理系统由管理员负责管理,数据库管理员负责管理整个数据库:弯路结构的修改、系统的维护、用户口令与权限的分配等;网络用户和本地用户通过管理员授权方式,可取得对不同数据的访问、查询、统计等权;数据库的数据输入维护、增、删、改由数据输入员负责.数据库服务器包括数据库、帮助库、备份库等,提供了与其它数据库的远程连接访问功能,并预留下与外部程序的接口,以适应网络时代的特点.