3.5 快速处理 MySQL 重复数据小妙招

作者：杨涛涛

正好最近在帮客户从达梦数据库迁移到 MySQL。我也来简单说说重复数据的处理。

存放在数据库中的数据分为三种：

1. 一种是经过严格意义过滤出来的数据。比如程序端过滤数据源、数据库端在表字段上设置 check 标记过滤数据源、设置触发器过滤、调用存储过程过滤等等；
2. 另一种是原始的没有经过任何处理的数据。比如程序端代码异常导致产生非正常的想要的数据、数据库端没有设置任何过滤规则的数据保留等等。这样会产生一系列垃圾数据，当然也包含了我今天要说的重复的数据；
3. 最后一种是 SQL 语句在执行过程中可能产生的重复数据。比如两表外联，总会产生一系列 NULL。

今天我要说的重复数据，不包含 SQL 语句在执行中产生的重复数据，只包含了原始重复数据的处理。接下来，用几个经典的场景来说下。

一、记录完全重复，这其实是最最简单的去重场景。

比如无主键的表 d1

```
mysql-(ytt/3305)->show create table d1\G
*************************** 1. row ***************************
       Table: d1
Create Table: CREATE TABLE `d1` (
  `r1` int(11) DEFAULT NULL,
  `r2` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
1 row in set (0.00 sec)
```

记录数总共为四百万。

```
mysql-(ytt/3305)->select count(*) from d1 limit 2;
+----------+
| count(*) |
+----------+
|  4000000 |
+----------+
1 row in set (0.18 sec)
```

可以看到足足有四分之三的记录是重复的。

```
mysql-(ytt/3305)->select count(distinct r1,r2) from d1 ;
```

```
+-----------------------+
| count(distinct r1,r2) |
+-----------------------+
|               1000000 |
+-----------------------+
1 row in set (2.68 sec)
```

比如记录（1，1）就有四条。

```
mysql-(ytt/3305)-> select * from db1 order by r1,r2 limit 5;
+------+------+
| r1   | r2   |
+------+------+
|    1 |    1 |
|    1 |    1 |
|    1 |    1 |
|    1 |    1 |
|    2 |    2 |
+------+------+
5 rows in set (1.65 sec)
```

这种去重非常简单，要么在数据库层做，要么把数据导出来筛选好在导到数据库里来。

在数据库里做，无非就是新建一张克隆表，完了把正常数据筛选出来，再重新命名后，删掉原来的表，步骤也不是非常繁琐，例子如下：

```
mysql-(ytt/3305)->create table d2 like d1;
Query OK, 0 rows affected (0.01 sec)
```

时间主要耗费在去重并且插入新表这里

```
mysql-(ytt/3305)->insert into d2 select distinct r1,r2 from d1;
Query OK, 1000000 rows affected (19.40 sec)
Records: 1000000  Duplicates: 0  Warnings: 0


mysql-(ytt/3305)->alter table d1 rename to d1_bak;
Query OK, 0 rows affected (0.00 sec)
mysql-(ytt/3305)->alter table d2 rename to d1;
Query OK, 0 rows affected (0.00 sec)
mysql-(ytt/3305)->drop table d1_bak;
Query OK, 0 rows affected (0.00 sec)
```

上面总共花了大概 20 秒的样子，再来看看在系统层面上去重，先导出数据，

```
mysql>select * from db1 into outfile '/var/lib/mysql-files/d1.txt';
Query OK, 4000000 rows affected (1.84 sec)
```

系统层面去重，用 OS 自带的工具 sort 和 uniq。

```
root@ytt-pc:/var/lib/mysql-files# time cat d1.txt |sort -g  |uniq > d1_uniq.txt

real    0m7.345s
user    0m7.528s
sys     0m0.272s
```

导入到原表，

```
mysql-(ytt/3305)->truncate table d1;
Query OK, 0 rows affected (0.05 sec)
root@ytt-pc:/var/lib/mysql-files# mv d1_uniq.txt  d1.txt
```

把处理好的数据直接导入到数据库

```
root@ytt-pc:/home/ytt/scripts# time mysqlimport -uytt -pytt -P3305 -h 127.0.0.1   --
    use-threads=2 -vvv ytt /var/lib/mysql-files/d1.txt
mysqlimport: [Warning] Using a password on the command line interface can be
    insecure.
Connecting to 127.0.0.1
Selecting database ytt
Loading data from SERVER file: /var/lib/mysql-files/d1.txt into d1
ytt.d1: Records: 1000000  Deleted: 0  Skipped: 0  Warnings: 0
Disconnecting from 127.0.0.1

real    0m3.272s
user    0m0.012s
sys     0m0.008s
```

看下处理好的记录，

```
mysql-(ytt/3305)->select * from d1 where 1 order by r1,r2 limit 2;
+------+------+
| r1   | r2   |
+------+------+
|    1 |    1 |
|    2 |    2 |
+------+------+
2 rows in set (0.40 sec)
```

OS 层面稍微效率高些，总体包括数据导出，数据去重，数据导入，差不多是数据库层时间的一半。

二、和第一种类似，不同的是表有主键，但是其他的字段记录值是重复的。

举个例子，表 d4 除了加了主键，其他的记录和之前的一模一样。记录如下：

```
mysql-(ytt/3305)->select * from d4 order by r1,r2 limit 5;
+---------+------+------+
| id      | r1   | r2   |
+---------+------+------+
|       1 |    1 |    1 |
| 3000001 |    1 |    1 |
| 2000001 |    1 |    1 |
| 1000001 |    1 |    1 |
|       2 |    2 |    2 |
+---------+------+------+
5 rows in set (1.08 sec)
```

但是这种一般就得需要和具体的业务商量了，比如我需要留下重复记录的最大主键值，比如上面这个，留下最大的 id 为 3000001 这条记录。这样的去重一条 sql 就搞定了，

```
mysql-(ytt/3305)->delete a from d4 a left join (select max(id) id from d4 group by
    r1, r2) b using(id) where b.id is null;
Query OK, 3000000 rows affected (23.29 sec)
```

去掉了 300W 行重复记录，剩下四分之一的正常数据。

```
mysql-(ytt/3305)->select count(*) from d4;
+----------+
| count(*) |
+----------+
|  1000000 |
+----------+
1 row in set (0.06 sec)
```

来看下效果，保留了最大值，其他的删掉了。

```
mysql-(ytt/3305)->select * from d4 order by r1,r2 limit 5;
+---------+------+------+
| id      | r1   | r2   |
+---------+------+------+
| 3000001 |    1 |    1 |
| 3000002 |    2 |    2 |
| 3000003 |    3 |    3 |
| 3000004 |    4 |    4 |
| 3000005 |    5 |    5 |
+---------+------+------+
5 rows in set (0.25 sec)
```