加密后的敏感字段还能进行模糊查询吗？该如何实现？

JAVA日知录 2023-06-08 08:32 Posted on 安徽

收录于合集
#开发实战
43个 >

前言

有一个问题不知道大家想过没？敏感字段数据是加密存储在数据库的表中，如果需要对这些敏感字段进行模模糊查询，还用原来的通过sql的where从句的like来模糊查询的方式肯定是不行的，那么应该怎么实现呢？这篇文章就来解决这个问题。

场景分析

假如有类似这样的一个场景：有一个人员管理的功能，人员信息列表的主要字段有姓名、性别、用户账号、手机号码、身份证号码、家庭住址、注册日期等，可以对任意一条数据进行增、删、改、查，其中姓名、身份证号码、手机号码字段要支持模糊查询。

简单分析一个场景，可以知道：手机号码、身份证号码、家庭人址字段数据是敏感数据，这些字段的数据是要加密存储在数据库里，在页面上展示的时候需要进行脱敏处理的。

如果用户想要查询真实姓名是包含有“张三”的所有人员信息，可以在页面上输入一个关键字，如“张三”，点击开始查询后，这个参数会传递到后台，后台会执行一条sql，如“`select * from sys_person where real_name like ‘%张三%’`”，执行结果中包含了所有用户真实姓名包含有“张三”的所有数据记录，如“张三”，“张三丰”等。

如果用户要查询手机号码尾号是“0537”的用户，后台执行类似与姓名模糊查询的sql，"`select * from sys_person where phone like '%0537'`"，肯定是得不到正确的结果的，因为手机号码字段在数据库中的数据是加密后的结果，而‘0537’是明文。身份证号码、家庭住址等其他敏感字段在模糊查询的时候也都有类似这样的问题，这也是敏感字段模糊查询的痛点，即模糊查询关键字与实际存储的数据不一致。

实现方案

下面分享几种解决方案：

第一种，先解密再查询

查询出目标表内所有的数据，在内存中对要模糊查询的敏感字段的加密数据进行解密，然后再遍历解密后的数据，与模糊查询关键字进行比较，筛选出包含有模糊查询关键字的数据行。

这种方法是最容易想到的，但有一个比较明显的问题是，模糊查询的过程是在内存中进行的，如果数据量特别大，很容易导致内存溢出，因此不推荐在生产中使用这种方法；

第二种，明文映射表

新建一张映射表，存储敏感字段解密后的数据与目标表主键的映射表，需要模糊查询的时候，先对明文映射表进行模糊查询，得到符合条件的目标数据的主键，再返回来根据主键查询目标表；

这种方法，实际上是有点掩耳盗铃的感觉，敏感字段加密存储的字段主要是考虑到安全性，使用明文映射表来存储解密后的敏感字段，实际上相当于敏感字段没有加密存储，与最被要对敏感字段加密的初衷相违背，因此不推荐在生产中使用这种方法；

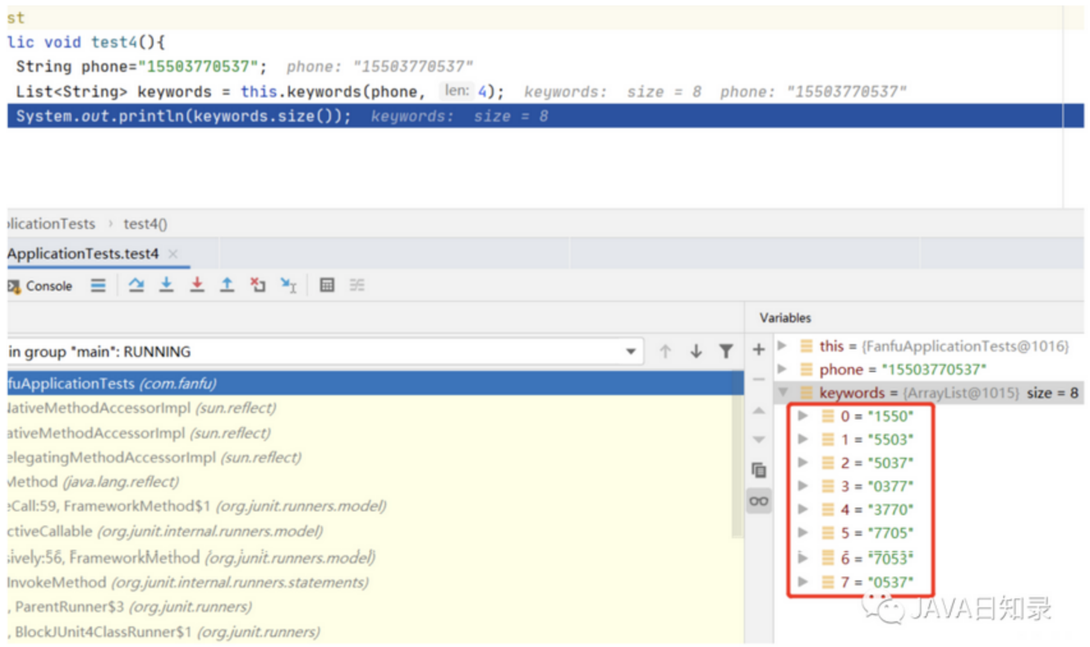
第三种，数据库层面进行解密查询

后台在执行查询sql时对敏感字段先解密，然后再执行like，以上面的人员管理列表模糊查询为例，即对sql的改造为：“`select * from sys_person where AES_DECRYPT(phone,'key') like '%0537'`”；

这种方法的优点是，成本比较小，容易实现，但是缺点很明显，该字段无法通过数据库索引来优化查询，另外有一些数据库无法保证数据库的加解密算法与程序的加解密算法一致，可能会导致可以程序中加密，但是无法在数据库中解密的或者可以在数据库加密无法在程序中解密的问题，因此不推荐在生产中使用这种方法；

第四种，分词密文映射表

这种方法是对第二种思路的基础上进行延伸优化，也是主流的方法。新建一张分词密文映射表，在敏感字段数据新增、修改的后，对敏感字段进行分词组合，如“15503770537”的分词组合有“155”、“0377”、“0537”等，再对每个分词进行加密，建立起敏感字段的分词密文与目标数据行主键的关联关系；在处理模糊查询的时候，对模糊查询关键字进行加密，用加密后的模糊查询关键字，对分词密文映射表进行like查询，得到目标数据行的主键，再以目标数据行的主键为条件返回目标表进行精确查询。



图片一：分组组合加密前

图片二：分组组合加密后

淘宝、阿里、拼多、京东等大厂对用户敏感数据加密后支持模糊查询都是这样的原理，下面是几个大厂的敏感字段模糊查询方案说明，有兴趣可以了解一下：

淘宝密文字段检索方案

- https://open.taobao.com/docV3.htm?docId=106213&docType=1

阿里巴巴文字段检索方案

- https://jaq-doc.alibaba.com/docs/doc.htm?treeId=1&articleId=106213&docType=1

拼多多密文字段检索方案

- https://open.pinduoduo.com/application/document/browse?idStr=3407B605226E77F2

京东密文字段检索方案

- https://jos.jd.com/commondoc?listId=345

这种方法的优点就是原理简单，实现起来也不复杂，但是有一定的局限性，算是一个对性能、业务相折中的一个方案，相比较之下，在能想的方法中，比较推荐这种方法，但是要特别注意的是，对模糊查询的关键字的长度，要在业务层面进行限制；以手机号为例，可以要求对模糊查询的关键字是四位或者是五位，具体可以再根据具体的场景进行详细划分。

为什么要增加这样的限制呢？因为明文加密后长度为变长，有额外的存储成本和查询性能成本，分词组合越多，需要的存储空间以及所消耗的查询性能成本也就更大，并且分词越短，被硬破解的可能性也就越大，也会在一定程度上导致安全性降低；

环境配置

- jdk版本:1.8开发工具：Intellij iDEA 2020.1
- springboot:2.3.9.RELEASE
- mybatis-spring-boot-starter：2.1.4

依赖配置

示例主要用到了SpringAop，加密是对称加密，用到了hutool工具包里的加密解密工具类，也可以使用自己封装的加密解密工具类。

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-aop</artifactId>
</dependency>
<dependency>
    <groupId>cn.hutool</groupId>
    <artifactId>hutool-all</artifactId>
    <version>5.3.3</version>
</dependency>
```

代码实现

1、新建分词密文映射表；

如果是多个模糊查询的字段，可以共用在一张分词密文映射表中扩展多个字段，以示例中的人员管理功能为例，新建`sys_person_phone_encrypt`表（人员的手机号码分词密文映射表），用于存储人员id与分词组合密文的映射关系

```
create table if not exists sys_person_phone_encrypt
(
   id bigint auto_increment comment '主键' primary key,
   person_id int not null comment '关联人员信息表主键',
   phone_key varchar(500) not null comment '手机号码分词密文'
)
comment '人员的手机号码分词密文映射表';
```

2、敏感字段数据在保存入库的时候，对敏感字段进行分词组合并加密码，存储在分词密文映射表；

在注册人员信息的时候，先取出通过AOP进行加密过的手机号码进行解密；手机号码解密之后，对手机号码按照连续四位进行分词组合，并对每一个手机号码的分词进行加密，最后把所有的加密后手机号码分词拼接成一个字符串，与人员id一起保存到人员的手机号码分词密文映射表；

```
public Person registe(Person person) {
    this.personDao.insert(person);
    String phone = this.decrypt(person.getPhoneNumber());
    String phoneKeywords = this.phoneKeywords(phone);
    this.personDao.insertPhoneKeyworkds(person.getId(),phoneKeywords);
    return person;
}
private String phoneKeywords(String phone) {
    String keywords = this.keywords(phone, 4);
    System.out.println(keywords.length());
    return keywords;
}

//分词组合加密
private String keywords(String word, int len) {
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < word.length(); i++) {
        int start = i;
        int end = i + len;
        String sub1 = word.substring(start, end);
        sb.append(this.encrypt(sub1));
        if (end == word.length()) {
            break;
        }
    }
    return sb.toString();
}
public String encrypt(String val) {
    //这里特别注意一下，对称加密是根据密钥进行加密和解密的，加密和解密的密钥是相同的，一旦泄漏，就无秘密可言，
    //“fanfu-csdn”就是我自定义的密钥，这里仅作演示使用，实际业务中，这个密钥要以安全的方式存储；
    byte[] key = SecureUtil.generateKey(SymmetricAlgorithm.DES.getValue(), "fanfu-csdn".getBytes()).get
    SymmetricCrypto aes = new SymmetricCrypto(SymmetricAlgorithm.DES, key);
    String encryptValue = aes.encryptBase64(val);
    return encryptValue;
}
public String decrypt(String val) {
    //这里特别注意一下，对称加密是根据密钥进行加密和解密的，加密和解密的密钥是相同的，一旦泄漏，就无秘密可言，
    //“fanfu-csdn”就是我自定义的密钥，这里仅作演示使用，实际业务中，这个密钥要以安全的方式存储；
    byte[] key = SecureUtil.generateKey(SymmetricAlgorithm.DES.getValue(), "fanfu-csdn".getBytes()).get
    SymmetricCrypto aes = new SymmetricCrypto(SymmetricAlgorithm.DES, key);
    String encryptValue = aes.decryptStr(val);
    return encryptValue;
}
```

3、模糊查询的时候，对模糊查询关键字进行加密，以加密后的关键字密文为查询条件，查询密文映射表，得到目标数据行的id，再以目标数据行id为查询条件，查询目标数据表；

根据手机号码的四位进行模糊查询的时候，以加密后模糊查询的关键字为条件，查询`sys_person_phone_encrypt`表（人员的手机号码分词密文映射表），得到人员信息id；再以人员信息id，查询人员信息表；

```
public List<Person> getPersonList(String phoneVal) {
    if (phoneVal != null) {
       return this.personDao.queryByPhoneEncrypt(this.encrypt(phoneVal));
    }
    return this.personDao.queryList(phoneVal);
}
<select id="queryByPhoneEncrypt" resultMap="personMap">
    select * from sys_person where id in
    (select person_id from sys_person_phone_encrypt
     where phone_key like concat('%',#{phoneVal},'%'))
</select>
```

rest-api#4

```
    "userName": "李先生",
    "loginNo": "10666",
    "phoneNumber": "15566662666",
    "sex": "男",
    "address": "河南省郑州市",
    "houseNumber": "3666",
    "bornYear": null,
    "bornMonth": null,
    "bornDay": null,
    "idcard": "411329200711201666"
},
{
    "id": 18113,
    "userName": "高先生",
    "loginNo": "g7668",
    "phoneNumber": "17876266666",
    "sex": "男"
```


示例完整代码：

- https://gitcode.net/fox9916/fanfu-web.git

END

最后说一句（别白嫖，求关注）

我的每一篇文章都是精心输出，如果这篇文章对你有所帮助，或者有所启发的话，帮忙**点赞、在看、转发、收藏**，你的支持就是我坚持下去的最大动力！

另外我的 **知识星球** 开通了，公众号回复关键词 知识星球 获取限量30元优惠券加入，每天不到3毛钱。目前更新了**SpringCloud alibaba开发实战、Kubernetes云原生实战、分库分表实战、设计模式实战、架构实战、一起学DDD 、SpringBoot 老鸟等，还有每周的送书活动等着你....**

收录于合集 #开发实战 43

‹ 上一篇 · 太强了！一个注解解决数据脱敏问题