中图分类号：TP392　　文献标志码：B　　文章编号：2095-641X(2016)04-0146-08

数据库 SQL 审查与性能优化技术研究与应用

杭聪，黄连月，黄鑫
（广西电网有限责任公司 信息中心，广西 南宁 530023）

摘要：对大型企业级管理信息系统后台数据库执行 SQL 审查（SQL Review），在表设计、索引设计、SQL 编码等方面搜集潜在的性能隐患及缺陷，可为后续开展数据库深层次性能优化工作理清思路、明确方向。文章通过提取典型的 SQL 审查对象，制定审查规则、赋予权重，并建立评分机制，形成了一整套 SQL 审查指标体系。权衡优化成本与收益，重点运用面向表索引和 SQL 优化的数据库性能优化技术，以相对低的成本使得企业管理信息系统性能有质的改善，显著提升响应速度。

关键词：数据库；SQL 审查；索引；性能优化

0　引言

“大集中”部署模式的企业级管理信息系统往往承载数以千计或万计的用户负载，海量数据集中存储，集中处理“井喷式”的用户请求，这些系统普遍面临以下问题：

1）历史及实时数据量非常大，后台处理能力受多种客观因素牵制，效率低，响应速度慢；

2）系统难以支持多并发、大结果集的数据查询，耗时往往超出用户容忍度；

3）同时支撑省级、市级和县级单位访问，用户覆盖面广，业务繁忙时段系统响应缓慢，极大地降低了用户满意度。

信息系统性能优化是解决上述问题的关键，同时也是知识密集、综合性强的工作，需立足于系统工程的角度，从硬件（服务器、存储系统、网络、安全系统）和软件（操作系统、中间件、数据库、应用程序）全方位开展优化[1]。但优化措施的实施难易程度通常与性能改善程度并不成正比，如改善硬件配置对系统性能的提升可能不如优化应用软件代码明显。而源代码优化则因涉及程序逻辑改写，势必付出较高的时间成本与人力成本。

折衷考虑，制定性能优化策略时应将重点放在平台类软件的优化上，如操作系统、中间件和数据库等。其中，数据库作为企业 IT 基础设施的核心部件之一，与网络、操作系统、存储设备皆有密切关联。数据库依照某种数据模型组织、存储及管理业务数据，其性能优化已成为保证企业 IT 基础平台健康运营的关键技术之一，对提升信息系统整体性能起到至关重要的作用。

1　数据库 SQL 审查

针对未上线的信息系统数据库执行 SQL 审查，目的是提前发现数据结构设计和 SQL 语句存在的导致数据库性能降低的隐患及缺陷，本质上属于一种性能评估方法。审查方式有人工核查和自动化工具核查，审查结果能为后续性能优化工作提供直接依据。由于性能调优成本随软件生命周期的推进呈几何增加趋势，因此审查活动介入的阶段越早越有利，应尽量避免系统投运后再为调优付出高昂的代价。

1.1　基于定量分析的 SQL 审查指标体系

本文汇集了影响数据库性能的主要因素作为审查对象（即指标），为每一个对象制定审查规则，并根据其对性能的影响程度赋予权重，建立评分制度，最后形成了一整套 SQL 审查指标体系。该体系可根据

数据库领域专家的经验灵活地进行完善，如在数据建模阶段开展审查可适当增加表设计类的审查对象和规则，在系统开发结束后的测试阶段则增加索引类和 SQL 类的审查对象和规则。数据库 SQL 审查指标集如图 1 所示。

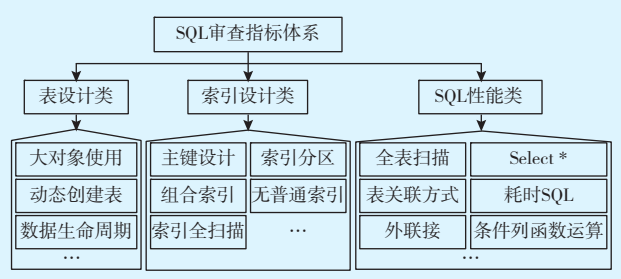

图 1 数据库 SQL 审查指标集

Fig.1 Database SQL review index set

数据库 SQL 审查规则及评分规则见表 1 所列。数据库 SQL 审查评定等级见表 2 所列。从权重分配看，影响数据库性能的主要因素来自 SQL 语句和表索引，这 2 个方面作为审查的重点，必然是后续性能调优工作的重点。数据库 SQL 审查评分满分为 100 分，每个指标分配相应分值和权重，每发现一个违反审查规则的对象就相应扣分，扣减后的分值作为该指标得分，每项指标得分再乘以权重，最后求和。总得分可表示为：总分 = ∑ (单个指标得分 × 指标权重)，其中单个指标得分 = 指标分值 – 单项扣分 × 违规数量。

1.2 应用效果

为了验证 SQL 审查指标体系的应用效果，笔者在广西电网年度应用系统性能评估优化专项工作中

表 1 数据库 SQL 审查规则及评分规则

Tab.1 Database SQL review rules and scoring rules

指标类别	类别权重	指标名称	指标权重	审查规则	分值	评分方法	审查过程输出物
表设计	20%	大对象使用	30%	大对象类型的数据应存储在独立表空间或文件系统	6 分	每个使用大对象数据类型，但未分配独立表空间或文件系统的表扣 0.1 分【60 张表以内】	大对象类型的表清单
		动态创建表	40%	应避免大量动态实体表创建导致数据字典过大	8 分	每个动态表扣 0.01 分【800 张表以内】	动态实体表清单
		数据生命周期	30%	数据增长比较快的表应制定转储策略，如历史数据分离	6 分	每个未设计数据转储策略的大表扣 0.1 分【60 张表以内】	数据增长比较快的表清单
索引设计	40%	主键设计	20%	能否为表关联查询的执行计划提供有效合理的索引	8 分	每个无主键表且无唯一索引的表，扣 0.05 分【160 张表以内】	无主键表清单
		组合索引	20%	能否以最少的索引数量满足尽量多的 SQL 查询需求	8 分	3 个以上索引的表，组合索引个数 / 索引总数 < 40%，每个表扣 0.1 分【80 张表以内】	组合索引个数 / 索引总数 <60% 的表及索引清单
		索引全扫描	20%	应避免索引全扫描导致 SQL 性能低下	8 分	每个发生索引全扫描的 SQL 查询，扣 0.05 分【160 条 SQL 语句以内】	全扫描的索引 +SQL(执行计划)
		无普通索引	30%	应避免非主键字段缺少普通索引导致 SQL 查询发生全表扫描	12 分	每个只有主键无普通索引的表扣 0.1 分【120 张表以内】	无普通索引的表清单
		索引分区	10%	对表进行分区后是否对索引做合理分区	4 分	每出现一张分区了但索引未进行分区的表扣 0.1 分【40 张分区表以内】	未分区索引的表和索引清单
SQL 性能	40%	全表扫描	20%	避免对大表进行全表扫描	8 分	每张大小超过 100 MB 的表，每存在一条执行全表扫描的 SQL 语句扣 0.05 分【160 条 SQL 语句以内】	全表扫描的 SQL 语句清单及执行计划
		条件列函数运算	20%	避免条件列因参与函数运算导致索引无法使用	8 分	每一条存在条件列函数运算的 SQL 语句扣 0.01 分【800 条 SQL 以内】	SQL 语句清单及执行计划
		表关联方式	10%	避免采用 Sort Merge (合并排序) 的表联接操作	4 分	每个产生 Sort Merge (合并排序) 表联接的 SQL 语句 0.1 分【40 条 SQL 以内】	SQl 语句清单及执行计划
		外联接	10%	避免由于数据模型问题，导致开发误用 distinct	4 分	外联接 SQL 语句中，每个存在 distinct 的 SQL 语句扣 0.1 分【40 条 SQL 以内】	SQl 语句清单及执行计划
		select *	20%	避免使用 select* 子句导致索引不生效，同时减少网络开销	8 分	每一条使用 select * 的 SQL 语句扣 0.02 分【400 条 SQL 以内】	使用 select * 子句的 SQl 语句清单
		耗时的 SQL	20%	避免 Top SQL 占用系统大部分的资源，导致系统整体过慢	8 分	每一条执行耗时超过 10 s 以上的 SQL 语句扣 0.1 分【80 条 SQL 以内】	Top SQl 语句清单

表 2 数据库 SQL 审查评定等级

Tab.2 Database SQL review rating

序号	审查得分	评定等级	等级说明
1	≤ 60 分	差	业务数据库在表、索引设计和 SQL 执行性能上整体表现较差，存在很多严重的性能问题隐患；随着业务数据量的增加，性能问题将日益明显；需要根据审计输出列表和审计报告中的建议对扣分项进行彻底修改，尽早排除隐患
2	60~69 分	及格	业务数据库在表、索引设计和 SQL 执行性能上整体表现及格，存在较多严重的性能问题隐患；随着业务数据量的增加，性能问题将日益明显；需要根据审计输出列表和审计报告中的建议对扣分项进行逐项修改，尽早排除隐患
3	70~79 分	中等	业务数据库在表、索引设计和 SQL 执行性能上整体表现中等，但有不少的审计项存在较严重的性能问题隐患；需要根据审计输出列表和审计报告中的建议对扣分项进行逐项修改，提升系统稳定性
4	80~89 分	良好	业务数据库在表、索引设计和 SQL 执行性能上整体表现良好，但仍有些审计项存在较严重的性能问题隐患；需要根据审计输出列表和审计报告中的建议对扣分项进行逐项修改，提升系统稳定性
5	≥ 90 分	优秀	业务数据库在表、索引设计和 SQL 执行性能上整体表现优秀，只有个别审计项存在性能问题隐患；需要根据审计输出列表和审计报告中的建议对扣分项进行修改，提升系统稳定性

选取省级大集中部署的某管理信息系统开展上线前 SQL 审查。该系统数据库被审查的对象共计 75 138 个，其中违反审查规则的对象有 768 个，审查总分为 71.19 分，表明数据库性能处于中等级别，存在一些严重的性能隐患及缺陷影响系统运行的稳定性。某企业管理信息系统 SQL 审查结果见表 3 所列。

对违规对象进行分析和统计，发现扣分较多的审查对象来自索引设计、SQL 语句结构不合理、SQL 语句执行计划不正确等。为了解决这些性能缺陷及隐患，选取核心或热点业务表的索引进行优化，对高消耗的 SQL 语句进行调优。

本次 SQL 审查取得了以下效果。

1）改变了当前单纯靠压力测试判断系统性能是否满足上线条件的现状。压力测试通过模拟用户负载和典型业务操作向系统施压，模拟程度有限，测试场景与投运场景必然存在相当差距，性能缺陷或隐患未能暴露出来。而 SQL 审查发现性能隐患和缺陷的几率远远超过模拟压力测试，从定量角度评估数据库性能表现，评分高于基准才准予系统上线，这样的衡量准则更为可靠。

2）利于有效、精准地解决应用性能问题。应用程序包括高级语言编写的代码和 SQL 代码，中间件访

表 3 某企业管理信息系统 SQL 审查结果

Tab.3 SQL review results of an enterprise management information system

指标类别	类别权重	指标名称	指标权重	分值	审查规则	审查数量	违规数量	单项扣分	审查得分
表设计	20%	大对象使用	30%	6 分	大对象类型的数据应存储在独立表空间或文件系统	168	2	0.1	5.8
		动态创建表	40%	8 分	应避免大量动态实体表创建导致数据字典过大	168	0	0.01	8
		数据生命周期	30%	6 分	数据增长比较快的表应制定转储策略，如历史数据分离	168	1	0.1	5.9
索引设计	40%	主键设计	20%	8 分	能否为表关联查询的执行计划提供有效合理的索引	168	32	0.05	6.4
		组合索引	20%	8 分	能否以最少的索引数量满足尽量多的 SQL 查询需求	168	3	0.1	7.7
		索引全扫描	20%	8 分	应避免索引全扫描导致 SQL 性能低下	12 327	49	0.05	5.55
		无普通索引	30%	12 分	应避免非主键字段缺少普通索引导致 SQL 查询发生全表扫描	168	116	0.1	0.4
		索引分区	10%	4 分	对表进行分区后是否对索引做合理分区	168	0	0.1	4
SQL 性能	40%	全表扫描	20%	8 分	避免对大表进行全表扫描	12 327	89	0.05	3.55
		条件列函数运算	20%	8 分	避免条件列因参与函数运算导致索引无法使用	12 327	237	0.01	5.63
		表关联方式	10%	4 分	避免采用 Sort Merge（合并排序）的表联接操作	12 327	10	0.1	3
		外联接	10%	4 分	避免由于数据模型问题，导致开发误用 distinct	12 327	0	0.1	4
		select *	20%	8 分	避免使用 select* 子句导致索引不生效，同时减少网络开销	12 327	227	0.02	3.46
		耗时的 SQL	20%	8 分	避免 Top SQL 占用系统大部分的资源，导致系统整体过慢	取决于 V$Sqlarea 中的 SQL 记录数	2	0.1	7.8
总分				100	综合得分				71.19

问数据库的所有操作均通过 SQL 代码来实现。因此索引类、SQL 类审查结果能精确定位相当一部分应用性能问题，为后续优化工作提供清晰的方向与思路。

3）降低了系统上线风险。相较于黑盒级性能测试方法，白盒级的 SQL 代码审查能发现 80% 潜藏的应用性能问题，经过优化能大幅降低系统上线后出现性能故障的几率。

2 数据库性能优化解决方案

基于系统工程的理念，具有普适性的数据库性能优化思维是围绕数据库生命周期的多个阶段，从设计、开发、安装部署到运行维护阶段，持续地探寻性能问题及解决方案。因此，根据优化工作介入的时间大致划分 3 个阶段：一是设计与开发阶段，主要对数据库逻辑和物理结构设计进行优化，使其在满足业务需求的前提下，性能达到最佳且开销最小；二是安装部署阶段，主要对数据库管理系统自身的参数、操作系统资源分配及内核参数进行优化，为应用提供良好的支撑平台；三是投运阶段，优化的范畴涵盖数据库、操作系统、应用、服务器、网络、统一存储等。根据业界统计的软件生命周期调优成本与调优收益变化曲线，发现调优成本通常随着软件生命周期的推进呈上升趋势，调优收益却随之逐渐降低并趋于零[2]。因此，数据库性能优化的整体解决方案是：在设计与开发阶段开展数据库 SQL 审查，并在此阶段完成表设计、索引设计和 SQL 语句等方面的性能缺陷及隐患整改；在数据库安装配置阶段，对主要性能参数进行优化；在数据库投运阶段，利用先进的数据库性能监控工具对其运行情况进行多维度监控与分析，分析维度包括数据库碎片、计算资源使用情况、等待事件、锁分析、表空间碎片、缓存命中率、内存管理、文件 I/O、SQL 执行计划的效率等，并关注业务高峰时段的数据库日志，随后针对表现异常的方面制定优化措施。常见的数据库性能优化技术的效果比重如图 2 所示。

由图 2 可知，设计开发阶段执行数据建模时，对数据模型进行优化的效果最明显，对应用性能的改善程度最大；其次是表索引及 SQL 语句优化技术；然后是改进 SQL 语句的执行计划（从数据库管理系统层面）；收益最小的则是参数优化[3]。从职责界定的角度来说，数据库设计开发阶段和安装部署阶段的性能优化工作应由信息建设部门主导，数据库投运阶段的性能优化由信息运维部门主导。本文将从信息运维的视角阐述数据库性能优化技术——索引优化及 SQL 优化。

图 2 常见的数据库性能优化技术的效果比重

Fig.2 Common database optimization techniques performance proportion

2.1 从战略的角度设计表索引

2.1.1 索引设计现状

纵观企业级大型管理信息系统数据库，索引创建的普遍现状如下：①随需而建，即发现某条 SQL 语句执行效率低、执行计划存在全表扫描时，通过查看 where 子句中的列、列上使用的运算符、列值的离散度等信息直接创建索引；②盲目而建，当系统功能增加或变更需要新增或修改 SQL 语句时，首选是新增索引，而未考虑利用现有索引，或者未定期删除不再使用的索引；③索引列的顺序设置欠佳，开发人员单纯通过 SQL 语句中列字段的离散度来决定索引列顺序，对性能产生了很大影响。

上述现状对应存在的劣势主要体现为以下 3 点：①随需而建的索引仅能满足单条 SQL 语句的使用，同张表的其他 SQL 语句在执行过程中可能缺少适宜的索引；②盲目而建的索引虽然能顾及某张表每条 SQL 语句的效率，但大量索引创建后可能存在重复、冗余，或是某些字段在不同索引里重复度高的现象，这不仅会导致索引对象占用庞大的存储空间，也会使数据库在制定 SQL 执行计划时经常选择错误的索引，从而严重影响执行效率；③未能基于所有的应用场景通盘考虑索引列的顺序，导致索引的列序设置不合理，比如未选择离散度高的列作为前导列，SQL 语句在执行过程中会读取大量无效数据块，磁盘吞

吐量高，易出现性能瓶颈。

为了解决这些现状，通过大量数据库优化的实践工作，建立了一套关系型数据库索引的设计方法，用于表索引设计及优化。

2.1.2 战略索引设计技术

战略索引设计，即从全局的角度设计索引，不是单纯为了某条 SQL 语句的效率去设计索引，而是通盘考虑一张表相关的所有 SQL 语句的性能需求，制定最佳的索引设计方案，力求提高 SQL 语句的整体执行效率，降低总体资源开销[4]。战略索引设计包括以下 3 个步骤。

1）核心表的精确定位。应用系统的数据库表数量多，小型应用通常有上百张，大型应用则有上千张。根据“二八原则”——选取百分之二十的表，解决百分之八十的应用性能问题，设计战略索引之前需准确定位哪些是核心业务表。可通过 3 种维度来筛选核心表：①业务的重要程度，表对应的功能模块用以完成客户的核心业务；②业务密集程度，表相关的 SQL 语句数量越多，说明该表汇集的业务操作越多；③表的数据量，数据量越大，说明表读写越频繁。

以某企业级管理信息系统为例，定位了 30 余张核心表，且从中根据重要程度区分为高、中、低作为优化的顺序。待优化的核心业务表如图 3 所示。

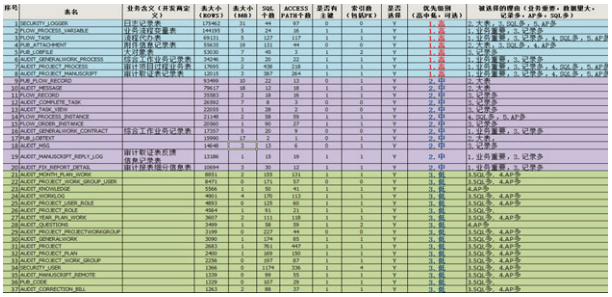

图 3 待优化的核心业务表

Fig.3 Core business list to be optimized

2）SQL 语句搜集与解析。核心业务表相关 SQL 语句的搜集与解析指的是搜集所有数据库对象和数据库内存缓存区中出现的 SQL 语句，继而筛选出与核心表相关的 SQL 语句并进行解析，最终获得核心业务表的访问路径。SQL 语句与访问路径的关系可能为 1:1 或 1:*M*，即一条语句可能对应一条访问路径，亦或对应多条访问路径。

3）战略索引设计。战略索引设计的首要原则是以最少的索引满足尽可能多的访问路径[5]。基于每张核心业务表对 SQL 语句进行分组，使用自动化分析工具对每组 SQL 语句的 where 条件子句进行解析，解析结果为访问路径，采用列名 + 运算符缩写来标记。在解析过程中，自动化分析工具能自动合并相同的访问路径，并将运算符替换成缩写符号。具体替换规则为：“=”替换为（=）；IN 替换为（IN）；BETWEEN 替换为（B）；<> 也替换为（B）；LIKE 替换为（L）；JOIN 替换为（J）。

以某企业级管理信息系统中的一张核心业务表 A 为例，对表 A 相关的 41 条 SQL 语句进行解析，获取共计 11 条访问路径，见表 4 所列。

表 4 访问路径列表

Tab.4 Access path list

序号	访问路径
1	CUST_NO(=), ARR_DT(B)
2	CUST_NO(=), PRICE_STD(=)
3	PRICE_STD(=), BNK_CD(=)
4	BNK_CD(=), SND_CD(=), ARR_DT(L)
5	PRICE_STD(=), ARR_DT(B)
6	BSE_CRD_NO(=),ACT_NO(=), PRICE_STD(=)
7	BSE_CRD_NO(=), ARR_DT(L)
8	PRICE_STD(=),BNK_CD(IN), RR_DT(L)
9	PRICE_STD[IN], BNK_CD(IN)
10	BNK_CD(=), ARR_DT(B)
11	SND_CD(=), BNK_CD(IN), ARR_DT(B)

基于以上 11 条访问路径，使用下列通用准则（见表 5）设计核心业务表的战略索引。

表 5 索引设计通用准则

Tab.5 General criteria for index design

准则	说明
是否经常被使用	列在不同访问路径中出现的次数
使用的运算符是否为“=”	列后面使用的运算符是否为“=”
哪个列具有更好的离散度	列中数据的分布，离散度 = 列值种类 / 总行数 ×100%，该结果值越大，说明列的离散度越好，反之表示离散度不好
经常基于哪些列字段来排序	众多 SQL 语句的 where 条件字句经常基于哪些列字段进行排序
哪个列将来会在业务中使用	将来业务变更或新增，那些列有可能被使用

设计过程分为 2 步，第 1 步为基于不同的列对访问路径进行分组，访问路径分组表见表 6 所列。

第 2 步为按分组完成战略索引设计，战略索引设计见表 7 所列。

表 6 访问路径分组表

Tab.6 Access path grouping table

序号	访问路径	分组基准列	分组
1	CUST_NO(=), ARR_DT(B)	CUST_NO	1
2	CUST_NO(=), PRICE_STD(=)		
2	CUST_NO(=), PRICE_STD(=)	PRICE_STD	2
3	PRICE_STD(=), BNK_CD		
5	PRICE_STD(=), ARR_DT(B)		
8	PRICE_STD(=), BNK_CD(IN), RR_DT(L)		
9	PRICE_STD[IN], BNK_CD(IN)		
6	BSE_CRD_NO, ACT_NO, PRICE_STD(=)		
10	BNK_CD, ARR_DT(B)	BNK_CD	3
4	BNK_CD, SND_CD, ARR_DT(L)		
3	PRICE_STD(=), BNK_CD		
8	PRICE_STD(=), BNK_CD(IN), RR_DT(L)		
9	PRICE_STD[IN], BNK_CD(IN)		
11	SND_CD, BNK_CD(IN), ARR_DT(B)		
7	BSE_CRD_NO, ARR_DT(L)	BSE_CRD_NO	4
6	BSE_CRD_NO, ACT_NO, PRICE_STD		

表 7 战略索引设计

Tab.7 Strategic index design

访问路径序号	分组基准列	索引结构
1,2	CUST_NO	CUST_NO PRICE_STD
2	PRICE_STD	PRICE_STD BNK_CD ARR_DT(B)
3		
5		
8		
9		
6		
10	BNK_CD	BNK_CD ARR_DT(L) SND_CD
4		
3		
8		
9		
11		
7	BSE_CRD_NO	BSE_CRD_NO
6		

从设计结果来看，一张拥有 40 条 SQL 语句和 11 条访问路径的核心表仅需创建 4 个索引，便可以满足其所有 SQL 语句的性能要求。

2.2 战略索引设计应用效果

战略索引设计的预期目的是大幅度降低数据库逻辑读写量和物理读写量（以数据块为单位），因此其应用效果可通过业务表使用战略索引后数据库产生的读写（I/O）量的缩减程度来体现。以核心业务表 A 为例，通过 Oracle EM 展现优化索引前后数据库产生的 I/O 总量（包括逻辑 IO 与物理 IO）变化，分别如图 3 和图 4 所示。

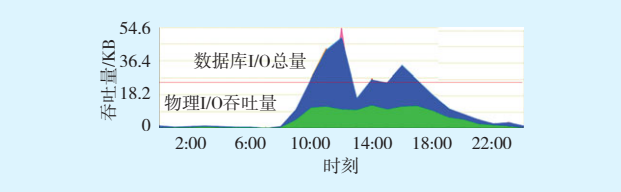

图 4 战略索引优化前数据库产生的 I/O 总量

Fig.4 Total amount of I/O generated by the database before strategic index optimization

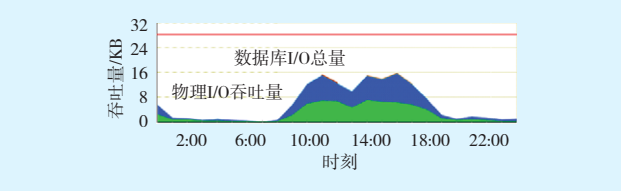

图 5 战略索引优化后数据库产生的 I/O 总量

Fig.5 Total amount of I/O generated by the database after strategic index optimization

由图 4 和图 5 可知，优化前数据库产生的 I/O 总量峰值达 54.6，优化后 I/O 总量的峰值降至 16。由此可见，采用战略索引优化方案能使数据库产生的 I/O 总量缩减三分之二。相对硬件升级方案，此优化技术的优势在于：

1）实施成本低，无需修改应用代码，极大地节省了人力成本与时间成本；

2）优化效果明显，持续时间长，在系统业务需求没有变更的前提下，能保证系统在 1~3 年内具有良好的性能表现。

3 SQL 优化技术

3.1 SQL 优化原理

SQL 语句是应用与数据库之间交互的唯一桥梁，其执行效率直接影响到系统响应时间及用户体验。SQL 语句的执行计划由数据库自身的优化器来制定，基于成本的优化器按照数学中排列组合原理对 SQL 语句中所包含的要素进行排列组合，并快速计算出各排列组合所消耗的资源和时间，最后从中

选择一个它认为消耗资源最少和时间最短的执行计划[6]。优化器是一个智能程度很高的计算机程序，它存在一定的局限性：当 SQL 语句不具备良好要素时，难以为其制定高效的执行计划。而影响执行计划效率的要素包括索引、统计信息、SQL 语法结构等[7]。

SQL 语句优化首先要判断哪些 SQL 要素存在优化空间。选择哪些优化点遵循的原则是观察 SQL 语句的执行计划，对比最初处理的数据行数、中间处理的行数及最终输出行数三者之间是否存在明显差异。若三者之间差距很明显，则说明在 SQL 语句执行过程中存在大量无效处理，通过人工干预能够去除这些无效部分，从而提高其执行效率。

在一个业务周期里（即包括业务高峰期和低峰期），可利用数据库性能监控工具筛选出执行耗时大于某基准值的 TOP SQL 语句列表，对其开展优化。具体优化步骤如下。

1）定位问题。详细分析执行计划，定位导致 SQL 语句执行时间长或数据库 I/O 负荷高的环节。

2）分析原因。分析 SQL 语句执行计划引起高消耗的具体原因，通常包括没有使用恰当的索引；使用无效索引导致读取了大量无效的数据块；没有选择恰当的表连接方式，执行了大量无效循环；表连接顺序不合适；过滤能力强的条件没有被优先执行；欠缺可选的索引，导致全表扫描等[8]。

3）提出解决方案。解决方案按照 SQL 优化的角度可分为 4 类：调整索引、改变表连接方式或顺序、调整 SQL 语法结构、改进执行计划。

4）测试效果。以 Oracle 数据库的表连接技术为例，可结合不同的应用场景，强制改变多个源表之间的连接方式，以获得执行的高效率。Oracle 数据库常见的表连接使用准则见表 8 所列。

表 8 Oracle 数据库常见的表连接使用准则

Tab.8 The rules of Oracle database table connection

序号	表连接方式	常用场景
1	排序合并连接	where 子句使用非等值运算符；多表关联字段均有索引
2	嵌套循环连接	驱动表比被驱动表小、被驱动表有唯一索引或离散度高的非唯一索引
3	哈希连接	where 子句使用等值运算符

完成同样功能的 SQL 语句，在语法结构上未必是单一的，可以有几种不同的写法，写法不同会导致 SQL 语句在执行效率上千差万别。在数据库性能优化的过程中，本文总结归纳了一些通用的 SQL 语句编写准则，依据这些准则改变 SQL 语句的语法结构，可适当改善 SQL 语句耗时、耗资源多的现象[9-12]。高效 SQL 语句编写准则见表 9 所列。

表 9 高效 SQL 语句编写准则

Tab.9 The programming rules of efficient SQL statement

序号	准则	说明
1	使用绑定变量	在 SQL 查询条件子句中使用变量而非常量，可减少 SQL 硬解析次数
2	共享 SQL 代码	相同的 SQL 语句应避免对象名称出现大小写不同、尽可能利用绑定变量统一使用同一个 SQL 语句，减少 SQL 硬解析次数
3	不要遗漏表之间的连接条件	进行表连接查询时若遗漏连接条件，会导致优化器进行笛卡尔积运算，对于大表来说极其耗费系统资源和时间
4	适当使用 commit	commit 所释放的资源而减少，执行 commit 命令后能释放相当一部分数据库资源
5	select 语句中尽量不要使用 * 符号	SQL 语句在解析的过程中 会将 * 号转换成表所有字段名，须通过查询数据字典完成，因此耗费更多时间
6	用“≥”运算符替代“＞”运算符	减少查询耗时
7	like 条件语句避免使用前缀百分号 %	以 % 为前缀的数据查询会导致全表扫描，应尽量避免，降低执行耗时
8	不要在索引列上使用函数	若 where 子句中的索引列是函数的一部分，优化器将无法使用索引而采用全表扫描，应尽量避免，降低执行耗时

3.2 SQL 优化的应用效果

以某企业级管理信息系统为例，采用以上优化策略对 36 条 SQL 语句开展优化，通过比较优化前后语句的耗时及其产生的数据库 I/O 吞吐量可知，耗时整体缩减了大约 92%，I/O 吞吐量整体降低了大约 93%，相当于执行效率整体提升了将近 12 倍，具体统计结果如图 6 所示。其中，单条 SQL 的执行效率最大提升达千倍，I/O 吞吐量最大降幅近乎 100%（图 6 红框选中行）。

4 结语

本文提出了一套基于定量分析的数据库性能审查指标体系，在系统投运前评估数据库性能，挖掘数据库在设计和编码方面潜藏的性能缺陷和隐患，并提出了基于系统工程理念的数据库性能优化思路，重点阐述了 2 种高收益、低投入的性能优化技术——战略索引设计与 SQL 语句优化。这些技术的应用效果表明，

	SQL_ID	优化前响应时间(s)	优化前IO	优化后响应时间(s)	优化后IO
2	bymmfr784zj7z	59.07	44202	0.59	1853
3	ggtsjppukyk93	18.49	45697	0.59	10151
4	cb60zjkhpxvd1	15.84	22643	0.77	4786
5	066rv9q52vay7	15.37	22700	0.79	4786
6	5dtt9gns5w3nj	14.04	661006	0.01	24
7	bxbtagc9natdn	12.77	21380	0.76	4790
8	2d761wdwudyx7	9.55	17723	0.76	4793
9	fth82cp86zr1y	9.21	202495	2.82	22757
10	fnru5dffjuhfm	9.14	20363	0.81	4787
11	0mzqragt1brg4	6.39	15718	0.77	4786
12	bxnsyqsghcf25	6.1	17896	0.82	4787
13	2zdhujduu3krv	5.96	47735	1.24	1521
14	5uaxv5xwd4uuw	3.18	40280	0.14	1151
15	51wsyk2ctdzc7	2.95	28116	0.79	10570
16	at1zhkxg204wj	2.71	13924	0.06	11002
17	298gz1akmc2zk	2.64	43337	0.58	3040
18	dw04m6t9at7pn	2.62	40734	0.59	1643
19	avp9pzuz2thc5	2.46	13689	0.32	1723
20	1wb90zbutb2vd	2.37	38275	0.63	1433
21	60ujcqzj0fs70	2.23	38544	0.31	981
22	453258abrqmfs	2.08	16072	1.04	3288
23	0yjyshbd6kbvx	2.07	142165	0	32
24	ggbh80mk9546d	1.91	18965	0.12	89
25	63n7xxgbbt5mc	1.76	41394	0.33	2430
26	65n0cc58w08vg	1.68	40512	1.19	3064
27	g9wt6szdrquty	1.67	43812	0.28	856
28	9jtbhctgxnmuu	1.65	34573	0.11	4493
29	4bvmz2aj6bm2p	1.59	50300	0.18	12532
30	dafta5x6n0m0f	1.54	4518	0.34	2649

图 6 SQL 语句优化前后的效果对比

Fig.6 The contrast of SQL statement optimization

与单纯提高硬件配置相比，除了节省高额硬件投资，还可从整体上显著地提升企业级管理信息系统性能，提升幅度可达 10 倍以上，并且相关方法论具有普适性，适用于不同的关系型数据库，具有很好的应用价值。

参考文献：

[1] 文平. Oracle数据库性能优化的艺术[M]. 北京: 机械工业出版社, 2012.

[2] 王继业, 程志华, 张宗华, 等. 大型管理信息系统数据库优化方法研究[J]. 电力信息与通信技术, 2015, 13(8): 1-4.
WANG Ji-ye, CHENG Zhi-hua, ZHANG Zong-hua, et al. Research on the database optimization method of large management information system[J]. Electric Power Information and Communication Technology, 2015, 13(8): 1-4.

[3] KUHN D, ALAPATI S R, PADFIELD B. Oracle索引技术[M]. 卢涛, 译. 北京: 人民邮电出版社, 2013.

[4] LAHDENMAKI T, LEACH M. 数据库索引设计与优化[M]. 曹怡倩, 赵建伟, 译. 北京: 电子工业出版社, 2015.

[5] 李华植. 海量数据库解决方案[M]. 北京: 电子工业出版社, 2010.

[6] 崔华. 基于Oracle的SQL优化[M]. 北京: 电子工业出版社, 2014.

[7] LEWIS J. 基于成本的Oracle优化法则[M]. 赵恒, 李政仪, 译. 北京: 清华大学出版社, 2007.

[8] 白鳝. Oracle优化日记[M]. 北京: 人民邮电出版社, 2010.

[9] 盖国强. Oracle DBA手记3: 数据库性能优化与原理解析[M]. 北京: 电子工业出版社, 2011.

[10] 杨小宁, 邰炜, 蔡忠林. 地市供电公司信息系统数据库优化实践[J]. 电力信息与通信技术, 2015, 13(5): 105-108.
YANG Xiao-ning, TAI Wei, CAI Zhong-lin. Optimization practice of information system database in local power supply company[J]. Electric Power Information and Communication Technology, 2015, 13(5): 105-108.

[11] 罗伟, 蒋苏湘, 周沿东, 等. 湖南电力营销系统数据库性能优化研究[J]. 电力信息与通信技术, 2014, 12(4): 30-34.
LUO Wei, JIANG Su-xiang, ZHOU Yan-dong, et al. Research on database performance optimization for Hunan power marketing system[J]. Electric Power Information and Communication Technology, 2014, 12(4): 30-34.

[12] 张云翔. 任务分发策略在Oracle数据库集群中的应用研究[J]. 电力信息与通信技术, 2015, 13(1): 90-94.
ZHANG Yun-xiang. Research on application of task distribution strategy in Oracle RAC[J]. Electric Power Information and Communication Technology, 2015, 13(1): 90-94.

编辑　邹海彬

收稿日期：2015-10-30

杭聪

作者简介：

杭聪（1980–），女，广西南宁人，高级工程师，研究方向为软件平台运维及性能优化；

黄连月（1975–），女，广西南宁人，高级工程师，从事电力信息运维管理工作；

黄鑫（1983–），男，广西南宁人，助理工程师，从事存储资源池运维工作。

Research and Application of Database SQL Review and Performance Optimization Technology

HANG Cong, *HUANG Lian-yue*, *HUANG Xin*

(Information Center, Guangxi Power Grid Co., Ltd., Nanning 530023, China)

Abstract: A SQL review for large enterprise management information systems' database can collect potential performance threats and bugs on table, index and SQL code, which can make a clear direction for the future work of deepening database's performance optimization. During the SQL review, we extract some typical targets weighted, establish review rules, build the scoring mechanism, Then, a set of SQL review index system is formed. After a tradeoff of costs and benefits, two kinds of database optimization technologies including database index tuning and SQL tuning are used to improve the performance and response speed of enterprise management information systems with low cost.

Key words: database; SQL review; index; performance optimization