# Oracle表空间切换方案

# 背景

由于之前数据库表和索引放在一个表空间导致表空间数据文件增长太快，文件数量即将达到Oracle表空间的限制，需要对表(没有分区，有些表数据量多达几十亿，文件大小TB级)进行表空间迁移，并对某些表改造为分区表。

# 可选方案分析

## 使用IMP/EXP

导入(import)导出(export)工具年头已久，将数据导出为二进制文件，将会在11g r2之后废弃，只用于遗留数据的导入导出

此工具可以完成所需功能，但有如下限制：

1. 导出数据量大时非常慢，只能分批操作
2. 不支持表名映射，需要第三个用户或者数据库参与
3. 需要操作用户有DBA权限
4. 需要有IMP/EXP工具操作权限
5. 在客户端操作，受到网络影响

## 使用IMPDP/IMPDP

EXPDP/IMPDP工具从10g开始引入，参数格式与EXP/IMP类似

优点：

1. 可以并发执行导入导出任务
2. 可以导入导出多个文件
3. 无需缓存，直接操作数据库和目标文件
4. 不受网络影响，expdp服务端程序
5. 支持对象映射，可直接映射表空间和表
6. 可以从未分区表导入到分区表

缺点：

1. 需要服务器约5TB的磁盘空间来存储二进制文件，可以分批导出
2. 一次性导入undo空间要足够大

## 使用alter table \*\* move tablespace \*\*\*

此命令是将一个表移动到另外一个表空间中

优点：

1. 对于大表move时，对此表的查询不受影响，只有DML操作受影响
2. 索引结构不受影响，只需move完成后rebuild
3. 与其它对象依赖关系不受影响，操作前不必为对象间的依赖关系操心
4. move操作可以parallel。
5. NOLOGGING选项可大大加快重建速度，如果要move的表是nologging的，则不需指定

缺点：

1. 当前的用户必须对原表空间和新表空间都有操作权限
2. 不能改变表结构从普通表转换为分区表
3. 数据太大可能操作不成功
4. 经过验证不会释放原表空间数据文件已经格式化的磁盘空间
5. 当目标表空间不足时会中途才提示失败

## 采用表重定义

把一个数据量非常大的普通表改造成分区表，如果离线操作能够解决问题，就不要用在线重定义，例如一些静态数据、历史数据的归档迁移，可使用CTAS、alter table move、或导出导入完成

使用场景：

1. 修改表的物理属性、存储参数
2. 将表迁移到别的表空间
3. 消除表碎片、释放空间
4. 在表中增加、删除或重命名字段
5. 大批量改变表中的数据

原理：

通过DBMS\_REDEFINITION包来实现，首先会创建一个快速刷新的物化视图作为过渡表，然后将源表的数据加载到过渡表中，并在源表上创建物化视图日志，以支持快速刷新同步数据

限制条件与风险：

1. 使用基于主键的方式，则原表与重定义后的表必须有相同的主键
2. 使用基于ROWID的方式，不能是索引组织表
3. 有物化视图或者物化视图日志的表、物化视图容器表、高级队列表、索引组织表的溢出表、拥有BFILE，LOGN列的表、 Cluster表、sys和system下的表、临时表不能在线重定义
4. 不支持水平数据子集
5. 在列映射时只能使用有确定结果的表达式，如子查询就不行
6. 中间表有新增列，则不能有NOT NULL约束
7. 原表和中间表之间不能有引用完整性
8. 在线重定义无法采用nologging
9. 表空间至少要留有比源表所用空间更大的剩余空间，也就说至少要事先创建5T的表空间
10. 对业务的影响小，但过程耗时较长，测试两千万的数据花费十多分钟 20
11. 如果源表上的事务操作过于频繁，可能会发生较严重的等待，不存在事务

参考：<https://www.sohu.com/a/166577098_505827>

## 采用CATS +RENAME

CTAS这种方法采用DDL语句，不产生UNDO，只产生少量REDO，建表后数据已经在分布到各个分区中，最后交换源表与目标表的名字即可

核心sql：create table t(id, time) partition by range (time) (partition t1 values less than (to\_date('201311', 'yyyymm')), partition t2 values less than (maxvalue)) nologging parallel 4 as select /\*+parallel\*/ id, time from s;

性能提升方式：

1. 加nologging:alter table t nologging;完成后根据需要将表修改为logging
2. 并行DDL: alter session enable parallel dml;
3. 查询并行

## 采用INSERT+RENAME

这种方法适用于包含大数据量的表转到分区表中的一个分区的操作，就是先建立分区表结构然后使用insert来实现，将满足一个分区的数据查到某个中间表中，然后交换中间表和目标表的分区，每个部分都做完后交换源表与目标表的名字(alter table p exchange partition p1 with table t)。

性能提升：

1. 表修改为nologging
2. 启用并行DML，alter session enable parallel dml;
3. 采用 append方式插入
4. 所有数据插入完成后再建索引

不足：

1. 仍然存在一致性问题，交换分区之后RENAME T\_NEW TO T之前，查询、更新和删除会出现错误或访问不到数据，不过当前不存在
2. 要求数据分布到多个分区中，会增加操作的复杂度，效率也会降低

## 数据清洗方式

这种方法就是先建立表结构，然后建立任务每个一段时间，从源表查询一定时间方位内的数据插入到新表中，最后交换源表与目标表的名字，操作简单。

性能提升：

1. 表修改为nologging，
2. 数据插入完成后再建索引
3. 启用并行DML，alter session enable parallel dml;
4. 采用 append方式插入

# 迁移原则

1. 将所有的索引都放在TM\_INDX表空间
2. 迁移后的表（或者分区）尽量均匀分布在新的表空间
3. 选择合适的表结构（聚簇表、IOT、分区表…），提升查询性能、可维护性
4. 合理设置表结构的参数，提升查询性能、节约存储空间
5. 选择合适的索引（函数、位图…）、删除不合理的无用的索引

# 迁移步骤

## 小表

对于小表或者数据量稳定的表不需要分区，不需要做表空间迁移，或者直接采用方案5（ctas+rename）方式均匀迁移到新的表空间中

## 迁移大表

首先排除方案3（不能释放空间）和方案4（效率低，基本不涉及事务）

1. 确认需要操作的表（迁移、重建、删除）
2. 删除功能重复的表
3. 先迁移数据量相对比较小的表（10-100G），分别验证方案1、2、5、6、7的效率和可行性
4. 根据验证结果选择最优的方式迁移大表

## 索引

将用不到的索引删除

将所有放在TM\_DATA表空间的索引rebuild到TM\_INDX表空间