
数据技术嘉年华
// Data Technology Carnival
开源 · 融合 · 数智化 — 引领数据技术发展 释放数据要素价值
中国DBA联盟
All China DBA Union
墨天轮

数据技术嘉年华

openGauss在民生银行的应用实践

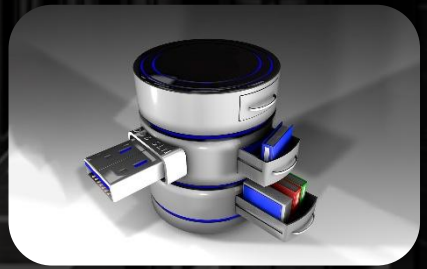
演讲人：孔再华

中国DBA联盟 All China DBA Union

墨天轮

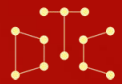
概述

民生银行选择openGauss作为主要的自主可控数据库产品，在过去几年内建设了比较全面的运维体系。本次交流重点关注openGauss在民生银行内部的应用架构实践情况。

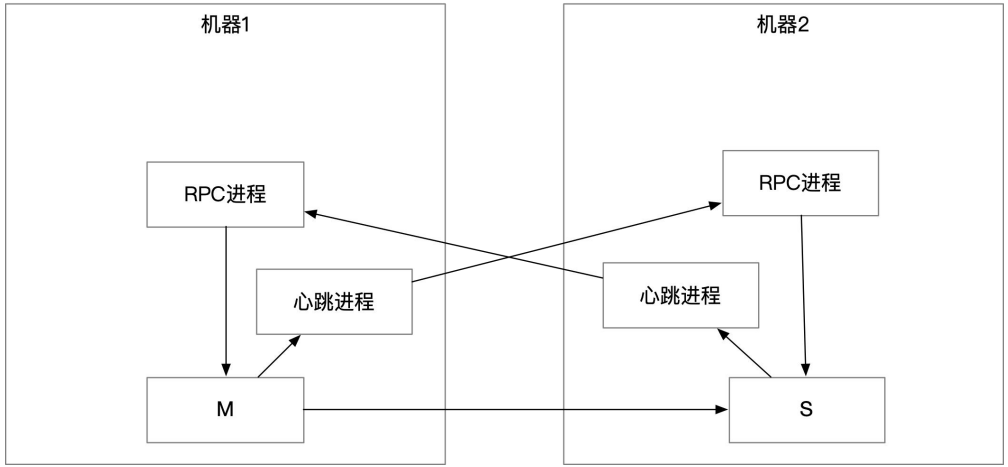


目录

CONTENTS



数据库高可用方案



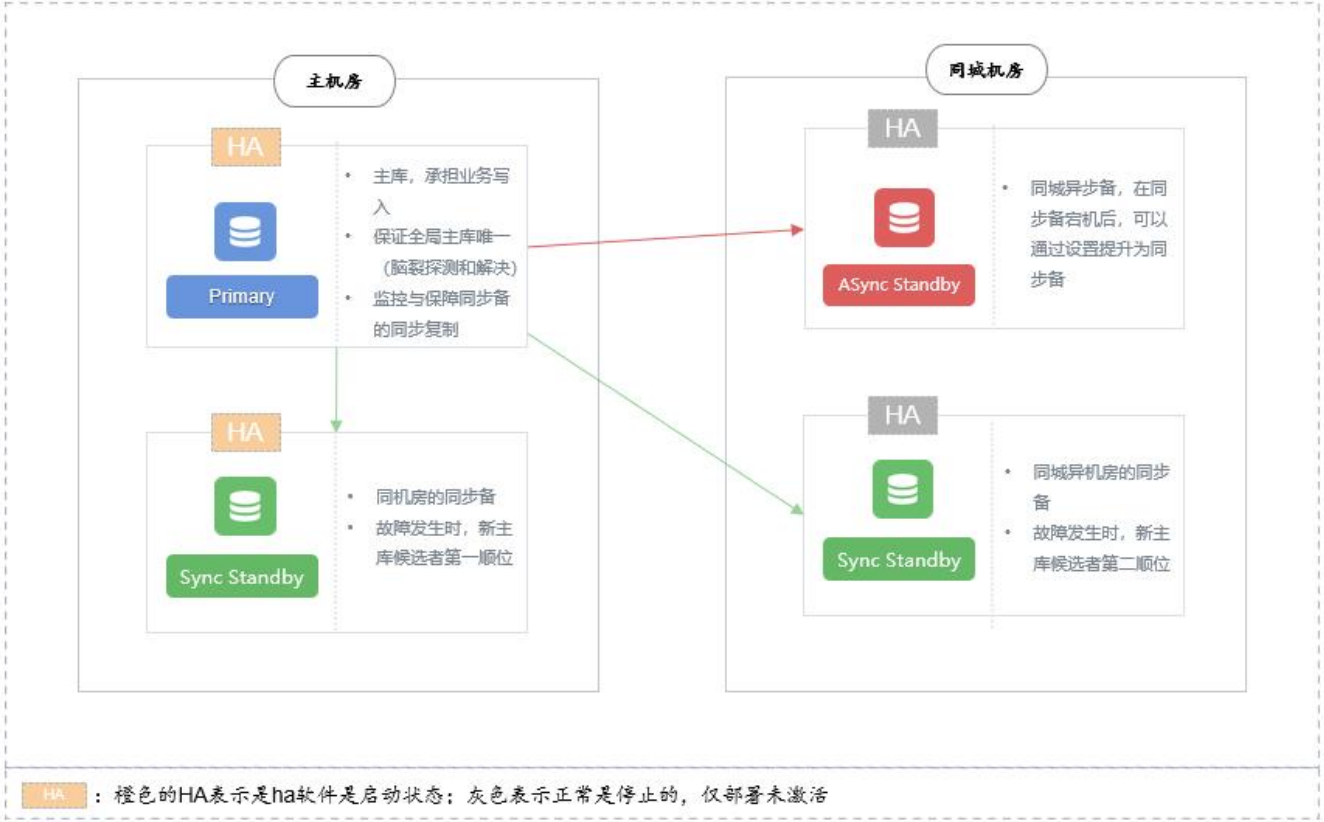
- 本地进程守护
- 主库丢失切换
- 文件系统检测
- 双主检测自杀
- 主库孤单自杀
- 从库丢失检测

虚拟机方案-同城双中心

一主一从

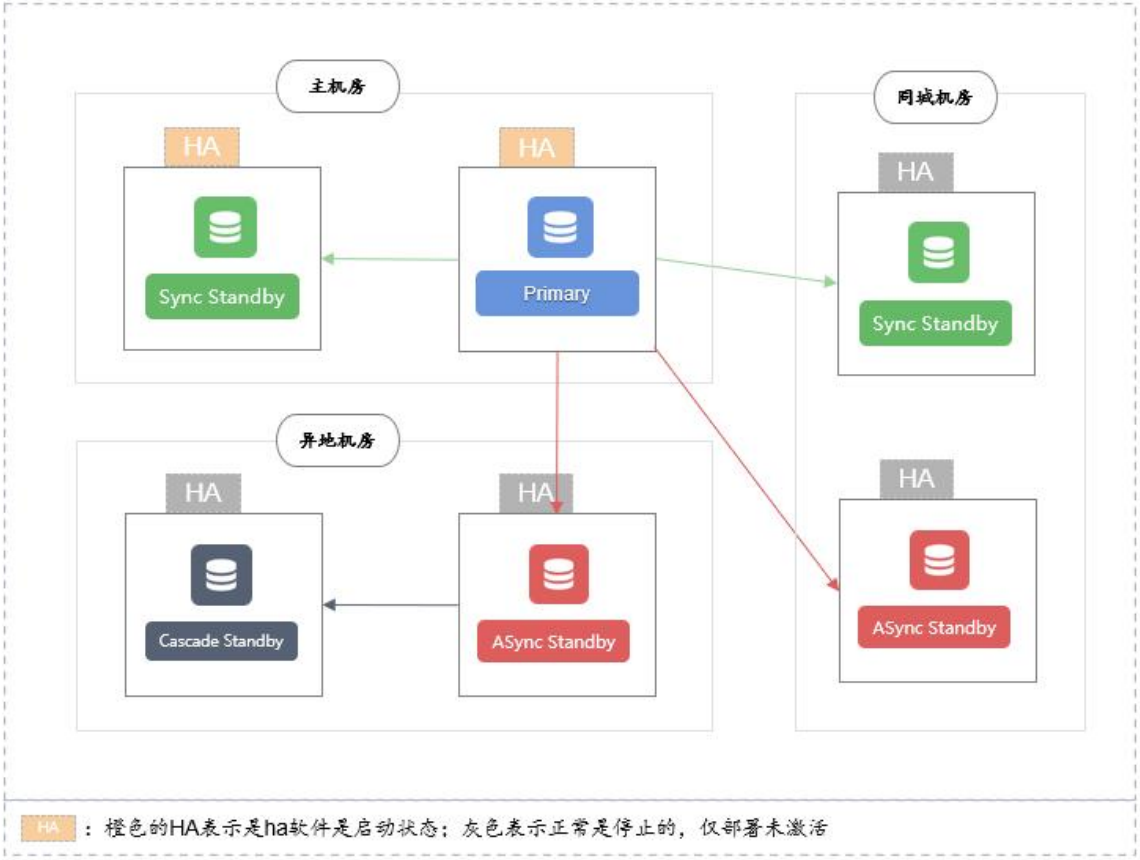
- 虚拟机高可用漂移，主从不切换
- HA守护数据库进程，故障自动拉起

物理机方案-同城双中心



- 本地同城数据同步
- 采用FIRST设置
- 本地HA激活故障切换
- 同城HA非激活状态（或仅守护）
- 同城切换人工干预

两地三中心容灾方案

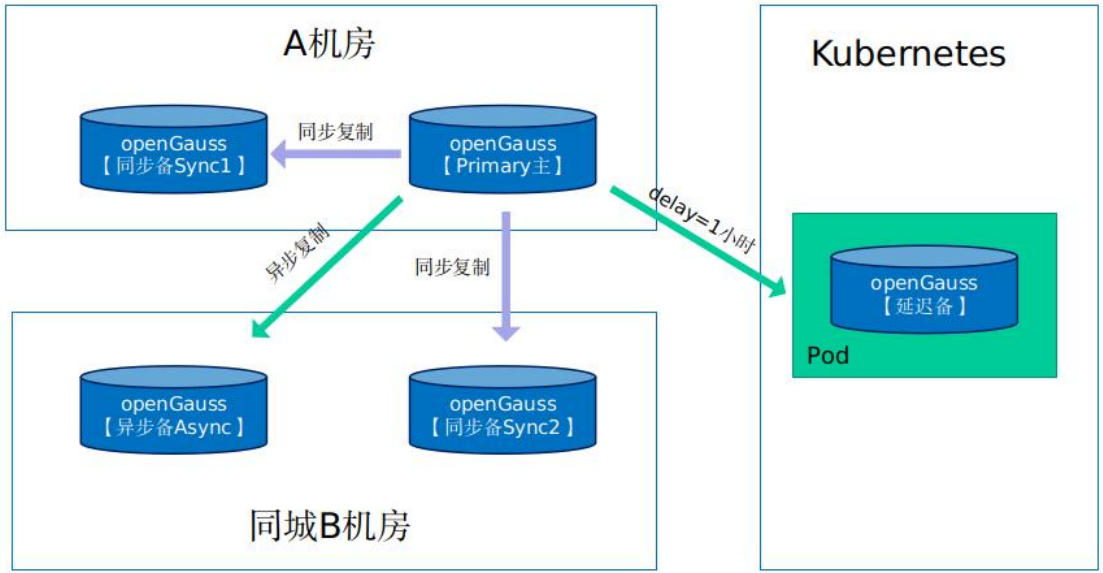


- 异地数据异步传输
- 异地级联复制

（无多级级联）

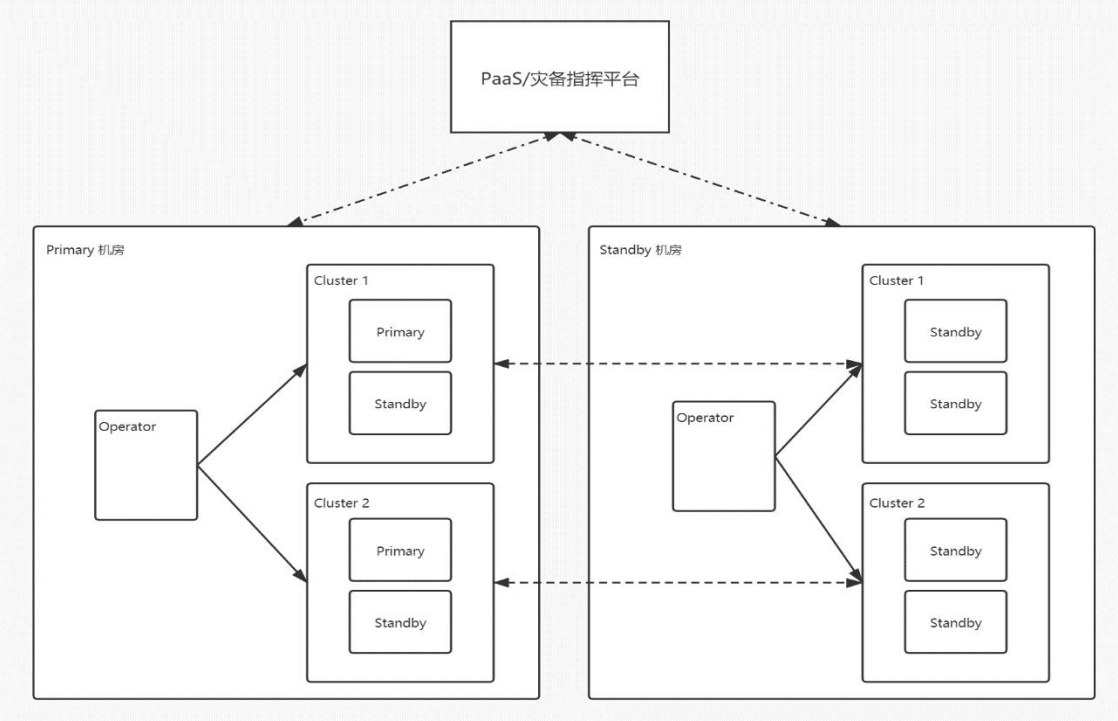
- 异地演练方案
- 异地切换方案
- 双集群同步方案验证中...

逃生库方案



- 延迟回放库（recovery_min_apply_delay）
- 云原生大规模部署
- 独立于OM集群管控
- 迅速恢复到目标时间点
- 可扩容承载业务

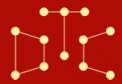
云原生方案 – operator



- 多中心集群部署
- 全生命周期管理
- 多网络插件支持
- 多存储插件支持
- 可维护状态
- 故障自愈

目录

CONTENTS



客户端自适应 – NO VIP

重要提示：不采用vip模式，而是配置所有主从的ip列表（通过dns配置主机名指向物理ip），利用客户端的自动辨识功能连接主库。（ VIP需要第三方集群管理工具来实现，增加复杂性。）

- JDBC客户端

openGauss jdbc客户端支持多ip配置，支持读写分离。

- ODBC客户端

支持多ip配置，不支持读写分离。只通过transaction_read_only属性区分是否为主库。

- Libpq

支持多ip配置，不支持读写分离。

- Python驱动和Go驱动

- 支持多ip和sha256加密版本。

”本地提交”优化

- 主库宕机后不能加入为新从库！
- 数据不一致！
- 分叉点为local commit
- Build覆盖数据很危险
- **增加同步点信息，比对后安全build。**

行存压缩

大小对比	tabsize_mb	idxsize_mb	totalsize_mb
表不压 索引不压	2232.78906250000	1956.56250000000	4189.35156250000
表不压 索引压	2232.75000000000	172.07894897461	2404.82894897461
表压 索引不压	307.01840209961	1956.38281250000	2263.40121459961
表压 索引压	307.01840209961	172.01693725586	479.03533935547

- 节省空间：整体压缩率较高。
- 参数复杂：参数较多，组合复杂，不易使用。
- 启用不便，需要新建表，迁移数据。
- 成熟稳定性需加强

行存压缩 – 性能比较

insert	执行CPU-avg	执行IO-avg	执行时间	TPS
表不压 索引不压	30.61%	59.5578%	2m13.376s	75818.70
表不压 索引压	30.88%	59.9852%	2m9.376s	78241.80
表压 索引不压	32.65%	60.5555%	2m9.396s	78418.20
表压 索引压	33.05%	58.9529%	2m6.373s	80902.30

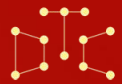
update	执行CPU-avg	执行IO-avg	执行时间	TPS
表不压 索引不压	30.06%	65.7894%	3m13.808s	52344.30
表不压 索引压	31.67%	65.4776%	3m11.841s	52497.20
表压 索引不压	34.62%	67.0600%	2m40.828s	63898.10
表压 索引压	34.78%	66.6355%	2m39.818s	64174.50

select	执行CPU-avg	执行IO-avg	执行时间	TPS
表不压 索引不压	77.65%	1.5828%	1m18.382s	141463.00
表不压 索引压	77.93%	1.6697%	1m14.365s	148813.00
表压 索引不压	78.60%	1.9277%	1m17.251s	141200.00
表压 索引压	78.65%	2.0880%	1m12.314s	154718.00

delete	执行CPU-avg	执行IO-avg	执行时间	TPS
表不压 索引不压	36.15%	49.8549%	1m35.747s	111379.00
表不压 索引压	38.38%	50.8250%	1m32.511s	117110.00
表压 索引不压	40.99%	50.9472%	1m34.671s	112881.00
表压 索引压	37.07%	56.7343%	1m35.824s	113375.00

目录

CONTENTS



分区表全局索引失效

- openGauss案例：
  - 某系统删除分区后，SQL性能变差，数据库堵塞，中间件资源耗尽，系统不可用。
- oracle案例：
  - Xx系统删除分区后，SQL性能变差影响交易，造成生产事件！影响巨大！
- 问题根因：
  - 分区表上建立全局索引后，如果此时操作分区，会导致全局索引失效！
  - 如需维持全局索引，需要人为加上update global index选项！
- 期望：
  - 更新全局索引为默认行为
  - 可异步更新，不影响交易

表膨胀问题

表膨胀(table bloat)是指表的数据和索引所占文件系统的空间，在有效数据量并未发生大的变化的情况下，不断增大

观察到的现象：

pg_stat_all_tables 视图查看 live_tup 和 dead_tup 的数量，发现n_dead_tup的数量是n_live_tup的N倍，膨胀十分严重

```
openGauss=# select * from pg_stat_all_tables where relname = 'vac_test' ;
-[ RECORD 1 ]-----+------------------------------
relid             | 16961
schemaname        | public
relname           | vac_test
seq_scan          | 328
seq_tup_read      | 9453687223
idx_scan          | 7319484949
idx_tup_fetch     | 66486320864
n_tup_ins         | 114580778
n_tup_upd         | 6439357491
n_tup_del         | 93271368
n_tup_hot_upd     | 4286447506
n_live_tup        | 21250167
n_dead_tup        | 1016968483
last_vacuum       |
last_autovacuum   | 2022-11-24 21:02:20.816616+08
last_analyze      |
last_autoanalyze  | 2022-11-24 21:02:20.816616+08
vacuum_count      | 0
autovacuum_count  | 232
analyze_count     | 0
autoanalyze_count | 0
last_data_changed | 2022-11-24 21:03:52.988215+08
```

严重的表膨胀的后果：

- 空间持续上涨，导读某一个点后，需要执行一个高额代价的vacuum full，但是vacuul full期间会阻塞所有的读写，这意味着在完成清理之前，都无法访问；
- 扫描的效率变低

产生表膨胀的原因：

openGauss astore存储引擎的MVCC原理：在写新数据时，旧的数据不删除，而是把新数据插入，旧的数据只是被标记为不可见。这些旧的数据又称为死元组（dead tuple）

表膨胀问题 – 解决方案

善用分区表

减少由于delete数据导致的表膨胀问题。通过表分区，当删除不需要的数据的时候，不delete，而是直接删对应的分区。因为delete只是让数据不可见，而不会真正的从磁盘空间删除

长事务

尽量避免长的SQL或者带有事务id的长事务

复制槽

加强对复制槽的关注，及时drop 失效的复制槽

参数

对 autovacuum 相关参数进行调整

分区表视图问题

分区表删分区需要先drop视图

```
openGauss=# \d part_test
            Table "public.part_test"
Column    |              Type              | Modifiers
------------+--------------------------------+-----------
id          | integer                        | not null
col1        | character varying(8)           |
create_date | timestamp(0) without time zone |
Partition By RANGE(create_date)
Number of partitions: 4 (View pg_partition to check each partition range.)

openGauss=# create view v_part AS select * from part_test ;
openGauss=# alter table part_test drop partition part1 ;
ERROR:  Cannot perform this operation because There are views or rules that depend on table part_test.
DETAIL:  N/A
HINT:  drop the views or rules first. Use pg_rules to find rules. Use pg_class, pg_rewrite, pg_depend, pg_namespacesql to find views
```

查看与该分区表相关的视图：

处理方法：先删除视图，等drop 分区之后再重建视图

查看视图的定义：

- 系统视图：pg_views 视图的definition字段
- 元命令：\d view_name
- 函数：pg_get_viewdef

```
openGauss=# select relname from pg_class where oid in(
                select c.ev_class
    from pg_depend a,pg_depend b,pg_class pc,pg_rewrite c
    where a.refclassid=1259
    and b.deptype='i'
    and a.classid=2618
    and a.objid=b.objid
    and a.classid=b.classid
    and a.refclassid=b.refclassid
    and a.refobjid<>b.refobjid
    and pc.oid=a.refobjid
    and c.oid=b.objid
    and pc.relname='part_test');
relname
-----------
v_part
(1 rows)
```

预编译like性能问题

建立btree索引，正常执行查询会走索引：

```
openGauss=# \d+ t
                    Table "public.t"
 Column |  Type   | Modifiers | Storage  | Stats target | Description
--------+---------+-----------+----------+--------------+-------------
 a      | integer |           | plain    |              |
 b      | text    |           | extended |              |
 c      | boolean |           | plain    |              |
Indexes:
    "t_b_idx" btree (b) TABLESPACE pg_default
Has OIDs: no
Options: orientation=row, compression=no

 openGauss=# explain (costs off) select * from t where b like ('c') ;
              QUERY PLAN
 ------------------------------------
 Bitmap Heap Scan on t
 Filter: (b ~~ 'c'::text)
 ->  Bitmap Index Scan on t_b_idx
         Index Cond: (b = 'c'::text)
 (4 rows)
```

采用PRAPARE绑定变量的方式进行模糊查询：
走的全表扫，不走索引

```
openGauss=# PREPARE select_t(text) AS select * from t where b like ($1) ;
PREPARE
openGauss=# explain (costs off) execute select_t('c') ;
    QUERY PLAN
---------------------
Seq Scan on t
Filter: (b ~~ $1)
(2 rows)
```

统计信息丢失

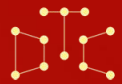
ALTER COLUMN .. SET DATA TYPE
修改列的类型会导致统计信息的删除

```
openGauss=# \d+ a
                        Table "public.a"
Column |         Type          | Modifiers | Storage  | Stats target | Description
--------+-----------------------+-----------+----------+--------------+-------------
id     | integer               |           | plain    |              |
info   | character varying(10) |           | extended |              |
Indexes:
    "idx_id_a" btree (id) TABLESPACE pg_default
Check constraints:
    "chk" CHECK (info::text = lower(info::text))
Has OIDs: no
Options: orientation=row, compression=no

openGauss=# select count(*) from pg_stats where tablename = 'a' and attname = 'info' ;
count
-------
    1
(1 row)

--# 修改列类型
openGauss=# alter table a alter COLUMN info type text;
ALTER TABLE
openGauss=# select count(*) from pg_stats where tablename = 'a' and attname = 'info' ;
count
-------
    0
(1 row)
```

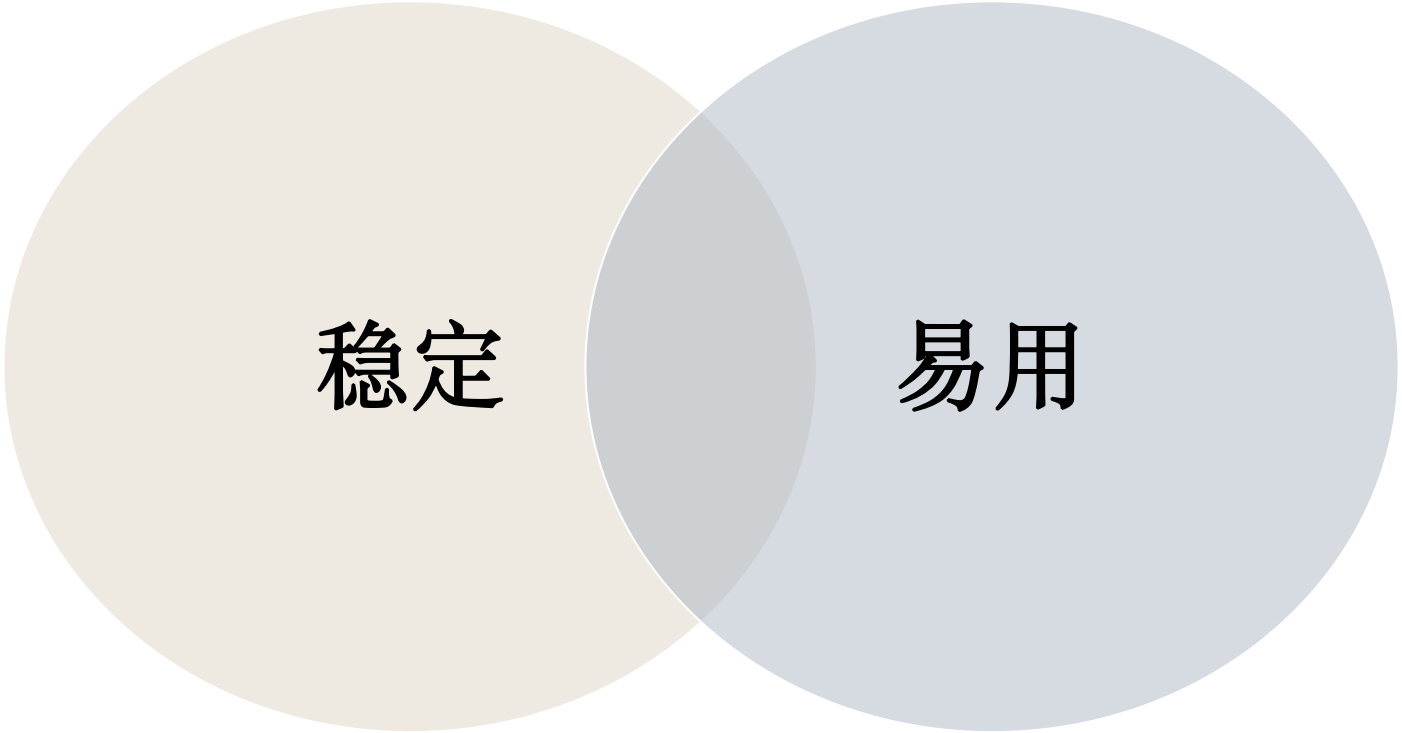
日常使用中需注意：
在更改完列的类型后，执行analyze重新收集该表的统计信息

目录

CONTENTS



产品需求期望



谢谢观看

THANKS FOR WATCHING