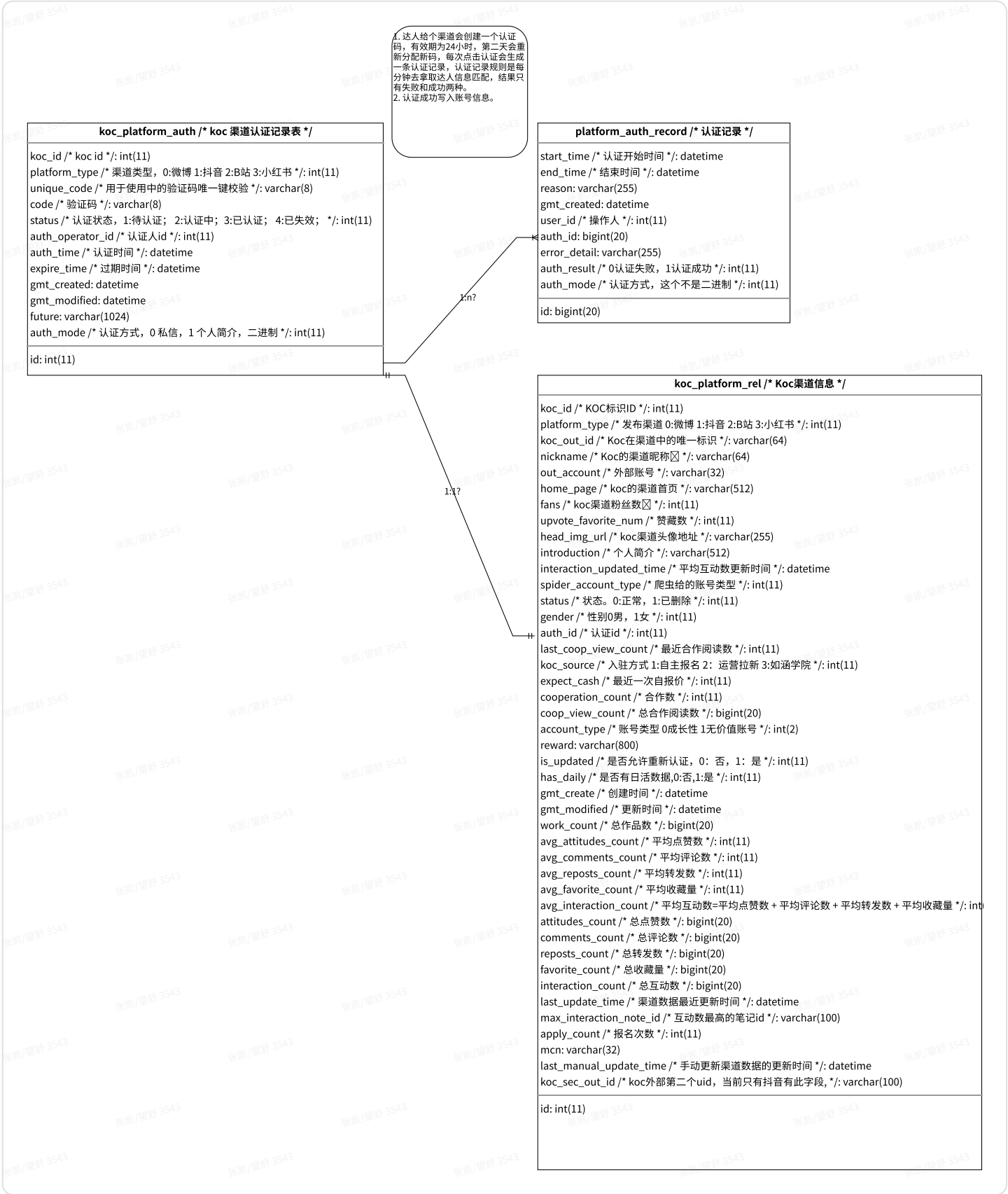
渠道认证老数据迁移方案以及兼容性处理

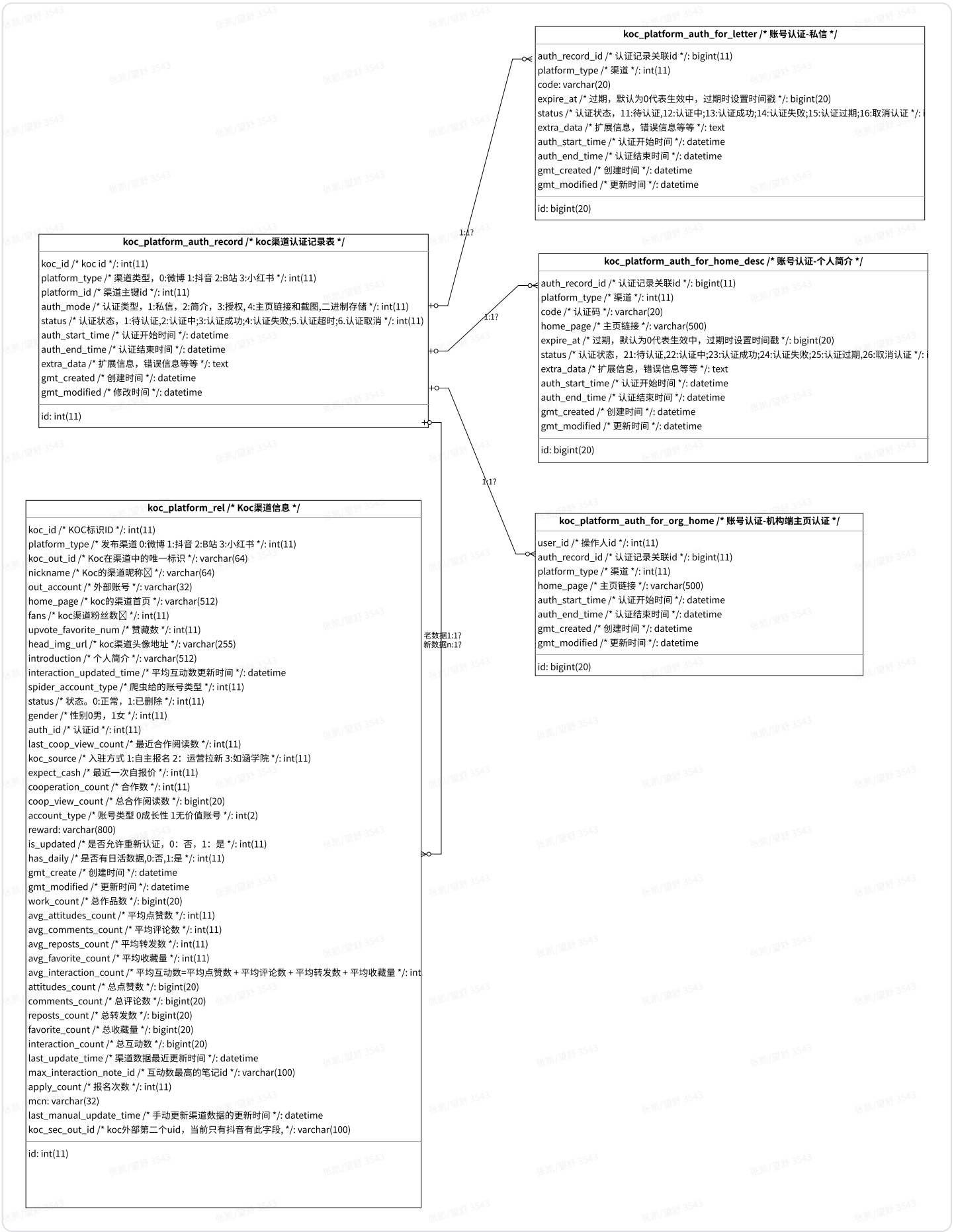
~~老数据迁移方案以及兼容性处理~~ ~~张凯/望舒~~ ~~4月25日 18:00~~

表结构分析

老结构


1. 达人给个渠道会创建一个认证码，有效期为24小时，第二天会重新分配新码，每次点击认证会生成一条认证记录，认证记录规则是每分钟去拿取达人信息匹配，结果只有失败和成功两种。
2. 认证成功写入账号信息。
koc_platform_auth /* koc 渠道认证记录表 */
koc_id /* koc id */: int(11)
platform_type /* 渠道类型，0:微博 1:抖音 2:B站 3:小红书 */: int(11)
unique_code /* 用于使用中的验证码唯一键校验 */: varchar(8)
code /* 验证码 */: varchar(8)
status /* 认证状态，1:待认证； 2:认证中；3:已认证； 4:已失效； */: int(11)
auth_operator_id /* 认证人id */: int(11)
auth_time /* 认证时间 */: datetime
expire_time /* 过期时间 */: datetime
gmt_created: datetime
gmt_modified: datetime
future: varchar(1024)
auth_mode /* 认证方式，0 私信，1 个人简介，二进制 */: int(11)
id: int(11)
1:n?
platform_auth_record /* 认证记录 */
start_time /* 认证开始时间 */: datetime
end_time /* 结束时间 */: datetime
reason: varchar(255)
gmt_created: datetime
user_id /* 操作人 */: int(11)
auth_id: bigint(20)
error_detail: varchar(255)
auth_result /* 0认证失败，1认证成功 */: int(11)
auth_mode /* 认证方式，这个不是二进制 */: int(11)
id: bigint(20)
1:1?
koc_platform_rel /* Koc渠道信息 */
koc_id /* KOC标识ID */: int(11)
platform_type /* 发布渠道 0:微博 1:抖音 2:B站 3:小红书 */: int(11)
koc_out_id /* Koc在渠道中的唯一标识 */: varchar(64)
nickname /* Koc的渠道昵称 */: varchar(64)
out_account /* 外部账号 */: varchar(32)
home_page /* koc的渠道首页 */: varchar(512)
fans /* koc渠道粉丝数 */: int(11)
upvote_favorite_num /* 赞藏数 */: int(11)
head_img_url /* koc渠道头像地址 */: varchar(255)
introduction /* 个人简介 */: varchar(512)
interaction_updated_time /* 平均互动数更新时间 */: datetime
spider_account_type /* 爬虫给的账号类型 */: int(11)
status /* 状态。0:正常，1:已删除 */: int(11)
gender /* 性别0男，1女 */: int(11)
auth_id /* 认证id */: int(11)
last_coop_view_count /* 最近合作阅读数 */: int(11)
koc_source /* 入驻方式 1:自主报名 2：运营拉新 3:如涵学院 */: int(11)
expect_cash /* 最近一次自报价 */: int(11)
cooperation_count /* 合作数 */: int(11)
coop_view_count /* 总合作阅读数 */: bigint(20)
account_type /* 账号类型 0成长性 1无价值账号 */: int(2)
reward: varchar(800)
is_updated /* 是否允许重新认证，0：否，1：是 */: int(11)
has_daily /* 是否有日活数据,0:否,1:是 */: int(11)
gmt_create /* 创建时间 */: datetime
gmt_modified /* 更新时间 */: datetime
work_count /* 总作品数 */: bigint(20)
avg_attitudes_count /* 平均点赞数 */: int(11)
avg_comments_count /* 平均评论数 */: int(11)
avg_reposts_count /* 平均转发数 */: int(11)
avg_favorite_count /* 平均收藏量 */: int(11)
avg_interaction_count /* 平均互动数=平均点赞数 + 平均评论数 + 平均转发数 + 平均收藏量 */: int
attitudes_count /* 总点赞数 */: bigint(20)
comments_count /* 总评论数 */: bigint(20)
reposts_count /* 总转发数 */: bigint(20)
favorite_count /* 总收藏量 */: bigint(20)
interaction_count /* 总互动数 */: bigint(20)
last_update_time /* 渠道数据最近更新时间 */: datetime
max_interaction_note_id /* 互动数最高的笔记id */: varchar(100)
apply_count /* 报名次数 */: int(11)
mcn: varchar(32)
last_manual_update_time /* 手动更新渠道数据的更新时间 */: datetime
koc_sec_out_id /* koc外部第二个uid，当前只有抖音有此字段, */: varchar(100)
id: int(11)

新结构



本次数据迁移影响的功能

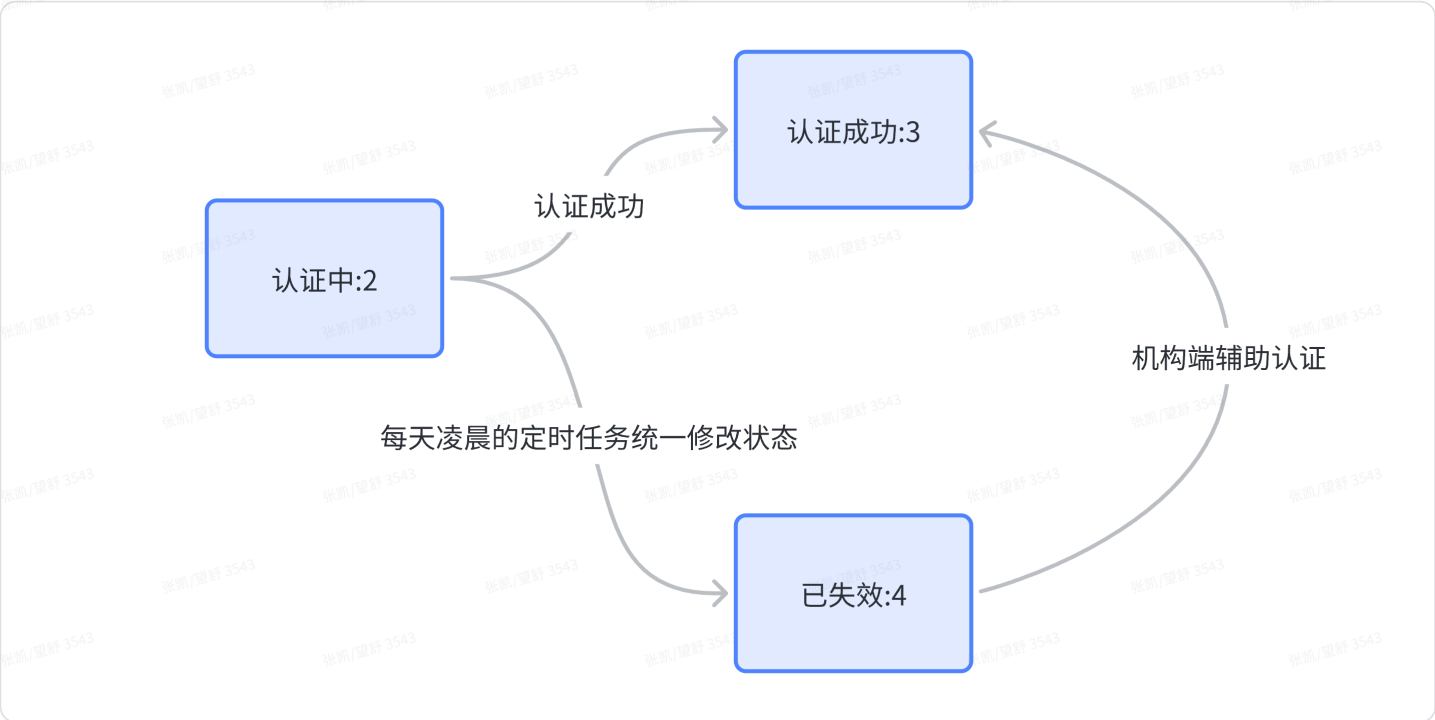
1. C端无影响，新功能上线后，原有的接口和数据依旧可以可使用，认证后的状态会写入到新表。
2. 机构端，

a. 列表页，
   老：原来每个认证会有多条认证记录，直到认证成功
   新：每个认证方式只会有一条最终记录，直到 成功，新加入一种状态 认证中，
b. 机构端协助认证，逻辑差不多，只是表结构变化，影响较小。添加 认证中或认证失败可以协助认证

老数据分析

老数据主表有三种状态，认证中、认证成功、认证失效

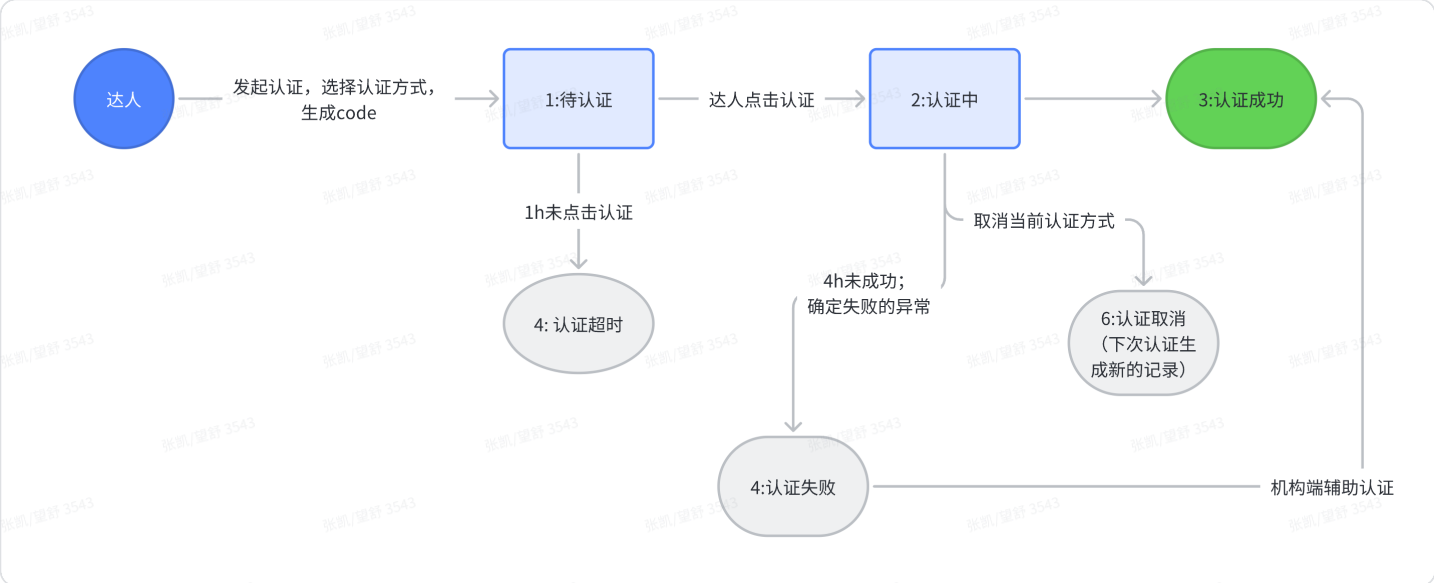
1. 达人给个渠道会创建一个认证码，有效期为24小时，第二天会重新分配新码，每次点击认证会生成一条认证记录，认证记录规则是每分钟内去拿取达人信息匹配，结果只有失败和成功两种。
2. 认证成功写入账号信息。



新数据结构分析

新的同样一个认证表，将不同的认证方式记录拆封到多个子表里面。

新表主状态流转



迁移方案设计

迁移数据方案：1. 发布后迁移数据的时候只迁移认证结束的（认证成功、认证失效），2. 在代码里，处理认证中这种情况。

1. 认证主表：老表 `drpcenter.koc_platform_auth` 表一一对应新表 `koc.koc_platform_auth_record` ，并且将它们的 `id` 对应起来。新表上线的时候，将id基数提到20W（目前线上 `drpcenter.koc_platform_auth` 数据的自增id在18W）

`koc.koc_platform_auth_record` 字段对应：

	A	B	C	D	E
1	**名称**	**评论**	**数据类型**	**非 null**	**迁移方案**
2	id		int(11)	TRUE	drpcenter.koc_platform_auth.id
3	koc_id	koc id	int(11)	TRUE	drpcenter.koc_platform_auth.koc_id
4	platform_type	渠道类型，0:微博 1:抖音 2:B站 3:小红书	int(11)	TRUE	drpcenter.koc_platform_auth.id.platform_type
5	platform_id	渠道主键id	int(11)	FALSE	select b.id from drpcenter.koc_platform_rel where a.id = b.auth_id order by b.id desc lim 1 -- 因为会有重复数据
6	auth_mode	认证类型，1:私信，2:简介，3:授权, 4:主页链接和截图,二进制存储	int(11)	TRUE	二进制字段，迁移完数据，根据字表重新统计并修改
7	status	认证状态，1:待认证,2:认证中;3:认证成功;4:认证失败;5.认证超时;6.认证取消	int(11)	TRUE	1. 认证成功 2.认证失败：老数据是认证过期，但是有认证记录的，记录为认证失败 3. 认证过期：老数据是认证过期，但是没有认证记录的
8	auth_start_time	认证开始时间	datetime	FALSE	根据字表的数据统计
9	auth_end_time	认证结束时间	datetime	FALSE	根据字表的数据统计
10	extra_data	扩展信息，错误信息等等	text	FALSE	对于第一阶段sql迁移的老数据加上 '{"fromOld": true}'
11	gmt_created	创建时间	datetime	TRUE	drpcenter.koc_platform_auth.gmt_created
12	gmt_modified	修改时间	datetime	TRUE	drpcenter.koc_platform_auth.gmt_modified

所以最终的SQL

```sql
-- 原来是认证成功和认证失败的
insert into koc.koc_platform_auth_record (id, koc_id, platform_type,
                                          platform_id, auth_mode, status,
                                          auth_start_time, auth_end_time, extra_
                                          gmt_created, gmt_modified)
select a.id,
       a.koc_id,
       a.platform_type,
       (select b.id from drpcenter.koc_platform_rel b where a.id = b.auth_id ord
       a.auth_mode,
       if(a.status = 3, 3, 4),
       null,                    -- 认证开始时间
       null,                    -- 认证结束时间
       '{"fromOld": true}', -- 额外信息,
       a.gmt_created,
```

```
        a.gmt_modified
from drpcenter.koc_platform_auth a
where a.status in (3, 4) -- 认证成功，认证失败的
  and exists(select 1 from koc.platform_auth_record record where record.auth_id
-- 有认证记录的才需要迁移，没有认证记录的不需要显示
;

-- 原来是认证超时，但是没有认证记录的，转换为超时
insert into koc.koc_platform_auth_record (id, koc_id, platform_type,
                                          platform_id, auth_mode, status,
                                          auth_start_time, auth_end_time, extra_
                                          gmt_created, gmt_modified)
select a.id,
       a.koc_id,
       a.platform_type,
       (select b.id from drpcenter.koc_platform_rel b where a.id = b.auth_id ord
       a.auth_mode,
       5,                    -- 认证超时
       null,                 -- 认证开始时间
       null,                 -- 认证结束时间
       '{"fromOld": true}', -- 额外信息，
       a.gmt_created,
       a.gmt_modified
from drpcenter.koc_platform_auth a
where a.status = 4 -- 认证超时
  and not exists(select 1 from koc.platform_auth_record record where record.auth
;
```

2. 认证记录：在老功能里面，每种认证方式的认证失败记录是有多条的（但认证成功只会有一条），目前我们子表的设计是每个认证在单个认证方式的字表里面只会有一条数据，一个状态，需要将字段聚合选择；然后子表里面的字段是以前两个表的字段，同时有主表的部分信息，和子表的信息。

   处理认证状态：如果有认证成功，则该认证为认证成功；如果是认证失败，则取最后一条认证记录的原因为认证失败的原因。

   示例：个人简介认证记录表：

名称	评论	数据类型	非 null	迁移方案
id		bigint(20)	TRUE	自增
auth_record_id	认证记录关联id	bigint(11)	TRUE	`koc.platform_auth_record.auth_id`

platform_type	渠道	int(11)	TRUE	auth_id对应的 `koc.koc_platform_auth_record.platform_type`
code	认证码	varchar(20)	TRUE	auth_id对应的主表 `drpcenter.koc_platform_auth.code`
home_page	主页链接	varchar(500)	TRUE	auth_id对应主表 `drpcenter.koc_platform_auth.future` ，然后从json里取出来 homePage字段
expire_at	过期，默认为0代表生效中，过期时设置时间戳	bigint(20)	TRUE	因为这次迁移的是认证结束的，迁移的数据码都应该是过期的，所以直接取 `drpcenter.koc_platform_auth.expire_time`
status	认证状态，21:待认证,22:认证中;23:认证成功;24:认证失败;25:认证过期,26:取消认证	int(11)	TRUE	按照 `koc.platform_auth_record.auth_result` 只有认证成功和认证失败两种状态
extra_data	扩展信息，错误信息等等	text	FALSE	1. 对于第一阶段sql迁移的老数据加上 '{"fromOld": true}' 2. 如果是简介认证，还需要设置标识符号； 3. 如果是认证失败的状态，还需要在json里加上认证失败的原因 errorMessage 字段，为老表的认证记录 `koc.platform_auth_record.reason`
auth_start_time	认证开始时间	datetime	FALSE	聚合 `koc.platform_auth_record.start_time` 最小值
auth_end_time	认证结束时间	datetime	FALSE	聚合 `koc.platform_auth_record.end_time` 最大值
gmt_created	创建时间	datetime	TRUE	聚合 `koc.platform_auth_record.gmt_created` 最小值
gmt_modified	更新时间	datetime	TRUE	聚合 `koc.platform_auth_record.gmt_created` 最大值

所以对于认证失败和认证成功查询字表的方式：

```
-- 私信方式
select re.auth_id,
                            if(max(re.auth_result) = 1, 13, 14) status,
                            min(re.auth_mode)                   auth_mode,
                            min(start_time)                     startt,
                            max(end_time)                       endt,
                            max(re.id)                          id,
                            min(gmt_created)                    gmt_created,
                            max(gmt_created)                    gmt_modified
                    from koc.platform_auth_record re
                    where re.auth_mode = 0
                    group by re.auth_id

```

3. 由于机构端是同步操作，其中机构端只迁移认证成功的状态

```
select a.id,
       min(a.platform_type),
       min(re.start_time),
       max(re.end_time),
       re.gmt_created,
       re.gmt_created,
       (select b.home_page
        from drpcenter.koc_platform_rel b
        where a.id = b.auth_id
        order by b.id desc
        limit 1) homePage,
       min(re.user_id)
from koc.koc_platform_auth_record a
         inner join koc.platform_auth_record re on a.id = re.auth_id
where re.auth_mode = 2  -- 人证据记录 mode = 2为机构端认证
  and re.auth_result = 1 -- 只查询认证成功
  and exists(select 1 from drpcenter.koc_platform_rel b where b.auth_id = a.id)
group by a.id
```

4. 迁移完字表后，根据字表数据计算主表的字段 认证开始时间和结束时间，mode

```sql
-- 给主表同步认证开始时间和结束时间，重新计算 mode
update koc.koc_platform_auth_record re
    inner join (select auth_record_id, min(auth_start_time) startt, max(auth_end
                from (select le.auth_record_id, le.auth_start_time, le.auth_end_
                      from koc.koc_platform_auth_for_letter le
                      union all
                      select de.auth_record_id, de.auth_start_time, de.auth_end_
                      from koc.koc_platform_auth_for_home_desc de
                      union all
                      select org.auth_record_id, org.auth_start_time, org.auth_e
                      from koc.koc_platform_auth_for_org_home org) dd
                group by auth_record_id) de on de.auth_record_id = re.id
set re.auth_start_time = de.startt
  , re.auth_end_time   = de.endt
  , re.auth_mode       = mode
where re.auth_start_time is null
```

1. 发布后统一处理老数据

```sql
-- 原来是认证成功和认证失败的
insert into koc.koc_platform_auth_record (id, koc_id, platform_type,
                                          platform_id, auth_mode, status,
                                          auth_start_time, auth_end_time, extra_
                                          gmt_created, gmt_modified)
select a.id,
       a.koc_id,
       a.platform_type,
       (select b.id from drpcenter.koc_platform_rel b where a.id = b.auth_id ord
       a.auth_mode,
       if(a.status = 3, 3, 4),
       null,                -- 认证开始时间
       null,                -- 认证结束时间
       '{"fromOld": true}', -- 额外信息，
       a.gmt_created,
       a.gmt_modified
from drpcenter.koc_platform_auth a
where a.status in (3, 4) -- 认证成功，认证失败的
  and exists(select 1 from koc.platform_auth_record record where record.auth_id
and not exists(select 1 from koc.koc_platform_auth_record main where main.id = a

-- 有认证记录的才需要迁移，没有认证记录的不需要显示
;

```

```
-- 原来是认证超时，但是没有认证记录的，转换为超时
insert into koc.koc_platform_auth_record (id, koc_id, platform_type,
                                          platform_id, auth_mode, status,
                                          auth_start_time, auth_end_time, extra_
                                          gmt_created, gmt_modified)
select a.id,
       a.koc_id,
       a.platform_type,
       (select b.id from drpcenter.koc_platform_rel b where a.id = b.auth_id ord
       a.auth_mode,
       5,                    -- 认证超时
       null,                 -- 认证开始时间
       null,                 -- 认证结束时间
       '{"fromOld": true}', -- 额外信息，
       a.gmt_created,
       a.gmt_modified
from drpcenter.koc_platform_auth a
where a.status = 4 -- 认证超时
  and not exists(select 1 from koc.platform_auth_record record where record.auth
  and not exists(select 1 from koc.koc_platform_auth_record main where main.id =

;
-- 认证成功，并且没有认证记录的，属于2022年改版前的数据，没有认证记录，不显示
insert into koc.koc_platform_auth_record (id, koc_id, platform_type,
                                          platform_id, auth_mode, status,
                                          auth_start_time, auth_end_time, extra_
                                          gmt_created, gmt_modified)
select a.id,
       a.koc_id,
       a.platform_type,
       (select b.id from drpcenter.koc_platform_rel b where a.id = b.auth_id ord
       a.auth_mode,
       3,
       null,                 -- 认证开始时间
       null,                 -- 认证结束时间
       '{"fromOld": true}', -- 额外信息，
       a.gmt_created,
       a.gmt_modified
from drpcenter.koc_platform_auth a
where a.status = 3
  and not exists(select 1 from koc.platform_auth_record record where record.auth
  and not exists(select 1 from koc.koc_platform_auth_record main where main.id =

;
-- 为2022年前认证成功的数据做迁移，统一处理为私信
insert into koc.koc_platform_auth_for_letter (auth_record_id, platform_type, cod
                                              auth_start_time, auth_end_time, gm
```

```
select a.id, a.platform_type, a.code, unix_timestamp(a.expire_time) * 1000, 13,
       (select b.gmt_create from drpcenter.koc_platform_rel b where a.id = b.aut
       from drpcenter.koc_platform_auth a
where a.status = 3
  and not exists(select 1 from koc.platform_auth_record record where record.auth
and not exists(select 1 from koc.koc_platform_auth_for_letter child where child.
;


-- 私信;
insert into koc.koc_platform_auth_for_letter (auth_record_id, platform_type, cod
                                              auth_start_time, auth_end_time, gm
select a.id,
       a.platform_type,
       b.code,
       unix_timestamp(b.expire_time) * 1000,
       c.status,
       if(c.status = 13, '{"fromOld": true}', JSON_SET('{
         "fromOld": true
       }', '$.errorMessage',
                                                  (select co.reason
                                                   from koc.platform_auth_r
                                                   where co.id = c.id))) ex
       c.startt,
       c.endt,
       c.gmt_created,
       c.gmt_modified
from koc.koc_platform_auth_record a
       inner join drpcenter.koc_platform_auth b on a.id = b.id
         inner join (select re.auth_id,
                            if(max(re.auth_result) = 1, 13, 14) status,
                            min(re.auth_mode)                   auth_mode,
                            min(start_time)                     startt,
                            max(end_time)                       endt,
                            max(re.id)                          id,
                            min(gmt_created)                    gmt_created,
                            max(gmt_created)                    gmt_modified
                     from koc.platform_auth_record re
                     where re.auth_mode = 0
                     group by re.auth_id) c on c.auth_id = a.id
where a.auth_mode = 0
  and not exists(select 1 from koc.koc_platform_auth_for_letter child where chil
;
-- 简介;
insert into koc.koc_platform_auth_for_home_desc (auth_record_id, platform_type,
                                                 auth_start_time, auth_end_time,

```

```sql
select a.id,
       a.platform_type,
       b.code,
       unix_timestamp(b.expire_time) * 1000,
       c.status,
       if(c.status = 23, b.future, JSON_SET(b.future, '$.errorMessage',
                                            (select co.reason
                                             from koc.platform_auth_record co
                                             where co.id = c.id)))
       c.startt,
       c.endt,
       c.gmt_created,
       c.gmt_modified,
       SUBSTRING_INDEX(SUBSTRING_INDEX(b.`future`, '{"homePage":"', -1), '"', 1)
from koc.koc_platform_auth_record a
         inner join drpcenter.koc_platform_auth b on a.id = b.id
         inner join (select re.auth_id,
                            if(max(re.auth_result) = 1, 23, 24) status,
                            min(re.auth_mode)                   auth_mode,
                            min(start_time)                     startt,
                            max(end_time)                       endt,
                            max(re.id)                          id,
                            min(gmt_created)                    gmt_created,
                            max(gmt_created)                    gmt_modified
                     from koc.platform_auth_record re
                     where re.auth_mode = 1
                     group by re.auth_id) c on c.auth_id = a.id
where a.auth_mode
  and not exists(select 1 from koc.koc_platform_auth_for_home_desc child where c
;

-- 机构端认证;
-- 机构端一定是认证成功的代表
insert into koc.koc_platform_auth_for_org_home (auth_record_id, platform_type,
                                                auth_start_time, auth_end_time,
                                                user_id)
select a.id,
       min(a.platform_type),
       min(re.start_time),
       max(re.end_time),
       re.gmt_created,
       re.gmt_created,
       (select b.home_page
        from drpcenter.koc_platform_rel b
        where a.id = b.auth_id
        order by b.id desc
        limit 1) homePage,
```

```
        min(re.user_id)
from koc.koc_platform_auth_record a
         inner join koc.platform_auth_record re on a.id = re.auth_id
where re.auth_mode = 2
  and re.auth_result = 1 -- 只查询认证成功
  and exists(select 1 from drpcenter.koc_platform_rel b where b.auth_id = a.id)
  and not exists(select 1 from koc.koc_platform_auth_for_org_home child where ch
group by a.id -- 避免以前有两条脏数据
;

-- 给主表同步认证开始时间和结束时间，重新计算 mode
update koc.koc_platform_auth_record re
    inner join (select auth_record_id, min(auth_start_time) startt, max(auth_end
                from (select le.auth_record_id, le.auth_start_time, le.auth_end_
                      from koc.koc_platform_auth_for_letter le
                      union all
                      select de.auth_record_id, de.auth_start_time, de.auth_end_
                      from koc.koc_platform_auth_for_home_desc de
                      union all
                      select org.auth_record_id, org.auth_start_time, org.auth_e
                      from koc.koc_platform_auth_for_org_home org) dd
                group by auth_record_id) de on de.auth_record_id = re.id
set re.auth_start_time = de.startt
  , re.auth_end_time   = de.endt
  , re.auth_mode       = mode
where re.auth_start_time is null
;
```

2. 线上处于认证中的数据处理

认证中的达人变为认证成功，需要在新表里面加上认证成功的记录，有两处一个是后台处理认证成功，一个是机构端协助认证。

i. 添加认证主表记录

ii. 添加对应认证方式的字表

iii. 支持幂等。

代码实现

`com.ruhnn.koc.jupiter.biz.manager.plat.PlatformAuthMigrateManager#writeNewAuthRecord`

3. 线上认证中到认证过期统一处理

认证中的达人认证码过期，根据是否有认证记录，记录状态为认证失败或认证过期

a. 在定时任务处实现新的认证处理操作

b. 过期处理需要支持幂等。

c. 如果任务出现中断，导致在新表并没有同步记录，最后在整体校验任务中依然可以同步。

`com.ruhnn.koc.jupiter.service.job.PlatformJob#expireAuthRecord`

4. 线上整体校验迁移数据

查询所有处于认证成功和认证失败的数据，根据老表 id 去新表匹配关联关系，如果在新表不存在数据记录，则执行数据补全操作。

`com.ruhnn.koc.jupiter.biz.manager.plat.impl.PlatJobManagerImpl#migrateOldAuthRecord`

5. 迁移数据周期

在前端C端功能上线 1 天后，所有的认证中都会处理为认证过期，直到后续不会有新的业务数据进入老表。

6. 数据校验

a. 由于新数据不会写到老表，根据老表 `drpcenter.koc_platform_auth` 为基础，查询对应校验除认证中的数据量是否一致，新老认证方式的数据认证状态是否一致。

- [x] ~~写sql or 代码日志提醒。~~

```sql
-- 认证成功
select * from drpcenter.koc_platform_auth a
         left join koc.koc_platform_auth_record b on a.id = b.id
         where a.status = 3 and (b.status is null or b.status <> 3);
-- 有认证记录，校验转台是否是认证失败
select * from drpcenter.koc_platform_auth a
                  left join koc.koc_platform_auth_record b on a.id = b.id
where  exists(select 1 from koc.platform_auth_record c where c.auth_id = b.id
    a.status = 4 and (b.status is null or b.status <> 4);
-- 没有认证，校验状态是否是已失效
select * from drpcenter.koc_platform_auth a
                  left join koc.koc_platform_auth_record b on a.id = b.id
where not exists(select 1 from koc.platform_auth_record c where c.auth_id = b
        a.status = 4 and (b.status is null or b.status <> 5);

-- 后续查询是否有认证中的数据，如果有继续执行定时任务将code过期，利用定时任务补充数据，
```

b. 机构端展示的数据规则会有变动，
以前是按照每次认证的记录，可能会有多条记录，排序是按照记录生成时间倒序排列；
现在每个认证方式始终只有一条记录，排序是按照当前记录的开始认证时间倒序，功能上不是很好校验数据是否一致。

c. 认证方式是重新用sql统计的，符合新的需求，不需要重新校验。

7. 注意问题

a. 目前只有活跃的老数据变更状态后会写到新表，新业务里面不会回写到老表。

b.