云和恩墨杨廷琨：日常运维中的技术决策

首页推荐

原创 墨天轮编辑部 2025-05-19 591

导读

在日常运维工作中，技术决策无处不在。每一次选择和判断，都会直接影响系统的稳定性与安全性。特别是在安全生产的背景下，正确的技术决策不仅能有效规避故障与风险，更是保障业务连续性的核心所在。

本期分享，墨天轮技术研究院院长、二线负责人杨廷琨将结合实际案例，为大家带来一场关于“日常运维中的技术决策”的深入思考与探讨，助力大家在复杂场景中做出更加稳妥、可持续的技术选择。

本文整理自直播分享，部分语句在不改变原意的基础上做了更改。

> 演讲资料：https://www.modb.pro/doc/145110
> 视频回放：https://www.modb.pro/video/10397

杨廷琨
墨天轮技术研究院院长，二线负责人

ACOUG核心专家，，高级咨询顾问，称"杨长老"；数年如一日坚持进行Oracle技术研究与写作，号称"Oracle的百科全书"，在自己的博客上累计发表了超过3000篇文章。

关于技术决策这个话题，前几周安全系列分享中，已有同事从操作层面介绍了如何避免误删表、防止连接错误数据库、规避不当操作等实用内容，为日常工作提供了指导。

今天，我想探讨更高层次的问题 —— **面对技术选择时，我们该如何思考和作出稳妥、可持续的决策**。这不仅涉及操作正确性，更关系到整体架构和方向的把控。为便于讨论，我准备了三个真实案例，期待与大家深入交流。

案例一：DDL删除字段导致核心系统全挂起

在数据库的日常运维中，DDL（数据定义语言）操作常常被认为是“轻量级”的，尤其是像添加或删除字段这样的结构变更。然而，真实的生产环境却给我们上了深刻的一课 —— **删字段并不简单，错误时机和操作方式，可能引发系统级灾难**。

这是我们团队亲历的一个客户故障案例，也成为我们后续在数据库平台设计和产品培训中反复引用的警示故事。

起因：一个删除字段的操作

某金融客户的数据库中有一张大表，表内存储着多年积累的业务数据。业务上线多年，表结构存在一定冗余，数据库团队计划清理掉一个“已经不用的字段”。考虑到只是“删字段”，也没安排停机时间窗口，就在白天业务高峰期直接执行了这条 DDL 语句。没想到，这条操作像是按下了“灾难按钮”。执行删除字段之后，系统逐渐表现出严重性能异常，最终所有业务请求被锁死，数据库不可用。客户紧急报警，我们团队介入分析。

- 某客户7*24系统出现业务挂起
- 检查发现运行了DDL语句删除字段
- 由于操作表很大，怕UNDO空间不足，使用了CHECKPOINT语句
- DDL导致表上的DML挂起，长时间影响业务，维护人员选择中止DDL
- 使用CHECKPOINT语句的DROP COLUMN是不可撤销操作，后续只能继续删除或删表
- 更严重的是目前表处于不一致状态，对表的查询都会被挂起

墨天轮编辑部
关注
906 文章 1002 粉丝 2564K+ 浏览量
获得了 8413 次点赞
内容获得 5503 次评论
获得了 5520 次收藏

TA的专栏
2025年磐维数据库培训 收录 24 篇内容
【2025 ACDU 中国... 收录 4 篇内容
数据库产业图谱 收录 9 篇内容

热门文章
12款受欢迎的实用数据库工具推荐 2022-11-10 39446浏览
2023年国产操作系统名录大全（2022.... 2022-04-15 26107浏览
国内数据库风云人物百人团 2022-09-23 25041浏览
DBA面试资源合集（含Oracle、MySQL、Redis等） 2021-10-15 24179浏览
墨天轮2022年度数据库获奖名单 2023-01-17 20804浏览

最新文章
2025年8月中国数据库排行榜：双星竞入三甲榜，TDSQL 连跃位次升 4天前 707浏览
2025年7月国产数据库大事记：GoldenDB创千万级大单，可信数据库大会召开... 6天前 784浏览
硬核对话即将开启：智能运维，真香？还是噱头！ 2025-07-24 168浏览
墨天轮「实操看我的」数据库主题征文活动启动 2025-07-22 379浏览
2025 ACDU中国行首站 ，8月2日广州见！ 2025-07-18 260浏览

目录
- 案例一：DDL删除字段导致核心系统全...
- 案例二：修改字段精度引发的数据处理...
- 案例三：数据坏块问题及金融客户的多...

图1:案例一背景

故障分析一：删除字段在大表上是高风险操作

绝大多数 DDL（如添加字段、修改字段名）只是修改数据字典，通常秒级完成。但删除字段属于特殊 DDL，会触发对表中每一个数据块的访问与清理操作，本质上类似一次全表级 DML，极其耗时。

> 这个操作没有预判到其真正影响，直接在生产高峰期执行，无异于自断经脉。

故障分析二：并发业务加剧锁竞争

在没有停机窗口的情况下执行这个操作，业务仍在高并发访问这张大表。而删除字段操作加了表级锁，长时间未释放，导致后续所有访问该表的业务全部被阻塞，排队等待锁。

> 数据库系统负载飙升，用户请求超时堆积，业务系统进入“假死”状态。

故障分析三：checkpoint 导致无法回滚

客户意识到问题后，尝试用 Ctrl+C 中断操作，本以为可以通过事务回滚挽救。没想到，执行 DDL 后不久，有人手动触发了 checkpoint —— 这意味着部分变更已经被写入数据文件，无法撤销。

CHECKPOINT

Specify CHECKPOINT if you want Oracle Database to apply a checkpoint for the DROP COLUMN operation after processing integer rows; integer is optional and must be greater than zero. If integer is greater than the number of rows in the table, then the database applies a checkpoint after all the rows have been processed. If you do not specify integer, then the database sets the default of 512. Checkpointing cuts down the amount of undo logs accumulated during the DROP COLUMN operation to avoid running out of undo space. However, if this statement is interrupted after a checkpoint has been applied, then the table remains in an unusable state. While the table is unusable, the only operations allowed on it are DROP TABLE, TRUNCATE TABLE, and ALTER TABLE DROP ... COLUMNS CONTINUE (described in sections that follow).

图2:checkpoint含义

以上的这些操作导致数据库陷入了“半完成状态”：删字段操作回不了头，也做不完，只能咬牙完成。故障持续了近1小时，客户技术团队最终紧急切换至备用数据库，由我们团队完成后续善后处理。

案例启发

这次事件给我们带来了深刻启发，也总结出如下三点教训：

1. **DDL 操作要评估表规模与运行状态**
   不要仅凭“经验”判断 DDL 是否安全，尤其在面对大表和复杂结构变更时，务必事先评估数据量与执行代价。

2. **生产环境慎做结构变更，特别是大表**
   建议安排专门的停机窗口或使用 Online DDL 工具，避免在业务高峰执行高风险 DDL。

3. **谨慎使用 checkpoint 和低层语句**
   不恰当的 checkpoint 操作可能直接破坏回滚机制，让系统陷入不可控状态。

- DDL是有风险的操作，大部分DDL需要在闲时或可维护窗口进行
- ALTER TABLE DROP COLUMN是高风险操作，执行时间和持有锁的时间取决于表的大小，需要在明确的可维护窗口内进行
- ALTER TABLE DROP COLUMN CHECKPOINT N是导致业务不可用的罪魁祸首，一旦发起操作没有反悔的机会，几乎所有情况都不应该考虑该语句
- 对于临时删除列，使用SET UNUSED COLUMN语句
- 彻底删除列推荐使用在线重定义，需要关注在线重定义的限制条件

图3:删除字段最佳实践

案例二：修改字段精度引发的数据处理难题

在实际客户场景中，遇到了一个表字段精度修改的问题。客户想将一个 number 类型字段的小数精度从 12 提升到 13（即由原来支持 2 位小数，提升为 3 位小数），但在执行时数据库却报错：
这是很多人在数据库操作中容易踩到的坑 —— **不能直接修改有数据的字段精度**。

```
SQL> create table t_num (id number, num number(10, 2));

表已创建。

SQL> insert into t_num
  2  with a as (select rownum, dbms_random.value(0, 999999999)/100 from dual connect by rownum <= 100000)
  3  select a.* from a, all_objects
  4  where rownum <= 10000000;

已创建 10000000 行。

SQL> commit;

提交完成。

SQL> alter table t_num modify (num number(10, 3));
alter table t_num modify (num number(10, 3))
                           *
第 1 行出现错误:
ORA-01440: 要减小精度或标度, 则要修改的列必须为空
```

图4:案例二背景

现场应对方案

一线同事第一时间提出的解决思路是：“既然不能直接改，那我就加一个新列，把老列的值复制进去，再删掉老列。尽管**并行加速**能解决部分性能问题，但是：运行这种方式**其实不是较优方案，并没有解决所有的问题。**”

方案	描述	执行时间	行迁移
方案一	直接更新新增列数据	207	44%
方案二	并行度4直接更新	77	44%
方案三	同时更新新增列和原始列	148	0
方案四	并行度4同时更新	40	0

图5:前期四种方案对比

后来，我们对这个问题进行了深入分析，逐步还原了背后的技术根因，发现主要问题在于以下几点：

- 新增字段并灌入数据，会导致每行记录的长度增加；
- 原表数据已经接近满页，此时一旦数据变长，原有位置放不下，就会引发 行迁移（row migration）；
- 行迁移会带来显著的性能问题，特别是后续的查询、更新操作可能明显变慢；

此外，删除列的操作本身也潜藏高风险，比如可能涉及元数据修改、对象锁定等，稍有不慎就会影响线上业务。

二线优化建议

针对这个场景，二线提出了几条实用的优化建议，希望为大家后续处理类似问题提供参考：

- 优先评估是否可以采用在线重定义（Online Redefinition）或其他更平滑的方式替代直接操作原表，从而降低对生产系统的冲击；
- 尽量避免大范围的 `UPDATE` 操作，可以考虑分批、按条件进行更新，降低资源消耗；
- 如果确实需要新增列并更新数据，应事先评估表空间使用情况，防止触发大规模行迁移；
- 在删除列之前，务必要做好备份和演练，确保不会对元数据造成不可逆的破坏。

```
SQL> create table t_num6 as select * from t_num;

表已创建。

SQL> update t_num6 set id = rownum;

已更新 10000000 行。

SQL> alter table t_num6 add primary key (id);

表已更改。

SQL> create table t_num_inter (id number primary key, num number(10, 3));

表已创建。

SQL> set serverout on size 1000000
SQL> exec dbms_redefinition.can_redef_table(user, 'T_NUM6')

PL/SQL 过程已成功完成。


SQL> exec dbms_redefinition.start_redef_table(user, 'T_NUM6', 'T_NUM_INTER')

PL/SQL 过程已成功完成。

SQL> exec dbms_redefinition.finish_redef_table(user, 'T_NUM6', 'T_NUM_INTER')
```

```
PL/SQL 过程已成功完成。

SQL> desc t_num6
 名称                      是否为空? 类型
 ----------------- -------- ----------------
 ID                NOT NULL NUMBER
 NUM                        NUMBER(10,3)
```

图6:优化方案：在线重定义

```
SQL> create table t_num7 as select * from t_num;

表已创建。

SQL> alter table t_num7 modify (num number(10, 3));
alter table t_num7 modify (num number(10, 3))
                           *
第 1 行出现错误:
ORA-01440: 要减小精度或标度, 则要修改的列必须为空


SQL> set timing on
SQL> alter table t_num7 modify (num number(11, 3));

表已更改。

已用时间:  00: 00: 00.04
```

图7:优化方案：在线修改

案例三：数据坏块问题及金融客户的多方案修复实践

在数据库运维中，坏块（block corruption）是一个不可忽视的问题，尤其在金融行业的 7×24 小时系统中，大面积坏块可能导致业务中断和数据一致性风险。

本案例客户来自金融行业，数据库系统长期在线运行。通过 ACO 的 date block corruption 视图，检测到约 600 条坏块记录，实际涉及更多非连续数据块，分布在多个数据文件中。尽管当前业务尚无明显异常，但若坏块增多或被频繁访问，可能严重影响系统稳定性和数据写入，需及时修复。

```
SQL> select * from v$database_block_corruption order by file#,block#;

     FILE#     BLOCK#     BLOCKS CORRUPTION_CHANGE# CORRUPTIO
---------- ---------- ---------- ------------------ ---------
         2     494066          2                  0 CORRUPT
         2    3866624        512                  0 CORRUPT
         2    3737600        512                  0 CORRUPT
         2    2372096       1024                  0 CORRUPT
         7    2148352        512                  0 CORRUPT
         7    1567744        512                  0 CORRUPT
         8    1099990         17                  0 CORRUPT
         8    1100268         20                  0 CORRUPT
         8    1100188         61                  0 CORRUPT
        12     160769        511                  0 CORRUPT
        12     160768          1                  0 CORRUPT
        16     792441         15                  0 CORRUPT
        16    1435648         14                  0 CORRUPT
......
       292    2134333          2                  0 CORRUPT
       292    2134336          2                  0 CORRUPT

585 rows selected.
```

图8:案例三背景

修复方案比较

修复方式	适用场景	系统影响	停机时间	优点	缺点或风险
全库恢复（整体恢复）	坏块数量多，影响频繁，前台频繁报错，需确保数据一致性	全库业务不可用	长，可能数小时	保证数据一致性，适用于严重损坏场景	停机时间长，业务全面中断
数据文件级恢复	坏块集中在部分数据文件，影响范围可控，业务影响有限	局部业务可能受影响	中等，依恢复数量而定	操作灵活，可分文件分批恢复	每次操作范围小但整体耗时长，影响业务不可控，容易引发连锁访问错误
主备切换（DG切换）	有可用的同步备库，主库出现坏块但备库无报错，适用于7×24高可用场景	取决于备库承载能力与切换流程	短，一般10分钟~30分钟	避免主库直接受影响，便于主库修复	切换存在风险，需改IP、调整中间件配置，可能引发其他系统联动问题
数据块级恢复（Block Recover）	坏块数量少且分散，对业务影响不明显，有备份或ADG支持	几乎无业务影响	极短或无停机时间	修复精确、无感知	需具备完整备份或ADG支持，某些版本才支持自动修复

客户方案

综合评估业务影响、恢复复杂度与可控性后，最终为客户选择了Block Recover。这一方案在不中断系统运行的前提下，快速精准地修复了损坏的数据块，避免了业务停机、恢复复杂和不可控风险。

案例总结

当面临数据库坏块时，切勿一刀切地选择全库恢复或盲目切主备。应结合以下几个维度做出判断：

- 坏块数量与分布
- 业务影响程度
- 系统架构（是否有备库）
- 运维窗口的可接受时长

- 技术方案决策错误的代价比操作错误更大
- 技术方案的确定需要专家评估可行性、风险和代价
- 发生预期外的事故时，及时技术升级避免错过最佳时间窗口
- 深入理解技术原理才能找到最佳解决方案
- 多思考多复盘，有助于技术能力的快速提升

图9:案例总结

技术决策看似日常，却关乎系统的生命线。只有不断积累经验、沉淀方法，才能在关键时刻快速响应、精准判断，确保业务安全稳定运行。

以上就是我今天分享的所有内容，谢谢大家！

数据库 数据库运维

最后修改时间：2025-05-20 11:49:29

评论

分享你的看法，一起交流吧～

每日一步 LV.6

精彩！

2月前 点赞 评论

潇湘雨 LV.2

杨老师的博客哪里可以看到

2月前 点赞 1

墨天轮福利君 LV.5

个人博客地址：http://www.yangtingkun.net

2月前 点赞 回复

相关阅读

【DBA坦白局】第三期：作为DBA，你加过最晚的班是到几点？在干什么？

墨天轮编辑部 1400次阅读 2025-07-15 10:28:44

重磅 | 万里数据库GreatDB亮相上合组织数字经济论坛 以硬核科技共绘“数字丝路”新图景

万里数据库 592次阅读 2025-07-15 09:45:41

2025年7月国产数据库中标情况一览：长沙银行千万采购GoldenDB，秦皇岛银行近七百万采购TDSQL！

通讯员 586次阅读 2025-08-07 10:18:23

中国信通院2025上半年“可信数据库”新增标准解读

大数据技术标准推进委员会 562次阅读 2025-07-21 10:45:15

希望和国产数据库厂商共建公共知识库

白鳝的洞穴 479次阅读 2025-07-28 12:24:07

ACDU周度精选 | 本周数据库圈热点 + 技术干货分享（2025/7/25期）

墨天轮小助手　469次阅读　2025-07-25 15:54:18

ACDU周度精选 | 本周数据库圈热点 + 技术干货分享（2025/7/17期）

墨天轮小助手　436次阅读　2025-07-17 15:31:18

GBASE南大通用亮相2025 CHITEC，当数据库“豫”见医疗信息化

GBASE数据库　345次阅读　2025-07-22 10:48:57

“融合进化 智领未来”电科金仓2025产品发布会成功举办！

金仓数据库　344次阅读　2025-07-16 10:28:54

2025信创500强 科蓝软件三度上榜

科蓝软件　291次阅读　2025-07-25 10:03:59