



# 深入数据库同步技术研究

Welcome to Bank of Tomorrow

presented by 许增伟

# 从RDB和NoSQL谈起

RDB vs. NoSQL

**RDB(Relational Database : 关系型数据库)**

联想词：二维表、主外键、事务、范式、索引、锁、SQL、Log、MVCC、MySQL、OLTP/OLAP、MPP...



互相融合

**NoSQL(No SQL -> Not Only SQL)**

联想词：K/V、列簇、文档、缓存、图形...

# 数据同步涉及的业务场景



.Apache Sqoop



.灾备  
.双主数据库  
.数据仓库

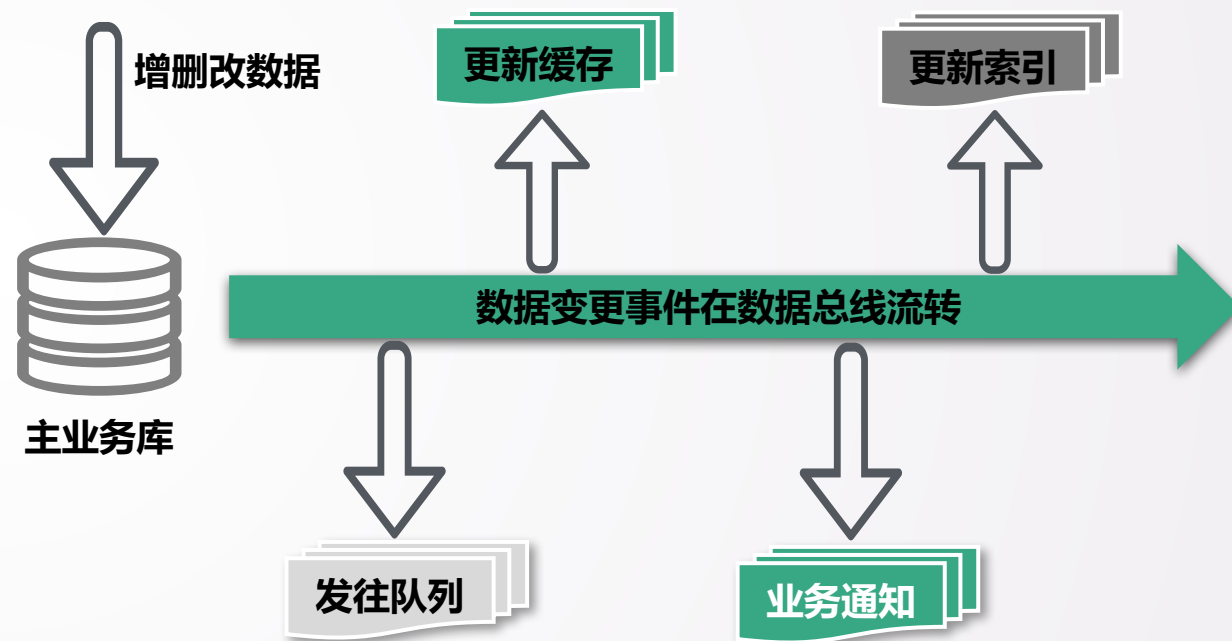


图1：构建数据总线(Data Bus)

# 数据同步的基本分类

我们可以从不同维度对数据同步进行分类



# 一个好的同步工具应该具备的特征

特征	描述
易用性	简单来讲，就是傻瓜化。
高性能	这里涉及到批处理操作，以及分析innodb聚簇索引的特点，尽量减少目的端B树索引结构的移动，减少i/o压力。
及时性	当源数据发生改变时，能否用尽量少的时间传播到目的端，从而达到较好的同步及时性。
可见性	通过某种机制，可以查看同步引擎执行每一个步骤的具体过程、耗时和结果，出现问题可以迅速定位。
健壮性	分布式环境下，一切都是不可靠的。比如源和目的挂掉怎么办，出现网络抖动怎么办，容器突然crash重启后能否恢复上次的同步状态等等。
异构性	可以满足多个异构系统/平台间的数据同步工作。
扩展性	基于拦截器机制，支持用户自定义扩展操作
时序性	需要把源数据按照数据发生变更的时间先后顺序，同步到目的端。

以上几个问题在本次分享中或多或少都有提及，其中时序性是本次分享的重点

# 数据同步的常见痛点

## 1.不同种类数据库字段的类型映射

获取  
表字段  
元信息

Oracle :

```
SELECT utc.column_id, utc.column_name, t_pk.position, utc.data_type,  
utc.data_length, utc.char_length, utc.data_precision, utc.data_scale, utc.nullable  
FROM user_tab_columns utc,  
(  
  select ucc.column_name, ucc.position  
  from user_constraints uc, user_cons_columns ucc  
  where uc.table_name = ucc.table_name  
  and uc.constraint_name = ucc.constraint_name  
  and uc.constraint_type = 'P' and uc.table_name = ?  
) t_pk  
where utc.table_name = ?  
and utc.column_name = t_pk.column_name(+)  
order by t_pk.position asc, utc.column_id asc;
```

MySQL : `information_schema.columns`


[https://docs.oracle.com/cd/E12151\\_01/doc.150/e12155/oracle\\_mysql\\_compared.htm#g1034154](https://docs.oracle.com/cd/E12151_01/doc.150/e12155/oracle_mysql_compared.htm#g1034154)

# 数据同步的常见痛点

## 2.源表主键要求

- Oracle : (1)有主键 (2)隐性主键(rowid)
- MySQL : (1)有主键

## 3.游标(Cursor)

- 客户端游标(Client Side Cursor)
  - 服务端游标(Server Side Cursor)  JDBC URL中添加useCursorFetch=true
- ```
PreparedStatement ps = conn.prepareStatement(sql,
ResultSet.TYPE_FORWARD_ONLY, ResultSet.CONCUR_READ_ONLY);
ps.setFetchSize(500); //可以根据业务情况调整fetch size大小
ps.setFetchDirection(ResultSet.FETCH_FORWARD);
```

## 4.源表分段处理

联合主键处理？

- Oracle :

```
select max(id) from (
select id from (select id from t1 where id >= ? order by id asc)
where rownum <= ?
)
```
- MySQL :

```
select max(id) from (
select @rownum:=@rownum+1 as rn, id
from (select @rownum:=0) r, t1
where id >= ? order by id asc
) t where rn <= ?
```



# 数据同步的常见痛点

## 5.增量数据判断：不靠谱的last\_modified\_date字段

```
alter table t add column last_modified_date timestamp not null default  
current_timestamp on update current_timestamp;
```

为什么不靠谱？last\_modified\_date反映了事务的**操作时间**(什么时候执行的插入/更新操作)，而非生效时间(**事务commit时间**)，这两个时间间隔可以相差很久。

怎么办？

- ✓ 对last\_modified\_date做出补偿，将时间往后推移
- ✓ 对于Oracle，我们有更好的ora\_rowscn伪列可以利用

注意：如果在创建Oracle表时未指定rowdependencies，则默认ora\_rowscn伪列是基于Oracle块的，这意味着，即使只插入/更新/删除一条记录，位于同块上的其他记录的ora\_rowscn也会变化，这会造成成百上千条无效记录被查询出来同步到目的表，既浪费了服务器资源，又浪费了网络带宽。

ora\_rowscn的另外一个问题是全表扫描的问题。

## 6.提升同步性能

- ✓ 查询源表时明确指定按主键排序
- ✓ 批量插入/更新/删除
- ✓ 最后提交
- ✓ /\*+parallel(table\_name n)\*/
- ✓ 临时禁用约束和索引

➡ JDBC URL中添加rewriteBatchedStatements=true



# 数据同步的常见痛点

## 7. 插入还是更新，这是一个问题

Oracle:

```
MERGE INTO table_name alias1
USING (table|view|sub_query) alias2 ON (join condition)
WHEN MATCHED THEN
    UPDATE table_name
    SET col1 = col_val1, col2 = col_val2
WHEN NOT MATCHED THEN
    INSERT (column_list) VALUES (column_values);
```

MySQL:

```
INSERT INTO table(column_list)
VALUES(value_list)
ON DUPLICATE KEY UPDATE column_1 = values(column_1),
column_2 = values(column_2) ...;
```

## 8. 源表有物理删除怎么办 启动数据比对线程，在目的表中删除目的表有而源表没有的记录

## 9. 恼人的编码问题：collation和utf8mb4 ALTER TABLE t CHARACTER SET = utf8mb4 , COLLATE = utf8mb4\_bin ;

## 10. 转义 { Oracle : " " 转移符 MySQL : ` ` 转移符

# 时序性探讨

## 1.时序性及其重要意义

什么是时序性？时序性就是事件发生的先后顺序。对于数据库来讲，事件的最小单元就是事务(即 Transaction，每提交一个事务就产生一个事件，无论事务涉及的数据量有多大或者参与的表有多少)。

作为一个正常的OLTP数据库，可能每时每刻在每张表上都在发生事务性操作。类似于事件溯源(Event Sourcing)机制，我们希望将源库上的事务按照发生的时间先后顺序（也就是事务的提交顺序），依次应用在目的库上，这样就会保证源表和目的表数据状态的一致性和事务的不可撕裂性。

所以，数据库的**时序性**有两个显著特征：一是以事务为单位，二是保持事务发生的先后顺序。

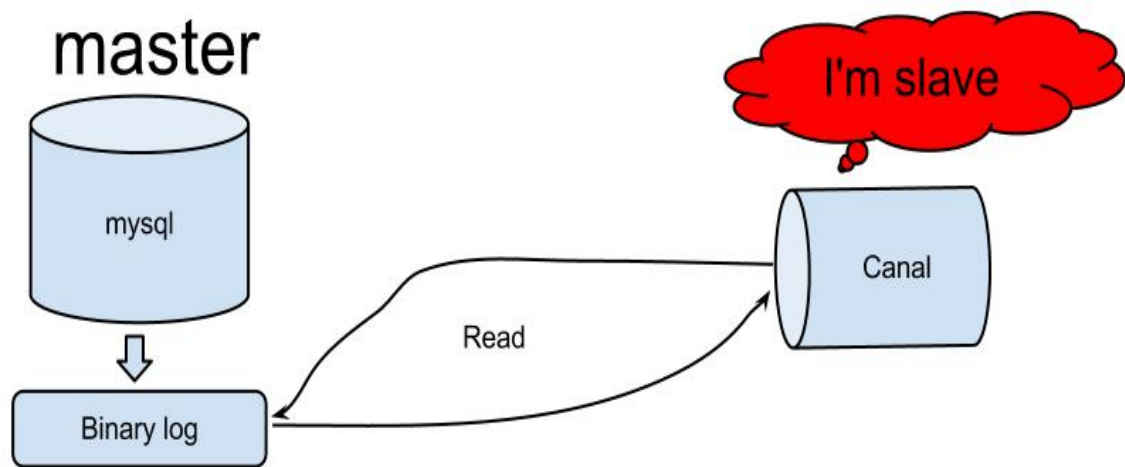
实现时序性的意义：

- ✓ 避免在目的库产生“子表数据先于父表数据”存在
- ✓ 实现源库级别的数据快照
- ✓ 是构建数据总线(Data Bus)的基础和先决条件

# 时序性探讨

## 2.方案选型

2.1 如果同步源是MySQL，我们采用了阿里开源的Otter/Canal框架，其原理就是伪装成MySQL的Slave。



**Otter的github地址：**

<https://github.com/alibaba/otter>

**Canal的github地址：**

<https://github.com/alibaba/canal>

图1：Canal框架原理图（来自官方）

# 时序性探讨

## 2.方案选型

2.2 如果同步源是Oracle，则实现时序性有几种流派。

### 日志挖掘

可以使用免费的Logminer，对在线日志或归档日志进行挖掘，从而解析出Redo SQL在目的库进行重放(Replay)，还可以使用商业版的Oracle GoldenGate，这两种方式都是基于日志挖掘方式，其缺点是需要数据库上进行诸多配置，对数据库不透明。

### 触发器

可以使用LinkedIn的Databus，它基于trigger+ora\_rowscn机制，在源表上添加trigger，需要部署多个存储过程包，侵入数据库更深。同时因为基于触发器，对源表写入有性能影响，也增加了发生死锁的可能性，部署这套东西会造成DBA的反感。

### 物化视图

比如阿里的yugong(愚公)开源项目，需要在每张源表上开启物化视图日志，并在目的库创建dblink和对应源表的物化视图，刷新物化视图读取增量更新，物化视图可以看做是变相的trigger，也不是特别好。

# 时序性探讨

## 2.方案选型

### 2.3 我们自研了da-syncer同步工具

以上针对Oracle同步源的解决方案都不适合我们，因为我们想在纯应用层搞定时序性的问题(只要开放select权限就行)，对数据库透明，无需在数据库上动刀增加额外配置，更不需要DBA参与。

我们强调纯应用层的解决方案，不提倡在数据库层面有大动作，这是因为：

- ① 数据库是别人的，不是你想加字段就加字段，想加trigger就加trigger，想改配置就改配置的；
- ② DBA资源是需要跨团队协调的，包括使用选型方案产生的后续维护工作；
- ③ 数据库出问题的时候(比如死锁了)，你是有潜力当背锅侠的；
- ④ 当项目假手于人，需要太多外部资源扶持，你的项目可能是遥遥无期的；

总的来说，凡事要靠自己，项目需要介入扶持的资源越多，就越具有不可控性。

# 时序性探讨

## 3.Oracle闪回版本查询简介

- ① Oracle闪回版本查询是Oracle10g+提供的允许你查询一张表距当前时间之前**有限时间段内**行记录事务版本的一种机制，它本身基于Oracle的SCN。 **那么问题来了，MySQL怎么闪回？**
- ② SCN(System Change Number)是Oracle的系统变更编号，它是一个长整形数，作为标识Oracle对象发生变化的编号，自动增长，全局使用。这里我们主要讨论SCN作为事务提交编号的场景。

```
create table test
(
  id number(38,0) not null,
  name varchar2(20) not null,
  primary key(id)
);

insert into test(id, name) values(1, 'a1');
update test set name = 'a2' where id = 1;
delete from test where id = 1;
insert into test(id, name) values(2, 'b1');
update test set name = 'b2' where id = 2;
```



```
select id, name, rawtohex(versions_xid) versions_xid,
       versions_operation, versions_startscn, versions_endscn,
       to_char(versions_starttime,'yyyy-mm-dd hh24:mi:ss')
       versions_starttime,
       to_char(versions_endtime,'yyyy-mm-dd hh24:mi:ss')
       versions_endtime
from test
versions between scn minvalue and maxvalue
order by versions_startscn asc;
```

# 时序性探讨

## 3.Oracle闪回版本查询简介

| ID | NAME | VERSIONS_XID     | VERSIONS_OPERATION | VERSIONS_STARTSCN | VERSIONS_ENDSCN | VERSIONS_STARTTIME  | VERSIONS_ENDTIME    |
|----|------|------------------|--------------------|-------------------|-----------------|---------------------|---------------------|
| 1  | 1 a1 | 0300130042600B00 | I                  | 8289572283        | 8289572969      | 2018-07-29 22:59:16 | 2018-07-29 22:59:29 |
| 2  | 1 a2 | 1500050021412200 | U                  | 8289572969        | 8289573094      | 2018-07-29 22:59:29 | 2018-07-29 22:59:38 |
| 3  | 1 a2 | 15000B00D1412200 | D                  | 8289573094        | (null)          | 2018-07-29 22:59:38 | (null)              |
| 4  | 2 b1 | 2A000000A8400100 | I                  | 8289573239        | 8289573317      | 2018-07-29 22:59:50 | 2018-07-29 22:59:56 |
| 5  | 2 b2 | 15000C0076352200 | U                  | 8289573317        | (null)          | 2018-07-29 22:59:56 | (null)              |

### 观察上图我们可以得出一些基本结论：

- ✓ 版本这个概念属于记录，随着记录上发生事务提交(插入、更新和删除)，这条记录的新版本不断形成。
- ✓ 伪列versions\_startscn和versions\_starttime是版本的开始或者说形成时候的SCN和时间戳，因为只有通过提交事务才能形成版本，所以它们也代表了事务提交时的SCN和时间戳。
- ✓ 伪列versions\_endscn和versions\_endtime是版本的结束SCN和时间戳。
- ✓ 从以上可知，每个版本都有生命周期，占据记录版本时间线上的某一段时间段。所以当前版本的versions\_endscn和versions\_endtime恰好是下个版本的versions\_startscn和versions\_starttime。
- ✓ 伪列versions\_xid和versions\_operation是形成版本时的事务ID和发生的操作(I=Insert, U=Update, D=Delete，你没看错，Delete也可以被检索出来)。
- ✓ 形成该版本时的用户字段值以及上述伪列字段可以随闪回版本查询一起被检索出来。



# 时序性探讨

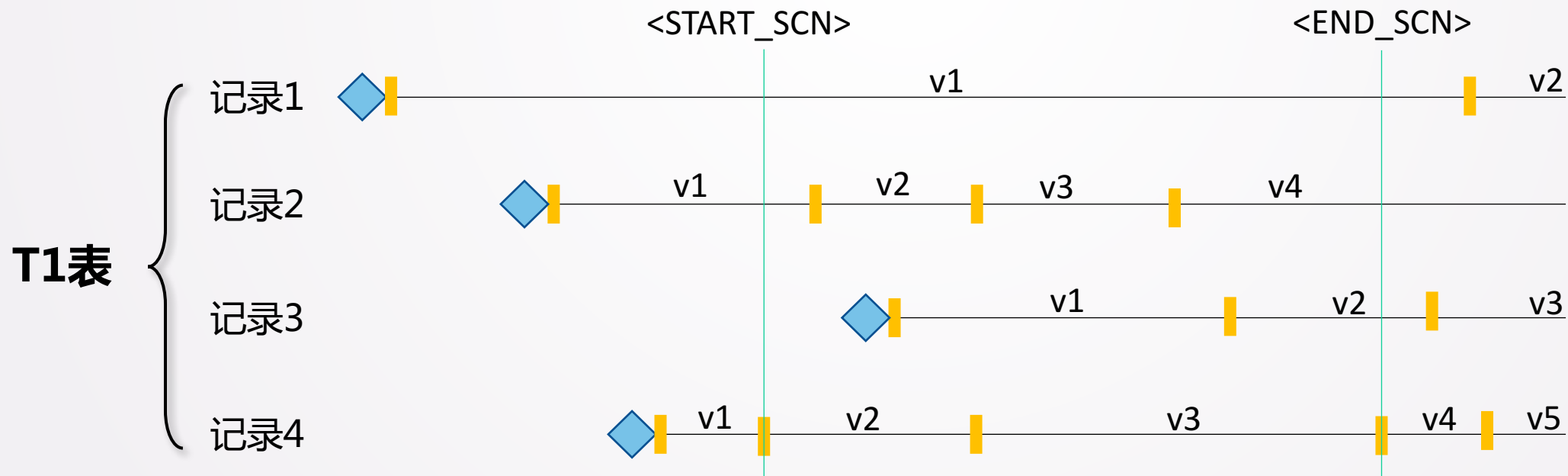
## 3.Oracle闪回版本查询简介

接下来看一下闪回版本查询起关键作用的过滤语句：

*versions between scn <START\_SCN> and <END\_SCN>*

<START\_SCN>和<END\_SCN>分别指明了查询版本时的开始、结束SCN条件值。如果两个条件分别指定了minvalue和maxvalue，则会把目前Oracle内部回滚段内所有可用的记录版本全部查询出来。

假设表t1的4条记录在时间线上的版本分布图如下，**蓝色菱形**代表记录插入，**黄色短竖线**代表版本开始或结束，**绿色长竖线**代表我们指定的<START\_SCN>和<END\_SCN>查询条件。



# 时序性探讨

## 3.Oracle闪回版本查询简介

- ✓ 无论指定任何的<START\_SCN>和<END\_SCN>, 至少会查询出一条记录的某一个版本。
- ✓ 记录1会查询出v1, 所有伪列字段都为null, 因为v1的versions\_startscn比 <START\_SCN>更靠前, versions\_endscn比<END\_SCN>更靠后。
- ✓ 记录2会查询出v1且v1的versions\_startscn为null, 因为v1的versions\_startscn比<START\_SCN>更靠前; 会查询出完整v2和v3; 会查询出v4且v4的versions\_endscn为null, 因为v4后续的新版本还未形成。
- ✓ 记录3会查询出完整v1; 会查询出v2且v2的versions\_endscn为null, 因为v2的versions\_endscn比<END\_SCN>更靠后。
- ✓ 记录4比较特殊, 因为v1的versions\_endscn(也是v2的versions\_startscn)正好等于<START\_SCN>, v3的versions\_endscn(也是v4的versions\_startscn)正好等于<END\_SCN>。在这种情况下会查询出完整的v2和v3, 会查询出v4且v4的versions\_endscn为null。
- ✓ 假如现在有一个需求, 要求查询出<START\_SCN>和<END\_SCN>之间形成的版本(即记录在该SCN间隔内有过事务提交行为), 且要求保持时序性的话, 我们只需要添加过滤条件:  
**versions\_startscn >= <START\_SCN> and versions\_startscn != <END\_SCN>**, 则记录2的v2、v3、v4, 记录3的v1、v2, 记录4的v2、v3这些符合要求的版本都会被过滤出来。

注意: 上图没有引入Delete操作, Delete操作形成的版本表现会稍有不同。

# 时序性探讨

## 3.Oracle闪回版本查询简介

在进行闪回版本查询时，一些需要特别关注的点和使用限制：

(1). 除了versions between scn <START\_SCN> and <END\_SCN>语法，还可以使用versions between timestamp <START\_TIME> and <END\_TIME>，SCN和TIMESTAMP在Oracle内部有映射，且通过scn\_to\_timestamp和timestamp\_to\_scn函数可以相互转换，但因为Oracle内部实际使用的是SCN机制，所以推荐使用SCN进行闪回版本查询。

(2). 随着时间的流逝，之前可以查询到某条记录的版本数据可能会逐步消失，这是因为版本数据不会在Oracle中永久留存，它依赖于Oracle多个参数的设置和动态因素，比如：

undo\_management: undo管理模式是否为auto。

undo\_retention: 版本数据在undo表空间的保留时长。

undo\_tablespace: 使用到的表空间，表空间大小。

可以使用show parameter显示以上参数当前的设置值。

如果在版本查询中undo表空间被其他事务覆盖，会出现错误：

ORA-01555: snapshot too old: rollback segmeng number xxx with name xxx too small

# 时序性探讨

## 3.Oracle闪回版本查询简介

(3). START\_SCN和END\_SCN不能是随意的整型数，必须是一个合法的SCN，否则：

**ORA-08181:** specified number is not a valid system change number

可以使用select timestamp\_to\_scn(systimestamp) from dual;获取当前时间戳对应的SCN。

(4). 即便是合法的SCN，START\_SCN也不能太旧，必须在undo\_retention参数指定的时间范围内，否则：

**ORA-30052:** invalid lower limit snapshot expression

(5). 如果表上发生了DDL操作(比如添加/删除字段，Truncate等)，进行闪回版本查询会出现以下错误：

**ORA-01466:** unable to read data - table definition has changed

(6). 伪列ora\_rowscn和versions\_startscn/versions\_endscn的关系

伪列ora\_rowscn是Oracle块上所在记录最后发生事务提交的记录版本的versions\_startscn值，versions\_startscn值可能会消失，但是ora\_rowscn伪列不会。

选项(2). (3). 和(4). 提醒我们：我们需要在尽量靠近当前时间域内进行闪回版本查询。

# 时序性探讨

## 4.如何利用Oracle闪回版本查询实现时序性

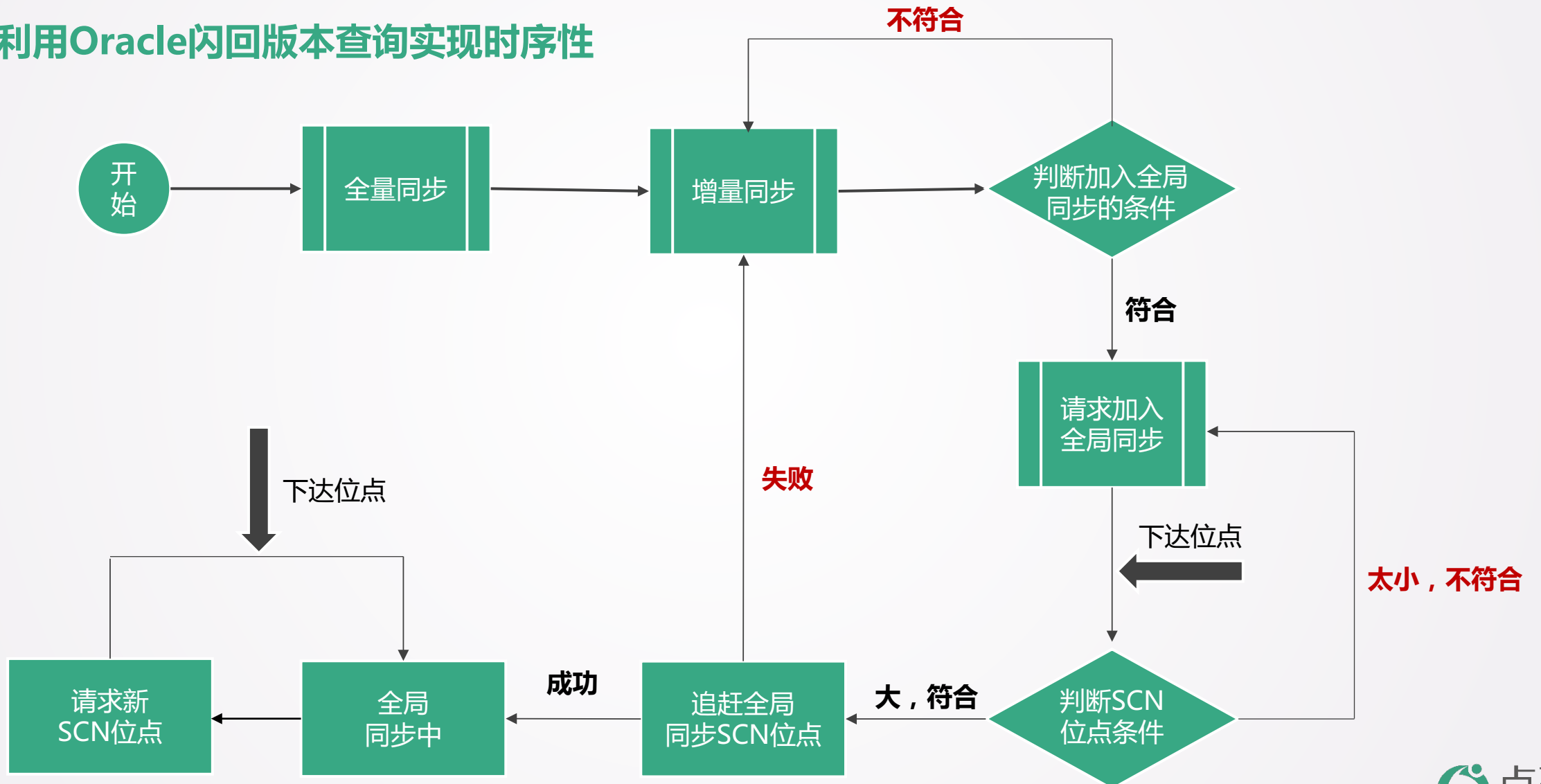
我们已经在单张表t1上实现了时序性，在多张表上如何实现？

|     | sn1 | sn2 | sn3 | sn4 | ..... | snn |
|-----|-----|-----|-----|-----|-------|-----|
| T1表 |     |     |     |     |       |     |
| T2表 |     |     |     |     |       |     |
| T3表 |     |     |     |     |       |     |
| T4表 |     |     |     |     |       |     |
| ⋮   |     |     |     |     |       |     |
| Tn表 |     |     |     |     |       |     |

问题：你想要数据一致性还是真正的数据时序性？

# 时序性探讨

## 4.如何利用Oracle闪回版本查询实现时序性



# THANKS FOR WATCHING



点融黑帮



# 关注3306π



社区公众号



社区QQ群