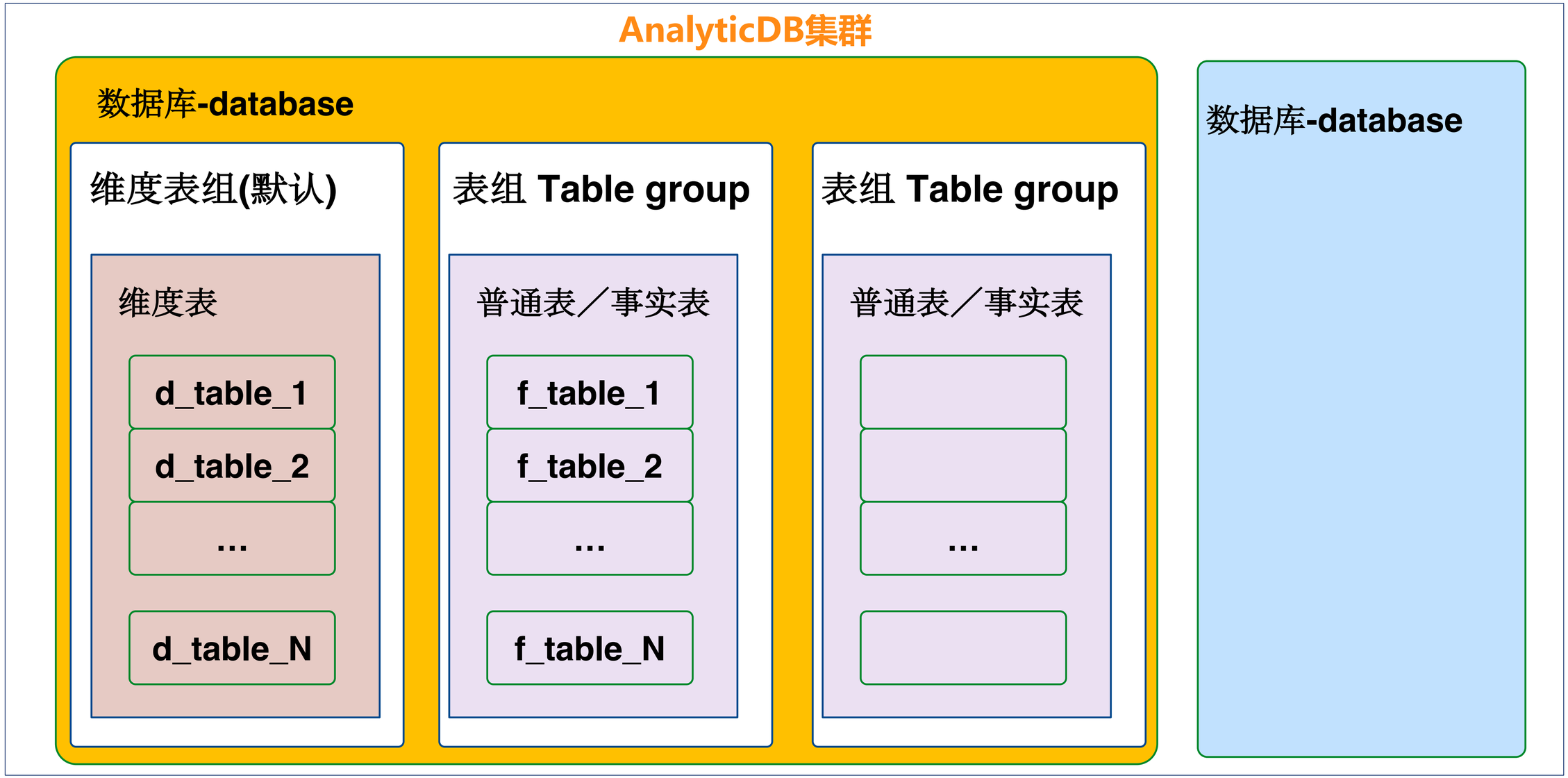
分析型数据库 AnalyticDB

表设计与规划

提纲

数据库逻辑对象


AnalyticDB集群
数据库-database
维度表组(默认)
维度表
d_table_1
d_table_2
...
d_table_N
表组 Table group
普通表／事实表
f_table_1
f_table_2
...
f_table_N
表组 Table group
普通表／事实表
...
数据库-database

基本概念—数据库、表组

- **数据库(Database)**：是最高层的对象，按数据库进行资源的分配、隔离和管理，实现了多租户的管理能力。

- **维度表组**: 可发生关联的数据表的集合，通常可以按业务模块划分不同的表组。一个数据库可以包含多个表组。

维度表组：自动创建，有且只有一个，不可修改和删除，用于存放维度表。
普通表组：或称事实表组，用来存放数据量较大的表，可以创建多个

- 一般按不同业务含义的表规划到不同的表组下，方便管理
- 两阶段查询引擎下，只支持同表组的表join，MPP引擎无此限制

基本概念：表

阿里云
aliyun.com

维度表：

又称复制表 / 广播表，即表的数据将复制到每个计算节点上。数据量较小；

普通表：又称事实表，存放数据量较大的表；

更新模式：按更新方式分为：实时更新（realtime）和批量表（batch）

- realtime--支持insert和delete，面向实时更新场景；
- batch--批量更新，通过odps批量同步（全覆盖、二级分区增量模式），不支持insert/delete；

对象命名规则

阿里云
aliyun.com

数据库对象（数据库、表组、表）命名规范：

1. 长度小于**64**个字符
2. 字符规范，命名必须以字符**[a-z]**开头，可以包含的字符有：**[a-z],[0-9],**下划线‘_‘，但结束符不能是下划线‘_‘，即正则符合：**^(?![0-9_])(?!.*?_$)[a-z0-9_]+$**
3. 不能包含以下关键字：**DEBUG,WARN,ERROR,__**
4. 不能等于**AnalyticDB**内部定义的保留字
5. 大小写不敏感，自动统一转换为小写

提纲

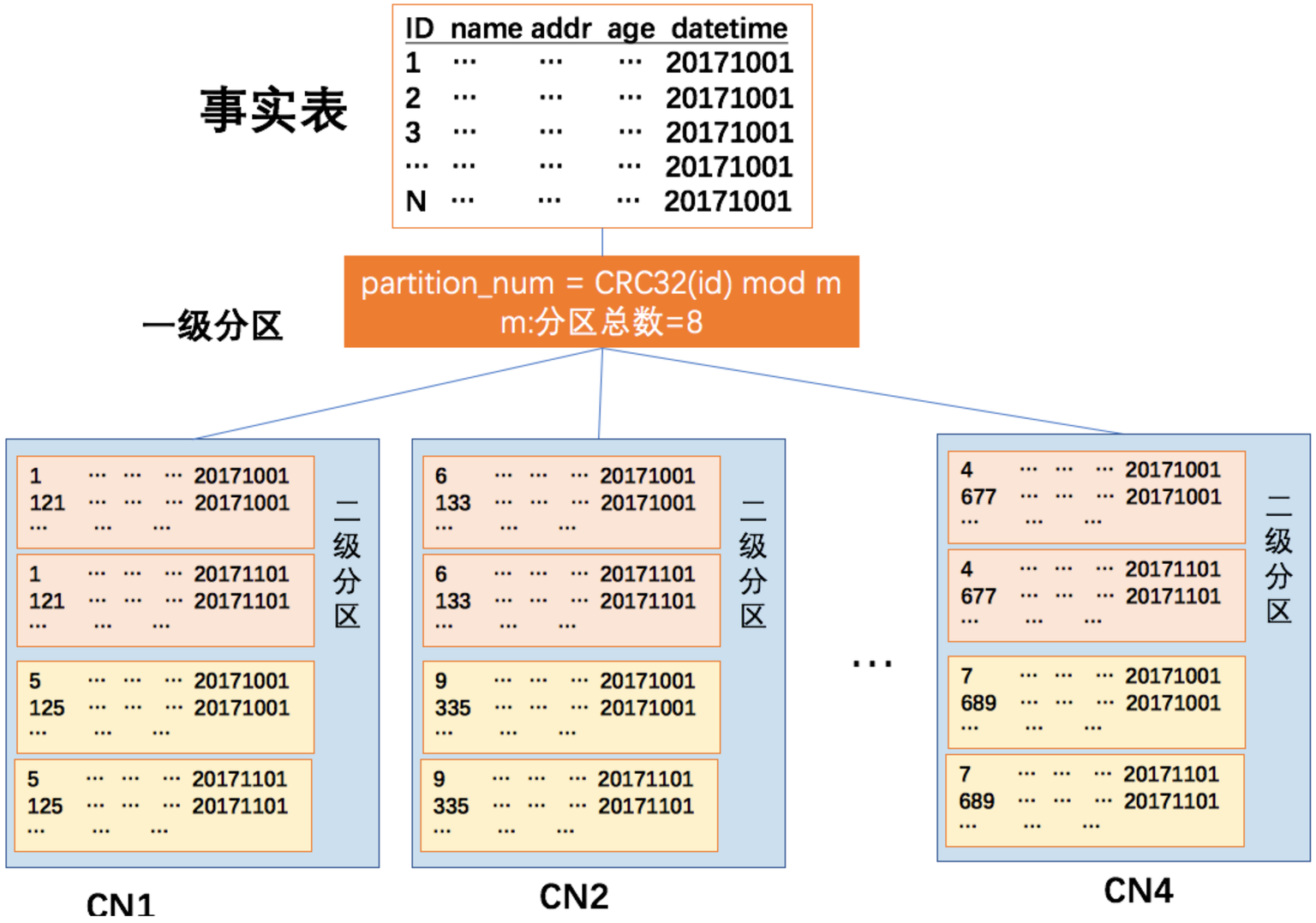
数据分布策略—事实表

支持2级分区策略

- 一级分区采用hash算法
- 二级分区采用list算法

数据分布示意说明：

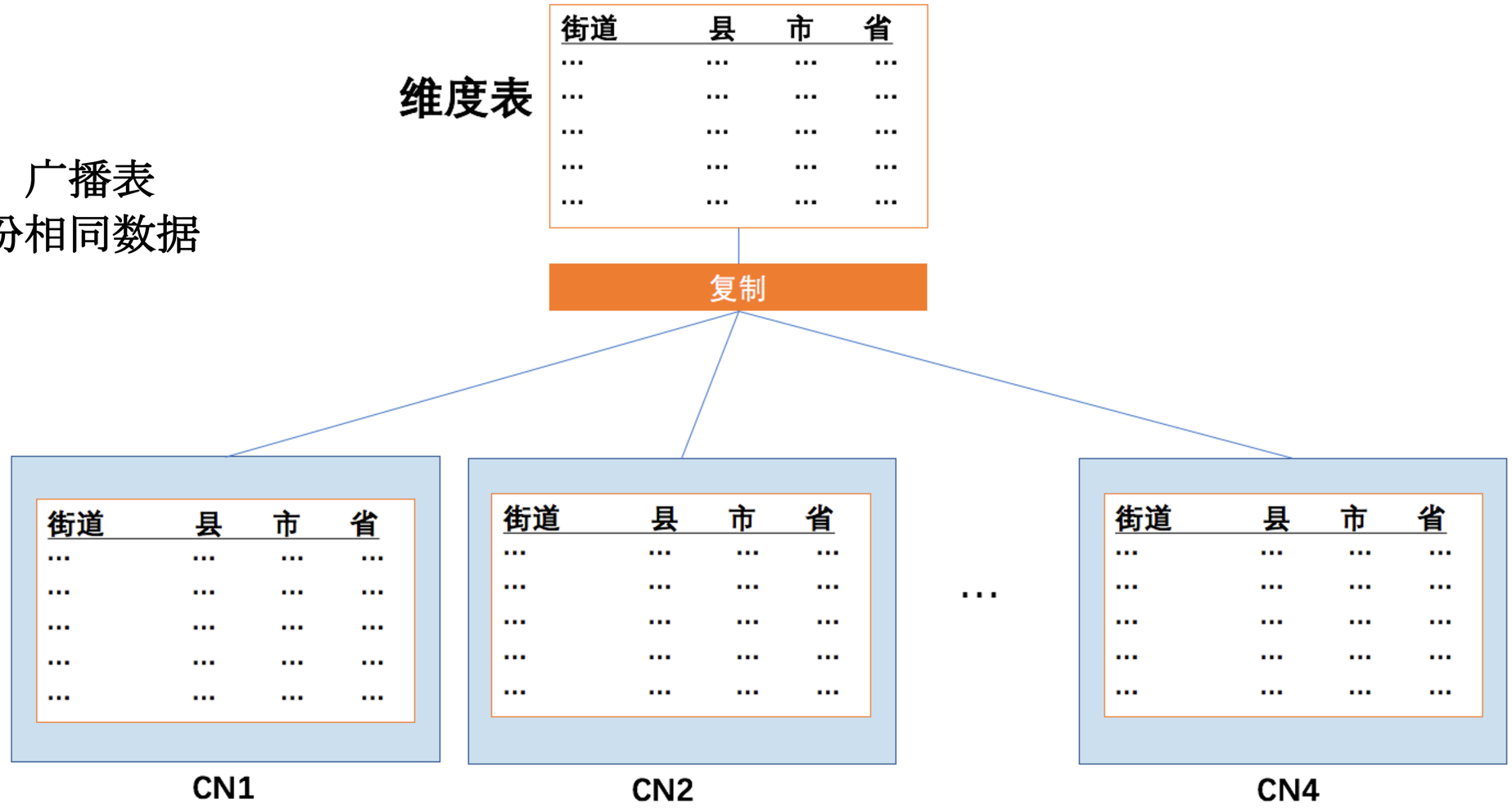
- 一级分区列：ID
- 一级分区数：8
- CN节点数：4
- 每个CN节点存储一级分区个数：2个
- 二级分区按月20171001,20171101,…



图：数据分布逻辑示意图

数据分布策略—维度表

- 维度表又称复制表、广播表
- 每个节点都复制一份相同数据



图：维度表数据分布逻辑示意图

AnalyticDB表DDL—数据分布相关

```
CREATE TABLE tab_name (
mail_id varchar COMMENT '',
scan_timestamp timestamp COMMENT '',
biz_date bigint COMMENT '',
org_code varchar COMMENT '',
org_name varchar COMMENT '',
dlv_person_name varchar COMMENT '', receiver_name varchar COMMENT '',
receiver_phone varchar COMMENT '', receiver_addr varchar COMMENT '',
product_no varchar COMMENT '', mag_no varchar COMMENT '',
PRIMARY KEY (mail_id,org_code,biz_date) )
PARTITION BY HASH KEY (org_code) PARTITION NUM 128
SUBPARTITION BY LIST KEY (biz_date)
SUBPARTITION OPTIONS (available_partition_num = 30)
CLUSTERED BY (org_code)
TABLEGROUP ads_demo
OPTIONS (UPDATETYPE='realtime') COMMENT '';
```

主键

一级分区

二级分区

聚集列

提纲

一级分区

基本原理：

- 表的一级分区采用**hash**分区
- 采用改良的CRC32算法
- 可指定任意一列作为分区列
- 不支持多列

partition_num = CRC32(id) mod m
m:分区总数=8

空值与null的处理

- 空值的hash值与字符串”-1”相同，空值过多可能导致数据不均衡(倾斜问题)
- null值情况，自动采用primary key其他非空列或者表的第一个列

一级分区列的选择

阿里云 aliyun.com

一级分区列选择依据：

- 选择值分布均匀的列，切勿选择数据倾斜的列作为分区列。
- 表join的列，如果有多个事实表（不包括维度表）进行join，选择参与join的列作为分区列。如果有多列join怎么办？可根据查询重要程度或者查询性能要求（例如某SQL的查询频率特别高），选择某列作为分区列，这样可以保证基于分区列join的查询性能具有较好的性能。
- 常用SQL包含某列的等值或in查询条件，选择该列作为分区列。如：Select … from tableName where id=123 and ….; 可以利用分区裁剪，实现高QPS。
- 选择大部分SQL都使用的join列或者where条件，有些情况不能满足所有的SQL

```
Select col1,col2 from tab1 join tab2 on tab1.id=tab2.id and ….
Select col1 from tableName where id=10 and name= ‘xx’
```

一级分区数的确认

阿里云 aliyun.com

一级分区个数评估依据:

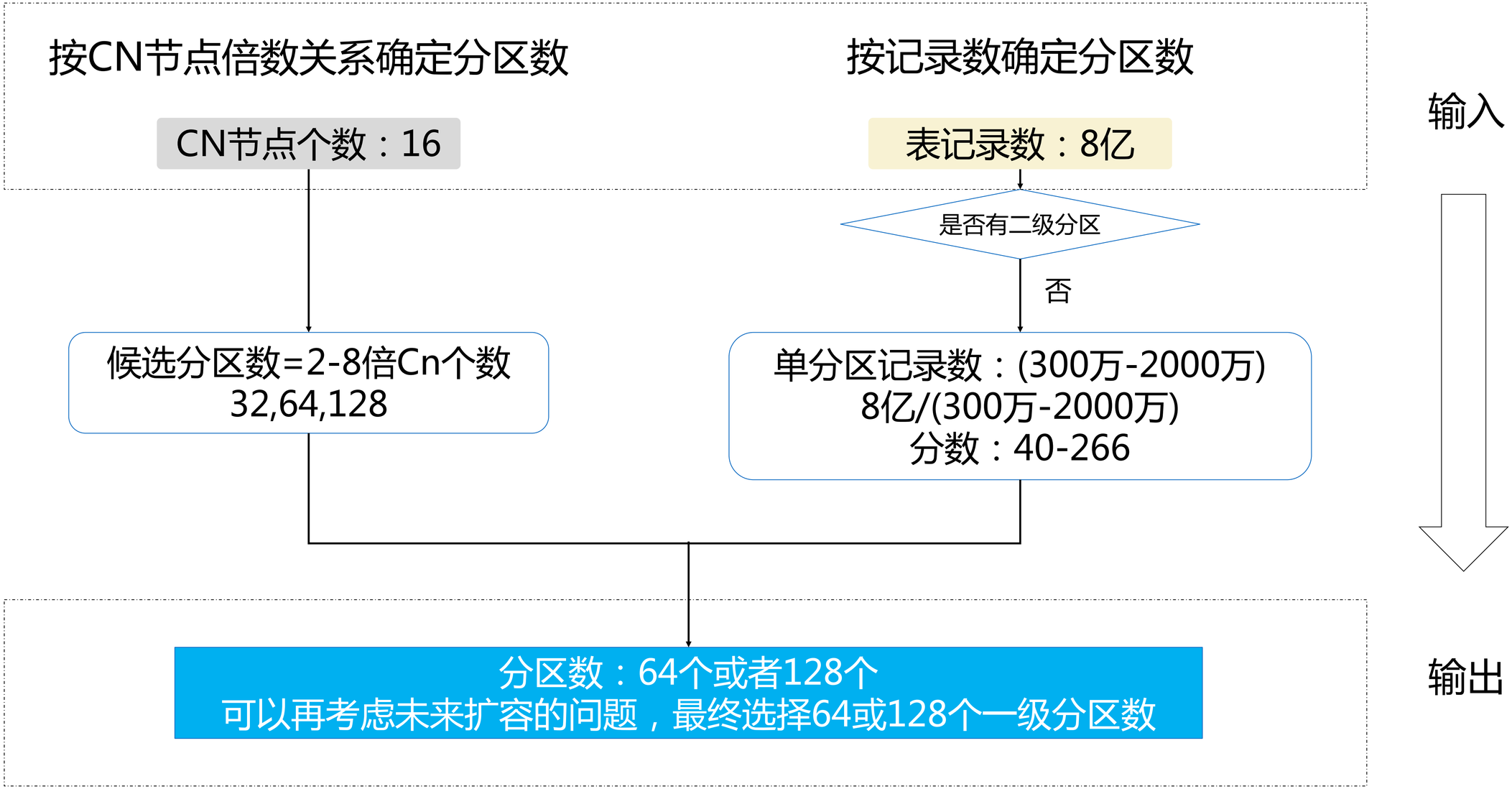
1. 如果表与其他表有join，那么join的多个事实表分区数尽量相同
2. 单分区的数据记录数建议为300万条到2000万之间。如果为二级分区，保证每个一级分区下的二级分区的记录数为300万条到2000万条之间。
3. 分区数应该是ECU数量 的倍数（2-8倍），一般是2的n次方，支撑灵活的扩容／缩容。

   --某DB为8个C8，则合理分区数可设置为： 16，32，64。

   --设置过大情况，如2个C8节点，则设置了128个分区。
4. 单表一级分区数最大值为256。在某些极其特殊的环境中，最大值为内部修改支持512。
5. 单计算节点的分区数（包括二级分区）不能超过1万。

注意:一级分区数不可修改。如需修改，必须删表重建

一级分区数计算示例


按CN节点倍数关系确定分区数
按记录数确定分区数
CN节点个数：16
表记录数：8亿
是否有二级分区
否
候选分区数=2-8倍Cn个数
32,64,128
单分区记录数：(300万-2000万)
8亿/(300万-2000万)
分数：40-266
输入
输出
分区数：64个或者128个
可以再考虑未来扩容的问题，最终选择64或128个一级分区数

提纲

- 数据库对象及命名规则
- 数据分布策略
- 一级分区
- 二级分区
- 聚集列
- 主键设计
- 数据类型
- DDL语句
- 最佳实践—案例说明
- 最佳实践—规避数据倾斜

二级分区

二级分区内部原理：

- 在一级分区的基础上，将每个一级分区再进行二次切分
- List分区，list中不同值的个数

语法形式：

- SUBPARTITION BY LIST KEY (biz_date)
- SUBPARTITIONOPTIONS(available_partition_num = 30)
- 其中biz_date为二级分区列，是一个bigint数据类型列
- 分区数为30个，可以按需修改二级分区个数

表记录情况

....	Datetime	bigint
....	2018-01-01	20180101
....	2018-01-02	20180102
....	2018-01-03	20180103

- 二级分区列，单独设计一个bigint类型
- 二级分区列的不同值的个数即是分区数
- 二级分区值需要应用端根据对应的时间映射为bigint
- 按日 / 周 / 月 / 年，应用控制：时间字段—分区列的映射

二级分区—最佳实践

何时考虑二级分区：

- 当表的记录数超过一定量（一般10亿）时
- 利用二级分区实现历史数据的自动清除

最佳实践：

- 单表二级分区数一般小于等于90
- 同时每个计算节点上总二级分区个数不超过1万个
- 每个一级分区下的二级分区包含的数据条数在300万到2000万之间

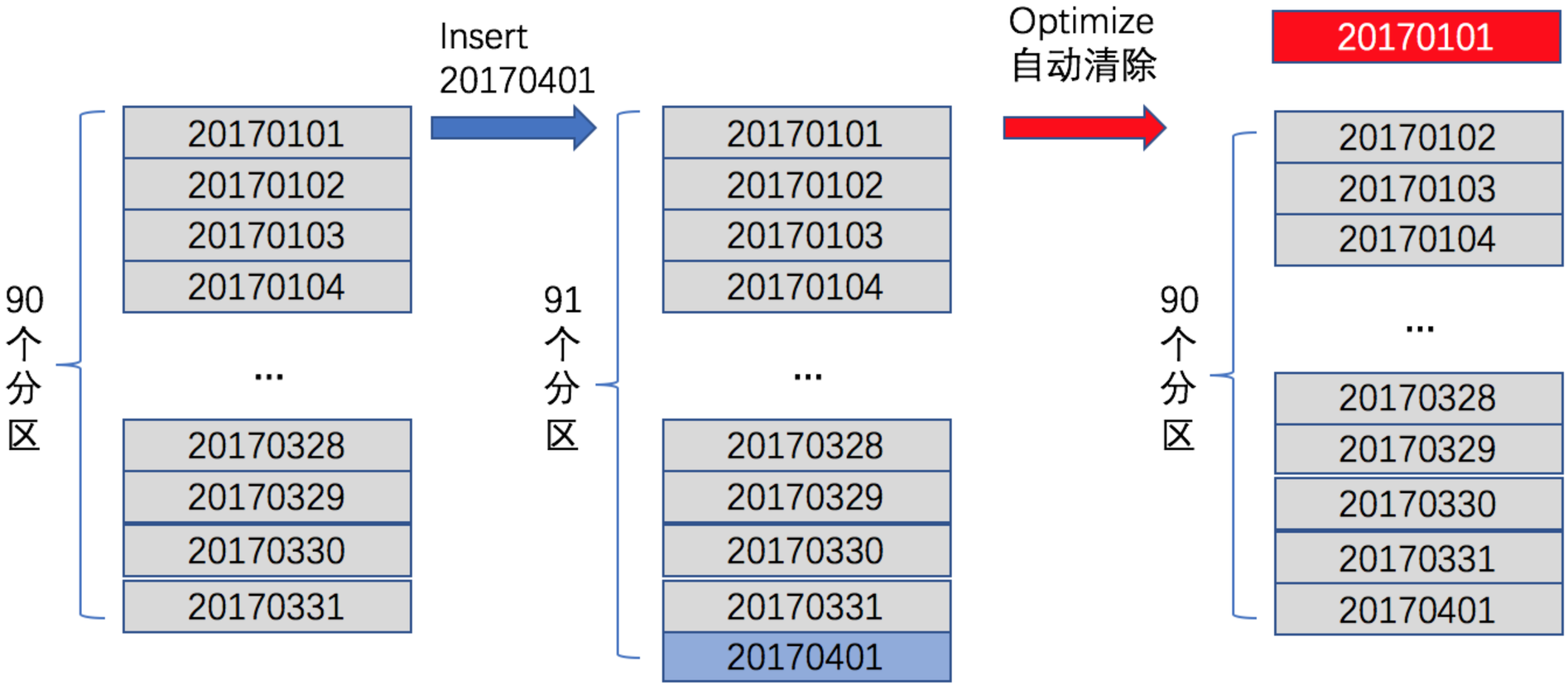
二级分区调整：

当数据存储周期发生变化情况，可以通过修改二级分区个数和分区值间隔来实现存储时间周期的动态调整

二级分区—数据清除机制

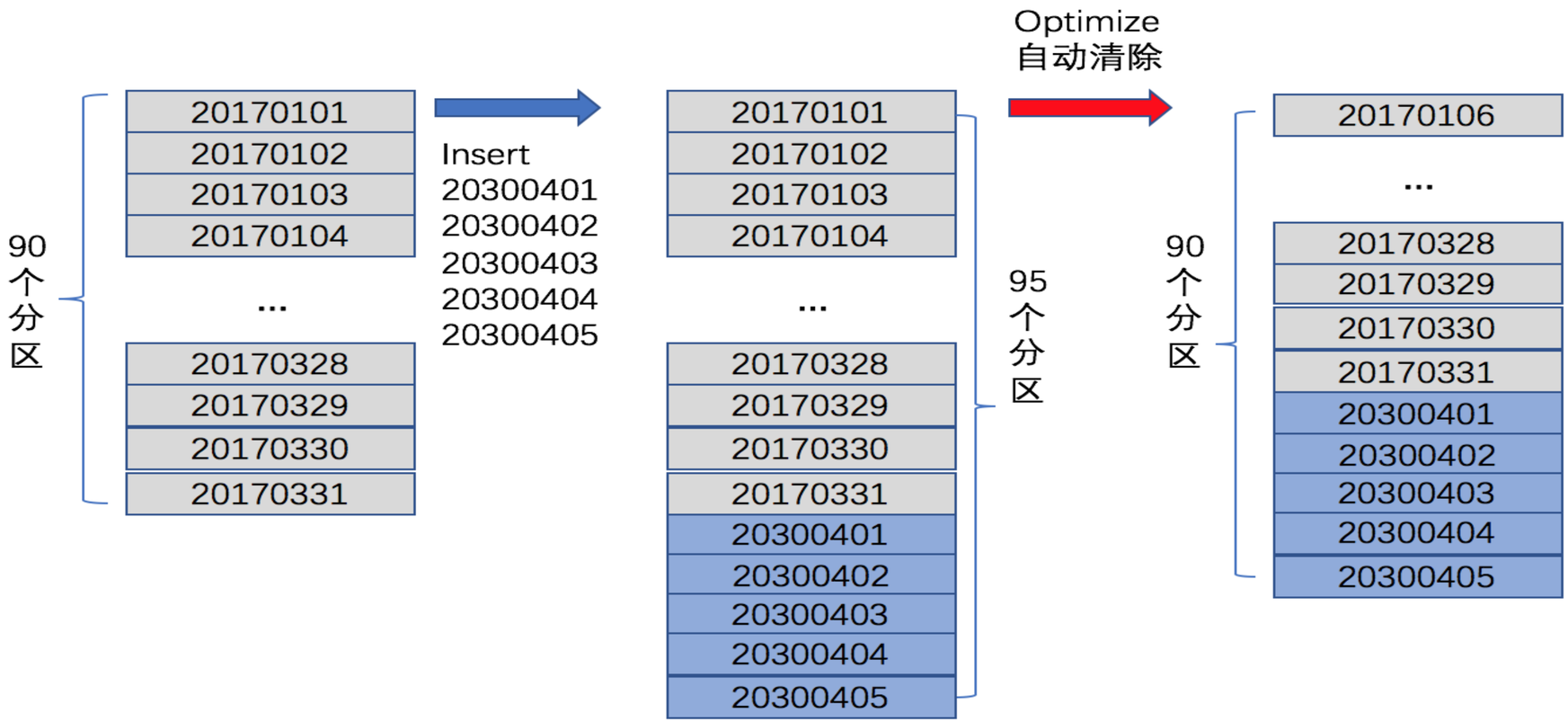
数据清理机制

- 表元数据记录的二级分区数N
- 每次optimize时记录当前表的二级分区个数M
- Optimize时根据对比 N<M，逻辑清除M-N个最小二级分区



二级分区—数据清除机制风险

由于新增5条记录，该表自动扩展出5个新的分区20300401-20300405，由于这5个新分区都是较大的值，当进行optimize操作时，将自动删除20170101-20170105的5个历史分区。



提纲

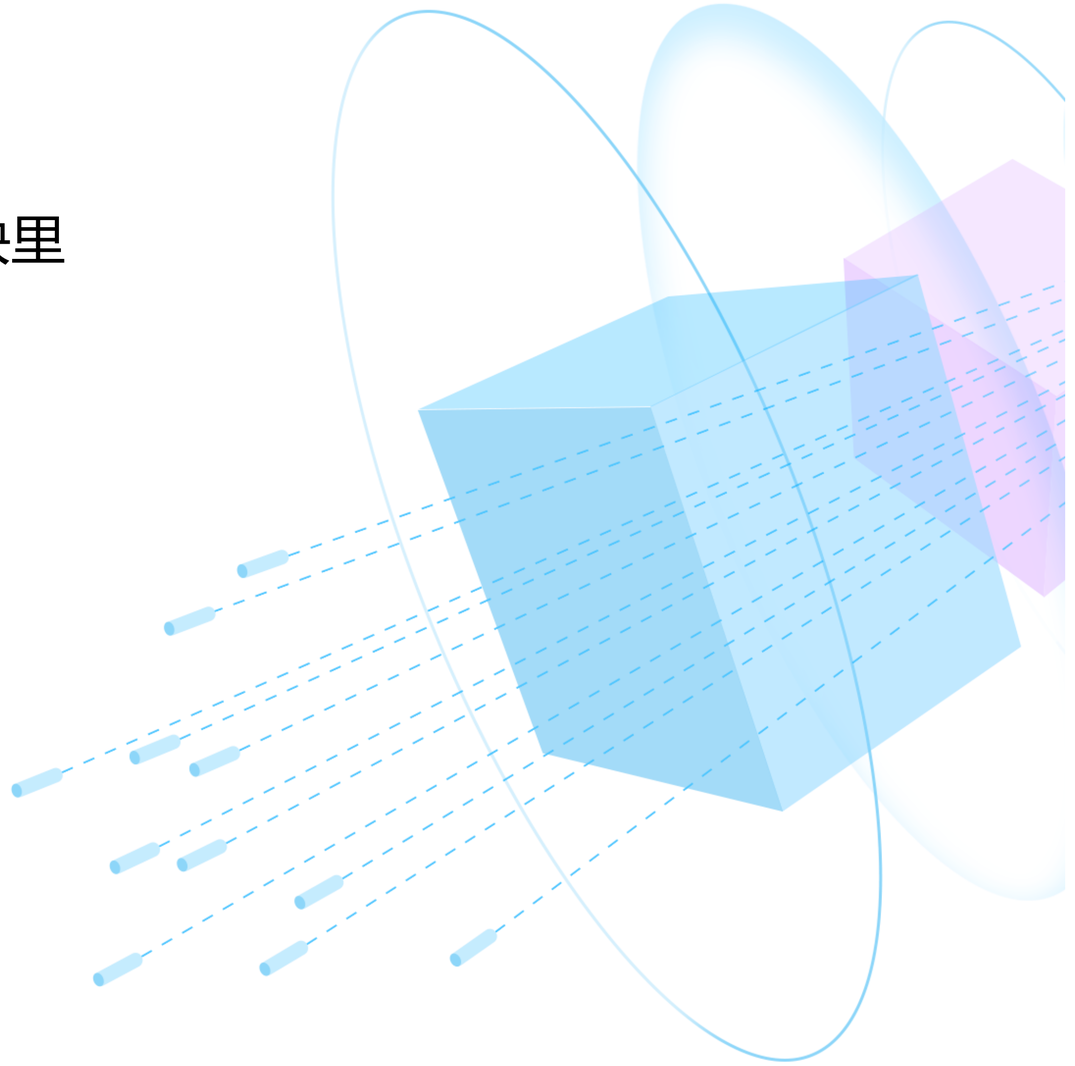
特色功能：聚集列

聚集列：

- 把列取值相同的数据，尽可能的存放在同样的区块里
- 可以指定一个列或者多个列作为聚集列
- 多个列的时候，字段顺序很重要
- 数据区分度很大的列效果更好
- 查询的条件中需要指定聚集列的内容或范围

注意：

- 通过改变物理存储位置，提高查询效率
- 聚集列不是必须的，如无必要，不要创建



聚集列

基本原理：

- 数据存储支持按一列或多列排序，保证该列值相同或相近的数据在磁盘同一位置
- 好处：当以聚集列为查询条件时，查询结果保存在磁盘相同位置，可以减少IO次数
- 一个表只能设计一个主聚集列，因此需要最合适的列/多列作为主聚集列

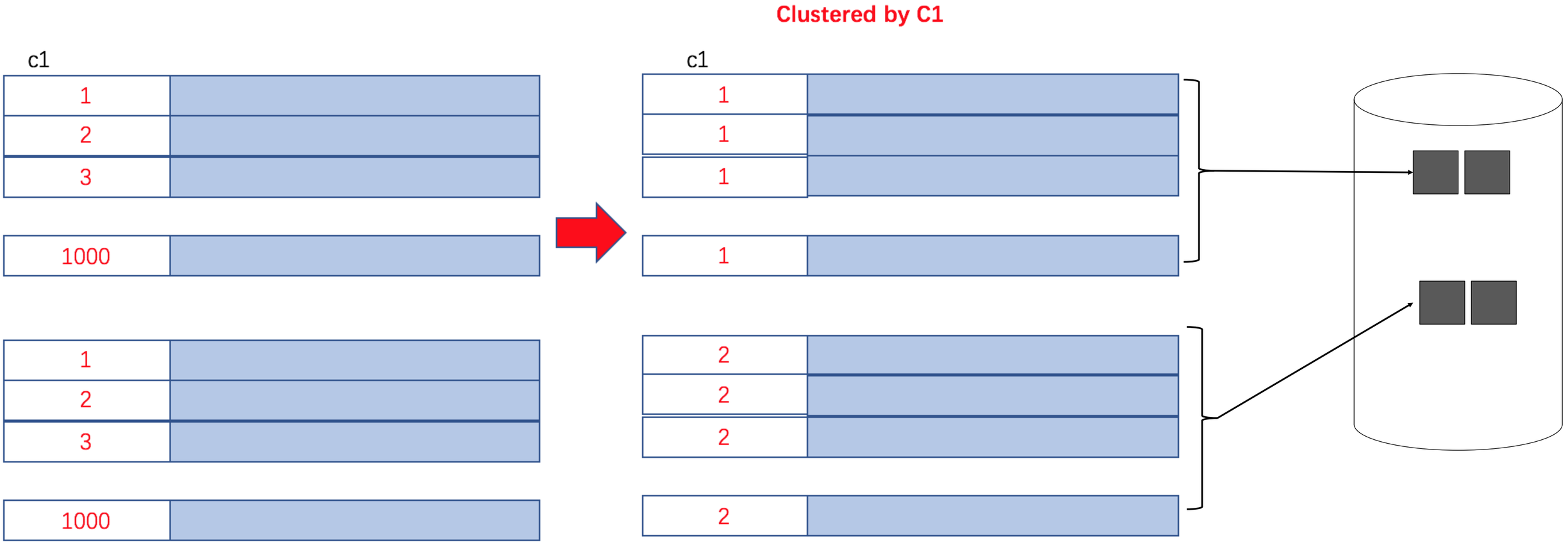
聚集列选择依据：

主要或大多数查询条件包括这一列，且该条件具有较高的筛选率。

Join 等值条件列（通常为一级分区列）作为聚集列。

聚集列—数据存储逻辑示意图

- 按c1列做聚集
- c1值相同的行，排序聚集在相同的数据块中



提纲

阿里云
aliyun.com

- 数据库对象及命名规则
- 数据分布策略
- 一级分区
- 二级分区
- 聚集列
- 主键设计
- 数据类型
- DDL语句
- 最佳实践—案例说明
- 最佳实践—规避数据倾斜

主键

- 实时表—realtime 必须包含有主键字段
- realtime表insert／delete，通过主键进行相同记录的判断，确定唯一记录
- 主键组成：(业务id+一级分区键+二级分区键)，如果表保护一级分区，二级分区，主键必须包含有一级分区列和二级分区列
- 从存储空间和insert性能考虑，一定要减少主键的字段数

- AnalyticDB的主键与通用数据库的主键在使用上有差异：
  --通用数据库除了唯一性，也自动创建index
  -- AnalyticDB数据库只是做数据唯一记录的判重 insert/delete操作

提纲

- 数据库对象及命名规则
- 数据分布策略
- 一级分区
- 二级分区
- 聚集列
- 主键设计
- 数据类型
- DDL语句
- 最佳实践—案例说明
- 最佳实践—规避数据倾斜

数据类型

据类型	类型名称	取值范围	字节数
boolean	布尔类型	0或1	1 bit
tinyint	微整数类型	-128到127	1字节
smallint	整数类型	-32768到32767	2字节
int	整数类型	-2147483648到2147483647	4字节
bigint	大整数类型	-92233720368547758O8到9223372036854775807	8字节
float	单精度浮点数	-3.402823466E+38到-1.175494351E-38, 0, 1.175494351E-38到3.402823466E+38, IEEE标准	4字节
double	双精度浮点数	-1.79769313486231 57E+308到-2.2250738585072014E-308, 0, 2.2250738585072014E-308 到 1.79769313486231 57E+308.	8字节
decimal	定精度类型	decimal(m,d)	动态
varchar	变长字符串类型	最大长度8k	变长
date	日期类型	‘1000-01-01’ 到 ‘9999-12-31’ 支持的数据格式为 'YYYY-MM-DD'	4字节
timestamp	时间戳类型	'1970-01-01 00:00:01.000' UTC 到 '2038-01-19 03:14:07.999' UTC. 目前支持的的数据格式为：'YYYY-MM-DD HH:MM:SS'	4字节
json	json	json	
clob	clob	clob	
multivalue	多值列类型		

数据类型使用示例

```
CREATE TABLE t_fact_customer (
  customer_id bigint COMMENT '',
  customer_name varchar COMMENT '',
  sex tinyint COMMENT '',
  age int  COMMENT '',
  total_buy_count BIGINT COMMENT '',
  total_buy_money float COMMENT '',
  longitude double  COMMENT '',
  latitude double  COMMENT '',
  birth_day date COMMENT '',
  phone_number varchar COMMENT '',
  home_addr varchar COMMENT '',
  first_login timestamp COMMENT '',
  buy_list multivalue,
  flag boolean,
  PRIMARY KEY (customer_id)
)
PARTITION BY HASH KEY (customer_id) PARTITION NUM 8
TABLEGROUP ads_demo
OPTIONS (UPDATETYPE='realtime')
COMMENT '';
```

```
insert into t_fact_customer VALUES
(1,
'王明',
0,
38,
1002323232222232323,
13.23,
2323.233,
2323.23,
'1980-10-01',
'11126411777',
'北京市朝阳区望京xxxx',
'2017-01-10 10:00:11',
'a,b,c',
1)
```

提纲

阿里云
aliyun.com

- 数据库对象及命名规则
- 数据分布策略
- 一级分区
- 二级分区
- 聚集列
- 主键设计
- 数据类型
- DDL语句
- 最佳实践—案例说明
- 最佳实践—规避数据倾斜

DDL—表组

创建表之前，需要创建表组

CREATE TABLEGROUP tablegroup_name;

DROP TABLEGROUP tablegroup_name;

删除表组前，需要删除表组下的所有表，如果表组下有表，提示如下错误信息。
SQL Error [18805] [HY000]: DropOperationFailedException:Can not drop non-empty tablegroup.

DDL—创建表语法

```
CREATE [DIMENSION] TABLE [db_name.]table_name (
  colname dataType [COMMENT 'col1'],
  ...,
    [primary key (colname list )]
)
[PARTITION BY HASH KEY(column name)] [PARTITION NUM N]
[SUBPARTITION BY LIST (column name)]
[SUBPARTITION OPTIONS (available_partition_num = M)]
[CLUSTERED BY (column list)]
[TABLEGROUP table_group_name]
[options (updateType='{realtime | batch}')]
[COMMENT '']
;
```

维度表，默认事实表

列定义

主键定义

一级分区定义

二级分区定义

聚集列定义

所属表组定义

表类型定义

DDL—创建表示例

```
CREATE TABLE tab_name (
mail_id varchar COMMENT '',
scan_timestamp timestamp COMMENT '',
biz_date bigint COMMENT '',
org_code varchar COMMENT '',
org_name varchar COMMENT '',
dlv_person_name varchar COMMENT '', receiver_name varchar COMMENT '',
receiver_phone varchar COMMENT '', receiver_addr varchar COMMENT '',
product_no varchar COMMENT '', mag_no varchar COMMENT '',
PRIMARY KEY (mail_id,org_code,biz_date) )
PARTITION BY HASH KEY (org_code) PARTITION NUM 128
SUBPARTITION BY LIST KEY (biz_date)
SUBPARTITION OPTIONS (available_partition_num = 30)
CLUSTERED BY (org_code)
TABLEGROUP ads_demo
OPTIONS (UPDATETYPE='realtime') COMMENT '';
```

主键

一级分区

二级分区

聚集列

表组

表类型

DDL—维度表

```
CREATE DIMENSION TABLE dim_table_name (
  col1 int  comment 'col1',
  col2 varchar comment 'col2'
  [primary key (col1)]
)
[options (updateType='{realtime | batch}')];
```

维度表默认归属到默认的维度表组，无需提前创建

DDL—智能索引

AnalyticDB默认为表所有列创建索引，无需create index

取消index -- disableIndex

参考原则：

- 只会出现在select子句中，不会在where子句中使用

```
CREATE TABLE  t_fact_orders (
  order_id varchar COMMENT '',
  customer_id varchar COMMENT '',
  goods_id bigint COMMENT '',
  numbers bigint disableIndex true COMMENT '',
  total_price double disableIndex true COMMENT '',    order_date bigint COMMENT '',
  PRIMARY KEY (order_id,customer_id,order_date)
)
PARTITION BY HASH KEY (customer_id) PARTITION NUM 16
SUBPARTITION BY LIST KEY (order_date)
SUBPARTITION OPTIONS (available_partition_num = 90)
TABLEGROUP ads_demo
OPTIONS (UPDATETYPE='realtime')
COMMENT '';
```

提纲

- 数据库对象及命名规则
- 数据分布策略
- 一级分区
- 二级分区
- 聚集列
- 主键设计
- 数据类型
- DDL语句
- 最佳实践—案例说明
- 最佳实践—规避数据倾斜

表设计--最佳实践

阿里云
aliyun.com

指导原则：

- 查询场景SQL倒推一级分区列的选择
- 一级分区列—分区裁剪
- 一级分区数—合理参考300万—2000万记录/单分区
- 多个join的事实表—相同的一级分区策略
- 二级分区—记录数一定规模方可考虑使用
- 聚集列—解决特定场景下性能问题

最佳实践—智能交通(1)

阿里云 aliyun.com

业务场景描述

城市交通--交通卡口过车监控业务
每个路口（卡口）摄像头拍摄通行车辆信息：车牌号、通过时间、卡口编号、行驶等信息

数据量及查询场景：
- 每日实时增量数据：1亿，存储5年数据
- 需要实时查询每个路口车辆通行情况，如，某一时间段，通过该路口的车辆总数，车辆详单情况
- 按车辆拍照号码查询一段时间的车辆的详单信息

一级分区列选择：车牌号或卡口编号

查询QPS分析：
- 按**车牌号码**查询一段时间的车辆的详单信息--高并发
- 需要实时查询每个**路口（卡口）**车辆通行情况—并发不多

最佳实践—智能交通(2)

阿里云 aliyun.com

建表语句

```
CREATE TABLE t_fact_vehicle_info ( uuid varchar NOT NULL
COMMENT '唯一标识', access_time varchar DEFAULT '0'
COMMENT '接收时间', gate_number varchar DEFAULT '0'
COMMENT '卡口编码', device_num varchar DEFAULT '0' COMMENT
'', pass_time varchar DEFAULT '0' COMMENT '过车时间',
vehicle_number varchar DEFAULT '' COMMENT '号牌号码',
vehicle_num_type varchar DEFAULT '' COMMENT '号牌种类', speed
double DEFAULT '' COMMENT '行驶速度', picture_url varchar,
pt_month bigint COMMENT '',
PRIMARY KEY (uuid,vehicle_number,pt_month) )
PARTITION BY HASH KEY (vehicle_number) PARTITION NUM 256
SUBPARTITION BY LIST KEY (pt_month)
 SUBPARTITION OPTIONS (available_partition_num = 61)
CLUSTERED BY (gate_number)
TABLEGROUP app_group
OPTIONS (UPDATETYPE='realtime')
```

- 按车辆号码一级分区，比较均匀的将数据分布到所有分区上
- 按卡口编号设置clusered by聚集列
- 二级分区按月存储，存储5年，61个月的数据

查询场景SQL

```
select * from t_fact_vehicle_info
where vehicle_number='xxxxx'
and pass_time between '2017-10-10 09:00:00'
and '2018-03-01 10:00:00'
and pt_month between 20171010 and 20180301;
```

- 车牌号码查询，一级分区列查询
- 分区裁剪，支持高QPS--并发上万，且可线性扩展
- 每个查询—10毫秒返回

```
select count(*) from t_fact_vehicle_info
where gate_number='xxxxx'
and pass_time between '2018-03-01 09:00:00'
and '2018-03-01 09:30:00'
and pt_month =20180301;
```

- 按卡口做统计查询
- 聚集列—卡口编号，查询性能优化提高10倍性能
- 查询性能在1秒内返回

最佳实践—电子商务(1)

阿里云 aliyun.com

业务场景描述

简化的业务场景有3个表：客户信息表、订单表、商品类型表（维度表）

数据量及查询场景：

- 客户信息表：5亿记录
- 订单表：每日新增订单5000万，存储3年数据，总数据量：550亿
- 商品类型表：100万
- 业务场景要求：统计不同客户群体，在不同时间段的统计数据，rt<1分钟

多表关联join查询

查询QPS分析：

- 客户群体分类关联订单数据进行统计
- 统计数据量比较大，需要并行计算

最佳实践—电子商务(2)

阿里云 aliyun.com

建表语句

```
CREATE TABLE  t_fact_orders (
  order_id varchar COMMENT '',
  customer_id varchar COMMENT '',
  goods_id bigint COMMENT '',
  numbers bigint COMMENT '',
  total_price double COMMENT '',
  order_time timestamp COMMENT '',
  order_date bigint COMMENT '',
  PRIMARY KEY (order_id,customer_id,order_date)
)
PARTITION BY HASH KEY (customer_id) PARTITION NUM 128
SUBPARTITION BY LIST KEY (order_date)
SUBPARTITION OPTIONS (available_partition_num = 36)
TABLEGROUP ads_demo
OPTIONS (UPDATETYPE='realtime')
COMMENT '';
```

- 事实表采用相同的一级分区**(join 列)**，且一级分区数相同

```
CREATE TABLE t_fact_customers (
  customer_id varchar COMMENT '',
  customer_name varchar COMMENT '',
  phone_number varchar COMMENT '',
  address varchar COMMENT '',
  last_login_time timestamp COMMENT '',
  age int COMMENT '',
  PRIMARY KEY (customer_id))
PARTITION BY HASH KEY (customer_id) PARTITION NUM 128
TABLEGROUP ads_demo
OPTIONS (UPDATETYPE=‘realtime’) COMMENT '';
```

```
CREATE DIMENSION TABLE t_dim_goods (
    goods_id bigint  comment  '',
    price double comment '',
    class bigint comment '',
    name VARCHAR comment '',
    update_time  TIMESTAMP comment '',
    primary key (goods_id)
)
OPTIONS (UPDATETYPE='realtime') ;
```

最佳实践—电子商务(3)

阿里云 aliyun.com

查询场景SQL

统计一段时间内，按年龄段统计订单的客户数量、订单销售总额

```
SELECT
case when cus.age<=30 then '<20' when cus.age>20
and cus.age<=30 then '20-30' when cus.age>30
and cus.age<=40 then '30-40' else '>40' end as age_range,
COUNT(distinct cus.customer_id),
SUM(total_price)
FROM
t_fact_customers cus left join t_fact_orders ord
on  cus.customer_id=ord.customer_id
where ord.order_time>='2017-10-01 00:00:00' and
ord.order_time<'2017-11-01 00:00:00'
AND ord.order_date=201710
group by age_range;
```

- **事实表join 一级分区列join**
- **并行计算**
- **Localjoin**

统计一段时间内，按商品种类统计订单的客户数量、订单销售总额

```
SELECT
goods.class,
COUNT(distinct cus.customer_id),
SUM(total_price)
FROM
t_dim_goods goods left join t_fact_orders ord
on  cus.goods_id=ord.goods_id
where ord.order_time>='2017-10-01 00:00:00' and
ord.order_time<'2017-11-01 00:00:00'
AND ord.order_date=201710
group by goods.class;
```

- **维度表与事实表join**
- **并行计算**
- **Localjoin**

最佳实践—气象监控(1)

阿里云 aliyun.com

业务场景描述

气象监控：全国数十万以上的气象监控站，每分钟采集一次气象数据

数据量及查询场景：
- 每分钟：10万+，存储100年数据（永久存储），1000亿+记录
- 查询某一个气象站10年或者60年的历史曲线数据
- 查询全部气象站某一时刻或者一段时间的统计数据

聚集列的加速，查询跨大时间范围的数据

查询分析：
- 查询单一站，设计为分区列查询，查询很长时间数据，需要考虑按站号进行聚集
- 同时考虑单一时间点所有的气象站的查询，需要设置合理的数据块大小
- 需要综合考虑两种类型查询性能

最佳实践—气象监控(2)

阿里云 aliyun.com

建表语句

```
CREATE TABLE t_fact_device_info ( record_id varchar NOT NULL
COMMENT '记录标识',
d_time varchar DEFAULT '0' COMMENT '数据时间',
 device_id varchar DEFAULT '0' COMMENT '气象站号',
V_1 double,v_2 double,v_3 double …..
pt_year bigint COMMENT '',
PRIMARY KEY (record_id,device_id,d_time,pt_year) )
PARTITION BY HASH KEY (device_id) PARTITION NUM 256
SUBPARTITION BY LIST KEY (pt_year)
 SUBPARTITION OPTIONS (available_partition_num = 100)
CLUSTERED BY (device_id)
TABLEGROUP app_group
OPTIONS (UPDATETYPE='realtime')
```

- 按气象站号一级分区，比较均匀的将数据分布到所有分区上
- 按气象站号设置clusered by聚集列
- 二级分区按年存储，存储100年
- 调整数据块大小到1M大小，兼顾两种查询性能

查询场景

```
select sum(v_1),max(v_1),avg(v1) from t_fact_device_info
where device_id='xxxxx'
and d_time between '2010-01-01 00:00:00'
and '2018-10-01 00:00:00'
;
```

- 查询某一气象站—一级分区列查询，支持高QPS
- 需要查询很长时间范围数据—需要聚集列的加速，提速100倍

```
select device_id,v_1,v_2 v_3 from t_fact_device_info
where
d_time =  '2018-10-01 00:00:00'
;
```

- 查询某一时刻，所有站的详细数据，返回数据量较大
- 分布式计算，数据分布在所有节点上
- 此类查询QPS一分钟<10个

提纲

- 数据库对象及命名规则
- 数据分布策略
- 一级分区
- 二级分区
- 聚集列
- 主键设计
- 数据类型
- DDL语句
- 最佳实践—案例说明
- 最佳实践—规避数据倾斜

最佳实践—规避数据倾斜—危害

阿里云 aliyun.com

数据倾斜带来的问题：

- 存储溢出，AnalyticDB每个计算节点分配了相同的存储空间，每个节点的存储空间也有基线，数据倾斜后，最直接的问题就是出现某些节点磁盘爆满，而无法继续写入数据，而其他节点有大量的可用空间。
- 计算长尾，由于每个节点的数据量不一样，数据量多的节点计算需要的I／O请求次数、内存大小、CPU、网络开销都远大于平均值，导致查询性能慢，甚至查询超时。

Notes:当出现数据倾斜时，直接导致业务中断，查询超时等故障，解决方案只能是重建表，重新导入数据。而当表记录数数十亿、上百亿、甚至千亿时，重建的代价非常大，需要中断业务较长时间。所以，必须在开始建表的时候需要充分理解业务数据分布情况，选择合理的分区键，否则在业务上线后带来业务中断问题，同时解决成本非常高。

最佳实践—规避数据倾斜—原因

大部分数据倾斜都归因于在设计、规划表时缺乏对业务数据分布真实情况的了解，以及缺少对AnalyticDB数据分布机制要求以及带来的严重后果的足够重视。

常见数据倾斜情况分为如下4种情况：

1. 一级分区列存在数据不均衡，例如：采用按省份代码进行分区，而不同省份的业务数据差异性非常大。
2. 沿用以前系统的分区策略，例如：采用按月份分区，容易出现查询数据集中在某一分区上。
3. 空值过多，如果一级分区列值包含大量‘ ’，容易导致‘ ’分区倾斜。‘ ’和null是有区别的，对于null 值AnalyticDB内部会自动根据主键的第一个非分区列进行hash。
4. 没有经过数据仓库建模（雪花、星型模型），而只是照搬交易型数据库建模

最佳实践—规避数据倾斜

1.了解表对应的业务，选择能将数据分布均匀的一级分区列

应该选择：身份证号、卡号、车牌号、设备号、手机号码、订单号码、机构编号等

不应该选择：月份、归属地（省份）、种类等

2.抽样数据分布调研

一级分区列不同值个数，select count(distinct 一级分区列) from tab；不同值个数过小一般不适合做一级分区

通过样例数据调研下数据分布情况，select 一级分区列, count(*) from tab group by 一级分区列 order by count(*) desc; 是否存在严重不均衡情况

3.空值‘’替换为null

AnalyticDB对null值的处理，自动选择primary key的第一个非分区键进行二次hash，但是‘’只能按‘’进行分区。

Thanks!

咨询邮箱：ADB_SUPPORT@service.alibaba.com