历史拉链表实战

历史拉链表是一种数据模型，主要是针对数据仓库设计中表存储数据的方式而定义的。所谓历史拉链表，就是指记录一个事物从开始一直到当前状态的所有变化信息。拉所有记录链表可以避免按每一天存储造成的海量存储问题，同时也是处理缓慢变化数据的一种常见方式。

应用场景

现假设有如下场景：一个企业拥有5000万会员信息，每天有20万会员资料变更，需要在数仓中记录会员表的历史变化以备分析使用，即每天都要保留一个快照供查询，反映历史数据的情况。在此场景中，需要反映5000万会员的历史变化，如果保留快照，存储两年就需要2X365X5000W条数据存储空间，数据量为365亿，如果存储更长时间，则无法估计需要的存储空间。而利用拉链算法存储，每日只向历史表中添加新增和变化的数据，每日不过20万条，存储4年也只需要3亿存储空间。

实现步骤

在拉链表中，每一条数据都有一个生效日期(effective_date)和失效日期(expire_date)。假设在一个用户表中，在2019年11月8日新增了两个用户，如下表所示，则这两条记录的生效时间为当天，由于到2019年11月8日为止,这两条就还没有被修改过，所以失效时间为一个给定的比较大的值，比如：3000-12-31

member_id	phoneno	create_time	update_time
10001	13300000001	2019-11-08	3000-12-31
10002	13500000002	2019-11-08	3000-12-31

第二天(2019-11-09)，用户10001被删除了，用户10002的电话号码被修改成13600000002.为了保留历史状态，用户10001的失效时间被修改为2019-11-09，用户10002则变成了两条记录，如下表所示：

member_id	phoneno	create_time	update_time
10001	13300000001	2019-11-08	2019-11-09
10002	13500000002	2019-11-08	2019-11-09
10002	13600000002	2019-11-09	3000-12-31

第三天(2019-11-10),又新增了用户10003，则用户表数据如小表所示：

member_id	phoneno	create_time	update_time
10001	13300000001	2019-11-08	2019-11-09
10002	13500000002	2019-11-08	2019-11-09

10002	13600000002	2019-11-09	3000-12-31
10003	13300000006	2019-11-10	3000-12-31

如果要查询最新的数据，那么只要查询失效时间为3000-12-31的数据即可，如果要查11月8号的历史数据，则筛选生效时间<= 2019-11-08并且失效时间>2019-11-08的数据即可。如果查询11月9号的数据，那么筛选条件则是生效时间<=2019-11-09并且失效时间>2019-11-09

表结构

- MySQL源member表

```
CREATE TABLE member(
             member_id VARCHAR ( 64 ),
             phoneno VARCHAR ( 20 ),
             create_time datetime,
             update_time datetime );
```

- ODS层增量表member_delta,每天一个分区

```
CREATE TABLE member_delta
            (member_id string,
             phoneno string,
             create_time string,
             update_time string)
PARTITIONED BY (DAY string);
```

- 临时表

```
CREATE TABLE member_his_tmp
            (member_id string,
             phoneno string,
             effective_date date,
             expire_date date
             );
```

- DW层历史拉链表

```
CREATE TABLE member_his
            (member_id string,
             phoneno string,
             effective_date date,
             expire_date date);
```

Demo数据准备

2019-11-08的数据为：

member_id	phoneno	create_time	update_time
10001	13500000001	2019-11-08 14:47:55	2019-11-08 14:47:55
10002	13500000002	2019-11-08 14:48:33	2019-11-08 14:48:33
10003	13500000003	2019-11-08 14:48:53	2019-11-08 14:48:53
10004	13500000004	2019-11-08 14:49:02	2019-11-08 14:49:02

2019-11-09的数据为：其中蓝色代表新增数据，红色代表修改的数据

member_id	phoneno	create_time	update_time
10001	13500000001	2019-11-08 14:47:55	2019-11-08 14:47:55
10002	13600000002	2019-11-08 14:48:33	2019-11-09 14:48:33
10003	13500000003	2019-11-08 14:48:53	2019-11-08 14:48:53
10004	13500000004	2019-11-08 14:49:02	2019-11-08 14:49:02
10005	13500000005	2019-11-09 08:54:03	2019-11-09 08:54:03
10006	13500000006	2019-11-09 09:54:25	2019-11-09 09:54:25

2019-11-10的数据：其中蓝色代表新增数据，红色代表修改的数据

member_id	phoneno	create_time	update_time
10001	13500000001	2019-11-08 14:47:55	2019-11-08 14:47:55
10002	13600000002	2019-11-08 14:48:33	2019-11-09 14:48:33
10003	13500000003	2019-11-08 14:48:53	2019-11-08 14:48:53
10004	13600000004	2019-11-08 14:49:02	2019-11-10 14:49:02
10005	13500000005	2019-11-09 08:54:03	2019-11-09 08:54:03
10006	13500000006	2019-11-09 09:54:25	2019-11-09 09:54:25
10007	13500000007	2019-11-10 17:41:49	2019-11-10 17:41:49

全量初始装载

在启用拉链表时，先对其进行初始装载，比如以2019-11-08为开始时间，
那么将MySQL源表全量抽取到ODS层member_delta表的2018-11-08的分区中，
然后初始装载DW层的拉链表member_his

```
INSERT overwrite TABLE member_his
```

```
SELECT
   member_id,
   phoneno,
   to_date ( create_time ) AS effective_date,
  '3000-12-31'
FROM
        member_delta
WHERE
        DAY = '2019-11-08'
```

查询初始的历史拉链表数据

member_his.member_id	member_his.phoneno	member_his.effective_date	member_his.expire_date
10001	13500000001	2019-11-08	3000-12-31
10002	13500000002	2019-11-08	3000-12-31
10003	13500000003	2019-11-08	3000-12-31
10004	13500000004	2019-11-08	3000-12-31

增量抽取数据

每天，从源系统member表中，将前一天的增量数据抽取到ODS层的增量数据表member_delta对应的分区中。这里的增量需要通过member表中的创建时间和修改时间来确定，或者使用sqoop job监控update时间来进行增联抽取。
比如，本案例中2019-11-09和2019-11-10为两个分区，分别存储了2019-11-09和2019-11-10日的增量数据。
2019-11-09分区的数据为:

member_delta.member_id	member_delta.phoneno	member_delta.create_time	member_delta.update_time	member_delta.day
10002	13500000002	2019-11-08 14:48:33	2019-11-09 14:48:33	2019-11-09
10005	13500000005	2019-11-09 08:54:03	2019-11-09 08:54:03	2019-11-09
10006	13500000006	2019-11-09 09:54:25	2019-11-09 09:54:25	2019-11-09

2019-11-10分区的数据为：

member_delta.member_id	member_delta.phoneno	member_delta.create_time	member_delta.update_time	member_delta.day
10004	13600000004	2019-11-09 14:48:33	2019-11-10 14:49:02	2019-11-10
10007	13500000007	2019-11-10 17:41:49	2019-11-10 17:41:49	2019-11-10

增量刷新历史拉链数据

- 2019-11-09增量刷新历史拉链表
  将数据放进临时表

```
INSERT overwrite TABLE member_his_tmp
SELECT *
FROM
  (
-- 2019-11-09增量数据，代表最新的状态，该数据的生效时间是2019-11-09，过期时间为3000-12-31
-- 这些增量的数据需要被全部加载到历史拉链表中
SELECT member_id,
       phoneno,
```

```
        '2019-11-09' effective_date,
                     '3000-12-31' expire_date
   FROM member_delta
   WHERE DAY='2019-11-09'
   UNION ALL
-- 用当前为生效状态的拉链数据，去left join 增量数据，
-- 如果匹配得上，则表示该数据已发生了更新，
-- 此时，需要将发生更新的数据的过期时间更改为当前时间.
-- 如果匹配不上，则表明该数据没有发生更新，此时过期时间不变
SELECT a.member_id,
       a.phoneno,
       a.effective_date,
       if(b.member_id IS NULL, to_date(a.expire_date), to_date(b.day)) expire_date
   FROM
     (SELECT *
      FROM member_his
    ) a
   LEFT JOIN
     (SELECT *
      FROM member_delta
      WHERE DAY='2019-11-09') b ON a.member_id=b.member_id)his
```

将数据覆盖到历史拉链表

```
INSERT overwrite TABLE member_his
SELECT *
FROM member_his_tmp
```

查看历史拉链表

member_his.member_id	member_his.phoneno	member_his.effective_date	member_his.expire_date	
10001	13500000001	2019-11-08	3000-12-31	
10002	13500000002	2019-11-08	2019-11-09	
10002	13500000002	2019-11-09	3000-12-31	
10003	13500000003	2019-11-08	3000-12-31	
10004	13500000004	2019-11-08	3000-12-31	
10005	13500000005	2019-11-09	3000-12-31	
10006	13500000006	2019-11-09	3000-12-31	

- 2019-11-10增量刷新历史拉链表

将数据放进临时表

```
INSERT overwrite TABLE member_his_tmp
SELECT *
FROM
  (
-- 2019-11-10增量数据，代表最新的状态，该数据的生效时间是2019-11-10，过期时间为3000-12-31
-- 这些增量的数据需要被全部加载到历史拉链表中
SELECT member_id,
       phoneno,
       '2019-11-10' effective_date,
                    '3000-12-31' expire_date
   FROM member_delta
   WHERE DAY='2019-11-10'
   UNION ALL
```

```
-- 用当前为生效状态的拉链数据，去left join 增量数据，
-- 如果匹配得上，则表示该数据已发生了更新，
-- 此时，需要将发生更新的数据的过期时间更改为当前时间.
-- 如果匹配不上，则表明该数据没有发生更新，此时过期时间不变
SELECT a.member_id,
       a.phoneno,
       a.effective_date,
       if(b.member_id IS NULL, to_date(a.expire_date), to_date(b.day)) expire_date
   FROM
     (SELECT *
      FROM member_his
    ) a
   LEFT JOIN
     (SELECT *
      FROM member_delta
      WHERE DAY='2019-11-10') b ON a.member_id=b.member_id)his
```

查看历史拉链表

member_his.member_id	member_his.phoneno	member_his.effective_date	member_his.expire_date
10001	13500000001	2019-11-08	3000-12-31
10002	13500000002	2019-11-09	3000-12-31
10003	13500000003	2019-11-08	3000-12-31
10004	13500000004	2019-11-08	2019-11-10
10004	13600000004	2019-11-10	3000-12-31
10005	13500000005	2019-11-09	3000-12-31
10006	13500000006	2019-11-09	3000-12-31
10007	13500000007	2019-11-10	3000-12-31

将以上脚本封装成shell调度的脚本

```
#!/bin/bash

#如果是输入的日期按照取输入日期；如果没输入日期取当前时间的前一天
if [ -n "$1" ] ;then
        do_date=$1
else
        do_date=`date -d "-1 day" +%F`
fi

sql="

INSERT overwrite TABLE member_his_tmp
SELECT *
FROM
  (
-- 2019-11-10增量数据，代表最新的状态，该数据的生效时间是2019-11-10，过期时间为3000-12-31
-- 这些增量的数据需要被全部加载到历史拉链表中
SELECT member_id,
       phoneno,
       '$do_date' effective_date,
       '3000-12-31' expire_date
   FROM member_delta
   WHERE DAY='$do_date'
   UNION ALL
-- 用当前为生效状态的拉链数据，去left join 增量数据，
-- 如果匹配得上，则表示该数据已发生了更新，
-- 此时，需要将发生更新的数据的过期时间更改为当前时间.
-- 如果匹配不上，则表明该数据没有发生更新，此时过期时间不变
SELECT a.member_id,
       a.phoneno,
```

```
       a.effective_date,
       if(b.member_id IS NULL, to_date(a.expire_date), to_date(b.day)) expire_date
   FROM
     (SELECT *
      FROM member_his
 ) a
   LEFT JOIN
     (SELECT *
      FROM member_delta
      WHERE DAY='$do_date') b ON a.member_id=b.member_id)his;
"

$hive -e "$sql"
```