数据库性能优化分享

----- 曹乃勋

分享目录

- 个人介绍
- 数据库优化
- 优化案例
- Q&A

个人介绍

10年数据库行业经验

具有丰富的数据库架构设计和运维经验

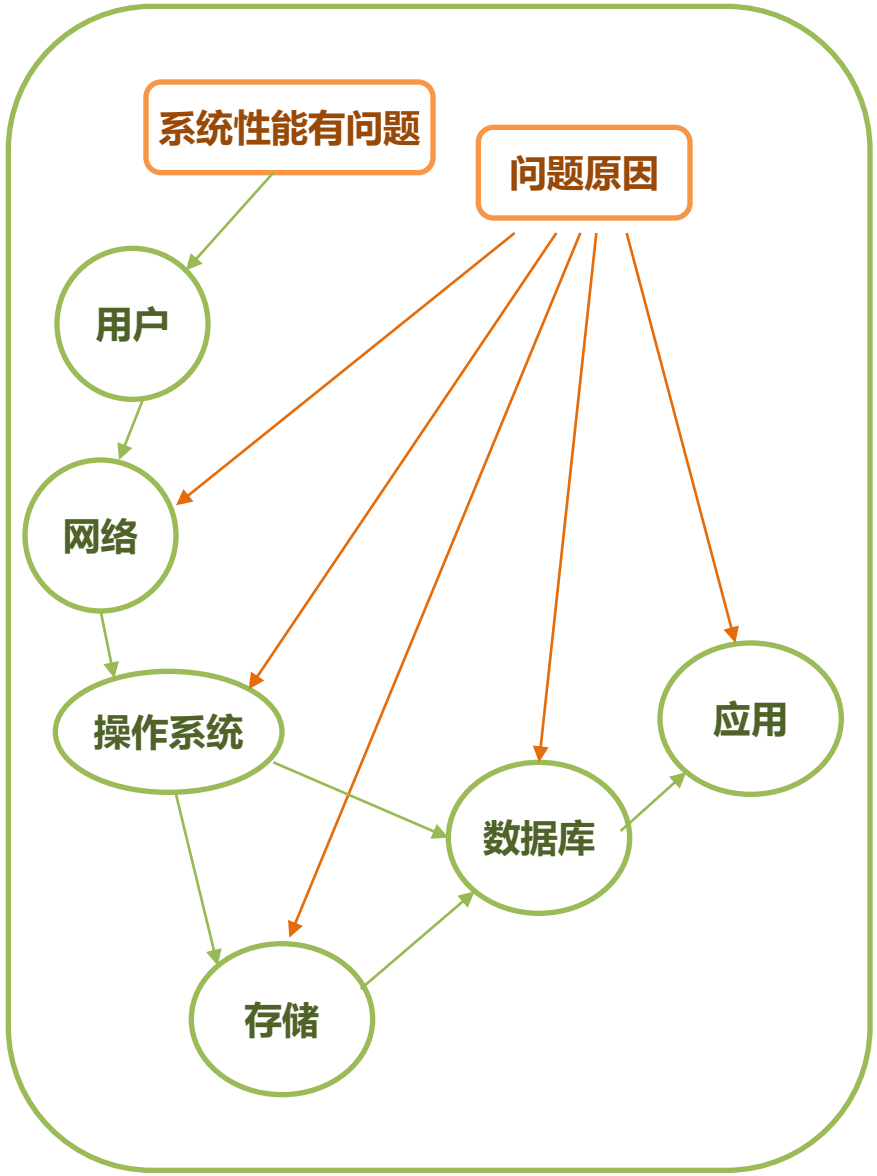
曾负责政府、通信、金融等多领域的项目建设和维护

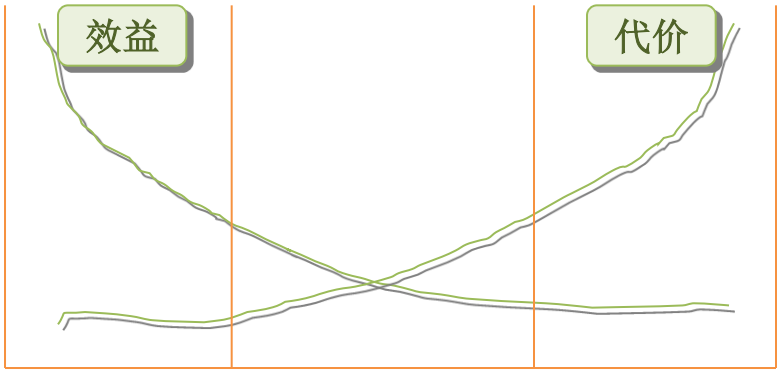
目前致力于运维标准化、自动化的改进和推动工作

Oracle优化的常见误区

- 通过调整参数达到优化目的
- 数据库性能问题与开发无关
- 开发阶段无需考虑性能问题
- 多表链接性能差
- Cpu利用率越低越好
- 大内存解决性能问题

正确的优化方法：从设计开始



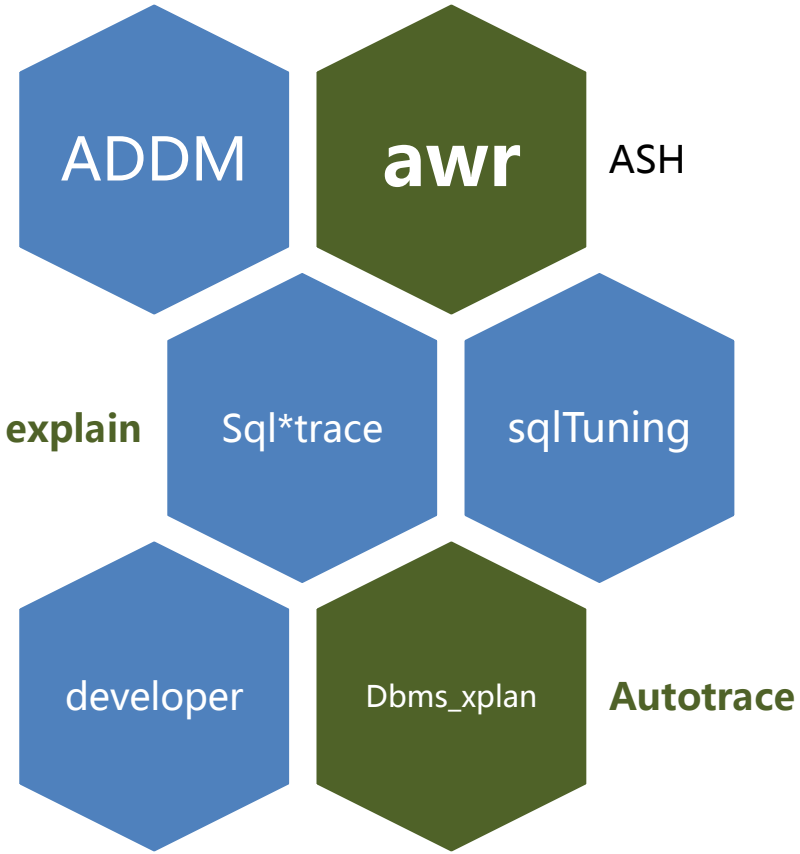


设计	业务规则；逻辑结构；物理结构；应用设计。
开发	索引策略；访问路径；合理pl/sql；减少锁冲突。
上线	资源优化和调整；OS平台优化和调整。

性能优化的量化指标

消耗时间	内存消耗	I/O消耗	语句分析次数
• Elapsed Time • Cpu Time	• Buffer Gets • Consistant Gets	• Physical Reads • Physical Writes	• Parses • Hard Parses • Soft Parses

性能分析工具


ADDM
awr
ASH
explain
Sql*trace
sqlTuning
developer
Dbms_xplan
Autotrace

读懂结果更重要

```
select IOID_ID0
from
 ls65_para_sid2.REGION_T where area_code = to_number(:v1) and length(ioid_id0)
```

```
SQL> explain plan for select * from scott.emp where mgr=7902;

Explained.

SQL> SEL
```

```
Execution Plan
----------------------------------------------------------
Plan hash value: 3956160932

--------------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |     2 |    76 |     3   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| EMP  |     2 |    76 |     3   (0)| 00:00:01 |
--------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("MGR"=7902)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
          7  consistent gets
          6  physical reads
          0  redo size
       1021  bytes sent via SQL*Net to client
        523  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

SQL>
```

优化绕不开的秘密之一：索引

常见索引

- ✓ B树索引
- ✓ 主键
- ✓ 唯一索引
- ✓ 函数索引
- ✓ 多字段复合索引
- ✓ 反转索引
- ✓ 位图索引
- ✓ 本地分区前缀索引
- ✓ 本地非前缀分区索引
- ✓ 全局分区
- ✓ 全文搜索索引

常见问题

- ✓ 索引被抑制
- ✓ 函数索引被不合理使用
- ✓ 复合索引的不合理使用
- ✓ 位图索引的不合理使用
- ✓ 索引泛滥

复合索引

- ✓ 前缀行
- ✓ 可选性

优化绕不开的秘密之二：绑定变量

绑定变量的方式

应用级绑定

Select * from emp
where empno= ： X

只需要做一次硬解析

系统级绑定

Cursor_sharing参数

Exact（默认）
Similar/force

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	100.00	Redo NoWait %:	100.00
Buffer Hit %:	99.82	In-memory Sort %:	100.00
Library Hit %:	89.29	Soft Parse %:	93.43
Execute to Parse %:	4.65	Latch Hit %:	99.93
Parse CPU to Parse Elapsd %:	0.01	% Non-Parse CPU:	90.25

优化绕不开的秘密之三：排序&连接

排序

- 减少排序、能不用就不用
- 尽量让排序在内存中完成
- 尽可能利用索引

连接

- 嵌套连接（nestloops）
- 哈希连接（Hash Join）
- 排序连接（Merge Join）

优化相关的话题：子查询 IN & Exist

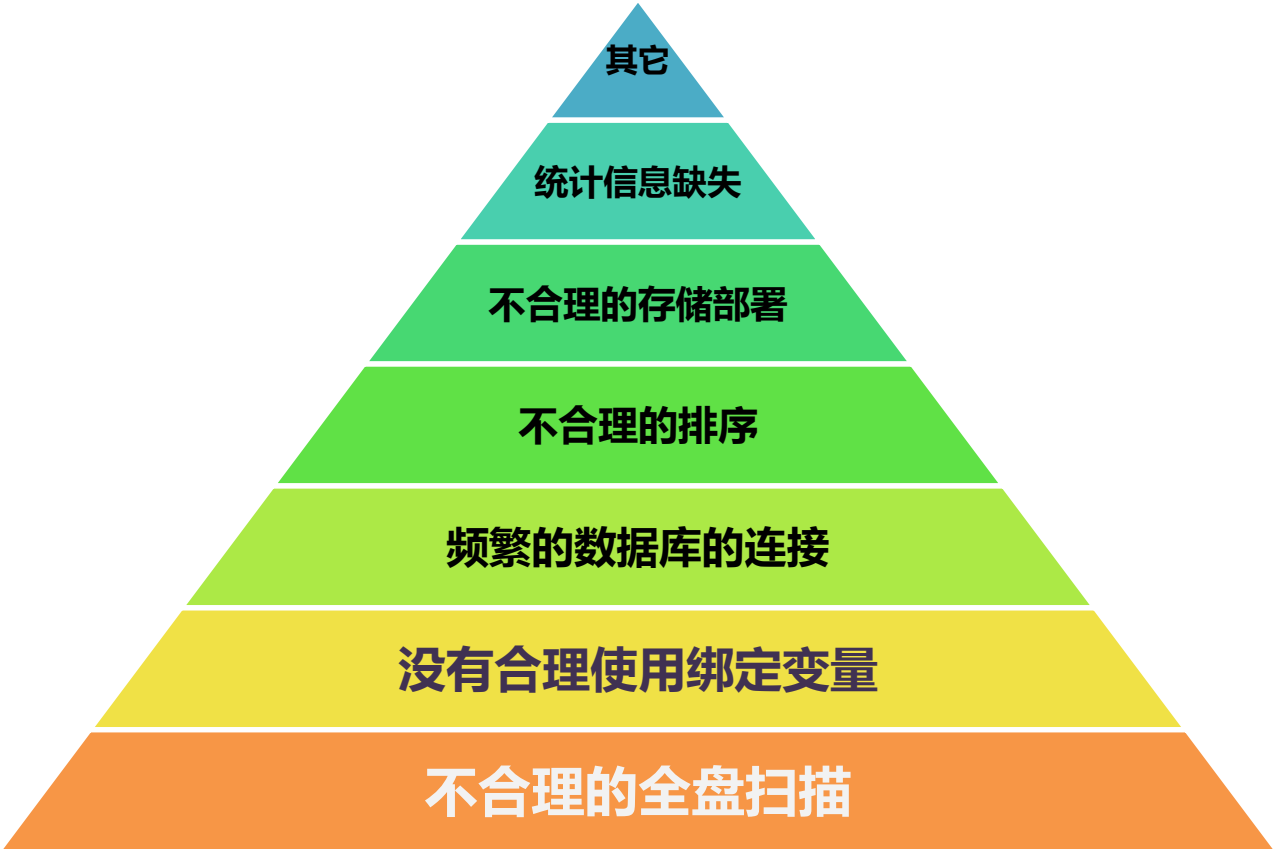
子查询

- 子查询会被转换为连接
- 子查询可读性不高
- 子查询会引导优化器走错误的执行路径

IN & Exist

- 10G之前会有不同的执行路径
- 11G 系统会有相同的执行路径
- 半（反）连接

导致数据库性能的常见问题



分区技术：不仅仅是为了优化

常见分区技术：范围分区、hash分区、list分区、复合分区

分区关注点

目标	分区是为了方便数据的管理，然后才是性能。
设计	分区表的设计 应用的设计 表空间的设计
索引	索引也应该合理的分区

并行：一把需要慎用的双刃剑

并行严格来说不能认为是性能优化，仅仅是使用资源换取时间

并行关注点

select	Select /*+ paralle(A,2) */ * from t1 A;
DML	Alter session enable parallel dml; Update /*+ parallel(2) */ ….. Insert /*+ append parallel(tablename,2)……
DDL	Create table as select …parallel;

RAC性能优化

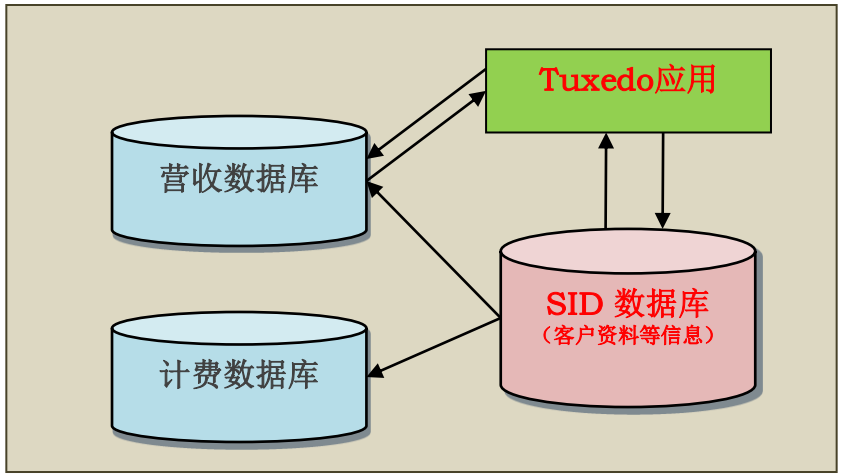
Global Cache and Enqueue Services - Workload Characteristics

Avg global enqueue get time (ms):	0.0
Avg global cache cr block receive time (ms):	0.7
Avg global cache current block receive time (ms):	0.7
Avg global cache cr block build time (ms):	0.0
Avg global cache cr block send time (ms):	0.0
Global cache log flushes for cr blocks served %:	4.1
Avg global cache cr block flush time (ms):	7.9
Avg global cache current block pin time (ms):	0.0
Avg global cache current block send time (ms):	0.0
Global cache log flushes for current blocks served %:	0.4
Avg global cache current block flush time (ms):	10.0

Global Cache and Enqueue Services - Messaging Statistics

Avg message sent queue time (ms):	0.0
Avg message sent queue time on ksxp (ms):	0.4
Avg message received queue time (ms):	0.0
Avg GCS message process time (ms):	0.0
Avg GES message process time (ms):	0.0
% of direct sent messages:	30.46
% of indirect sent messages:	66.86
% of flow controlled messages:	2.68

案例：某省电信计费库优化项目



- **SID库容量问题**
  总大小达到5.8TB，有19万个数据库对象，其中6万张数据表，1.4万个索引。容量只增不减，没有建立完整的数据生命周期管理策略。
- **SID库性能问题**
  业务高峰期CPU使用率超过80%，会话数达到19000个。
- **关键业务问题**
  TUXEDO核心应用，有时出现堵塞的现象，部分关键业务在集团考核中排名靠后。

典型问题

- **数据欠清理**：历史数据没有及时迁移；临时备份数据没有及时清理
- **分区不合理**：某些分区表只有1-2个分区；历史数据不按时间分区；某些大表未分区
- **索引不合理**：214个前导列重复的多余索引(单个索引最大73GB)；某TOP SQL因缺少索引而全表扫描；分区表全局索引影响数据在线清理
- **SQL待优化**：业务高峰时段的某些SQL存在明显的性能瓶颈，如全表扫描、超多表关联
- **连接数过高**：7天以上空闲等待的连接数共计6800个，分别占节点1和节点2的31%和42%
- **失效对象多**：6345个失效对象，占所有对象比例3.2%
- **TUXEDO关键业务的SQL有待优化**，Trace超过12万条的SQL语句的分析(针对TUXPAY22模块)。
- **没有配置合理的统计信息收集策略**，动态收集消耗CPU、易导致优化器生成非最优的执行计划。

Q & A

1、怎么看待数据在集中存储时以库为存储粒度或者以shema为存储粒度

2、如果看待对业务的了解程度对系统性能优化的帮助

谢谢