Klustron 表文件加密特性介绍及使用案例

**注意：**

 如无特别说明，文中的版本号可以使用任何已发布版本的版本号代替。所有已发布版本详见：<http://doc.klustron.com/zh/Release_notes.html>。

**本文目标：**

主要内容是首先介绍数据库表文件加密的特性，介绍如何使用表文件加密的配置步骤，如何在Klustron安装加密插件，然测试表文件加密使用测试案例，在数据库层面对加密表文件和非加密表文件进行测试对比，以及在操作系统层面直接查看加密表数据文件和非加密表数据文件的数据内容进行对比。

**1 Klustron 存储表文件加密介绍**

Klustron 中的存储引擎提供了表级别的加密选项，可以通过加密文件方式来保护表的数据。以下是有关表文件加密的方式和特点：

1.1表文件加密：表文件加密是一种将特定表的数据文件进行加密的方法。它允许你为一个或多个表启用数据文件级别的加密，从而保护存储在磁盘上的数据。

1.2 特点：

a) 透明性： 对于应用程序来说是透明的。应用程序无需进行任何更改，即可与加密的表交互。

b) 表级加密： 加密操作是在表的数据文件级别上执行的，只会影响到选择加密的表，而不是整个数据库。

c) 性能影响： 加密和解密操作会对性能产生影响，因为这需要额外的计算资源。具体的性能影响会受到硬件性能、加密算法和工作负载的影响。

d) 密钥管理：表文件加密使用主密钥来执行加密和解密操作。你可以使用 Klustron 存储引擎内置的密钥管理器来生成和管理主密钥。

1.3 如何使用表文件加密：

实际使用表文件加密的步骤大致如下：

a) 配置 Klustron 使用 InnoDB 存储引擎。

b) 生成或导入主密钥，用于加密和解密操作。这可以通过 Klustron 内置的密钥管理器完成。

c) 创建一个加密的表，并将表数据存储在其中。这个操作通常在表创建时自动进行。

1.4 总之，表文件加密是 Klustron 中保护表级别数据安全的一种方式。它提供了一种在存储文件级别上对特定表进行加密的方法。

**2 Klustron安装加密插件**

2.1在存储节点上安装加密插件，创建存放加密keyring文件目录‘/kunlun/mysql-keyring/’

[root@kunlun2 ~]# su - kunlun

[kunlun@kunlun2 ~]$ mkdir /kunlun/mysql-keyring/



2.2 修改所有存储节点配置文件，在配置文件中增加以下内容：

[kunlun@kunlun2 ~]$ vi /kunlun/storage\_datadir/57003/data/57003.cnf

early-plugin-load=keyring\_file.so

keyring\_file\_data=/kunlun/mysql-keyring/keyring



2.3 需要修改所有存储节点配置文件，例如这里还修改了

[kunlun@kunlun2 ~]$ vi /kunlun/storage\_datadir/57005/data/57005.cnf

[kunlun@kunlun2 ~]$ vi /kunlun/storage\_datadir/57007/data/57007.cnf

2.4 修改还配置文件后重启存储节点

[kunlun@kunlun2 ~]$ cd /kunlun/instance\_binaries/storage/57003/kunlun-storage-1.2.1/dba\_tools

[kunlun@kunlun2 ~]$ ./stopmysql.sh 57003

[kunlun@kunlun2 ~]$ ./startmysql.sh 57003

[kunlun@kunlun2 ~]$ cd /kunlun/instance\_binaries/storage/57005/kunlun-storage-1.2.1/dba\_tools

[kunlun@kunlun2 ~]$ ./stopmysql.sh 57005

[kunlun@kunlun2 ~]$ ./startmysql.sh 57005

[kunlun@kunlun2 ~]$ cd /kunlun/instance\_binaries/storage/57007/kunlun-storage-1.2.1/dba\_tools

[kunlun@kunlun2 ~]$ ./stopmysql.sh 57007

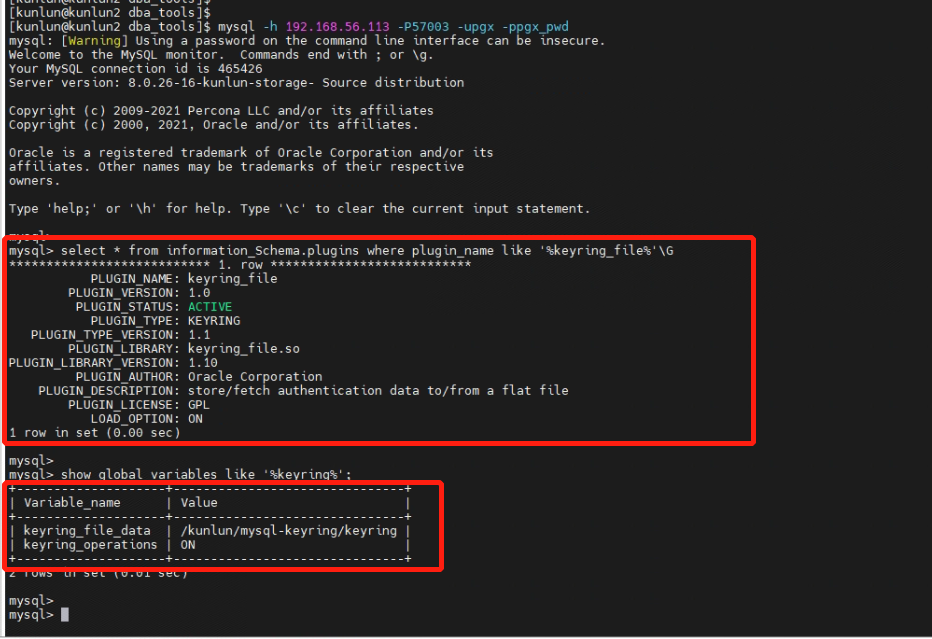
[kunlun@kunlun2 ~]$ ./startmysql.sh 57007

2.5 重启存储节点后验证成功安装加密插件

[kunlun@kunlun2 ~]$ mysql -h 192.168.56.113 -P57003 -upgx -ppgx\_pwd

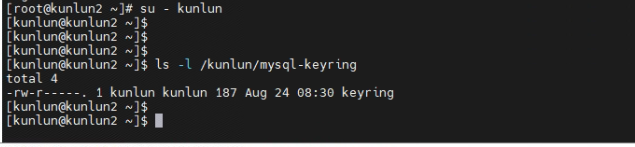
mysql> select \* from information\_Schema.plugins where plugin\_name like '%keyring\_file%'\G

mysql> show global variables like '%keyring%';



2.6 查看keyring文件已经生成

[kunlun@kunlun2 ~]$ ls -l /kunlun/mysql-keyring



完成安装加密插件。

**3 表文件加密使用案例**

这个是基于表文件加密测试案例，创建一张业务表product表，这张表没有做加密，创建一张业务表product\_encryp表，这张表做了加密；然后对没有做加密的product表和做加密的product\_encryp表分别插入相当的数据，查看这两张表的加密属性，并对这两种类型的表分别查看数据进行对比。

3.1连接到数据库，创建数据库用户、创建一个测试数据库，并给用户授权。

[root@kunlun1 ~]# su - kunlun

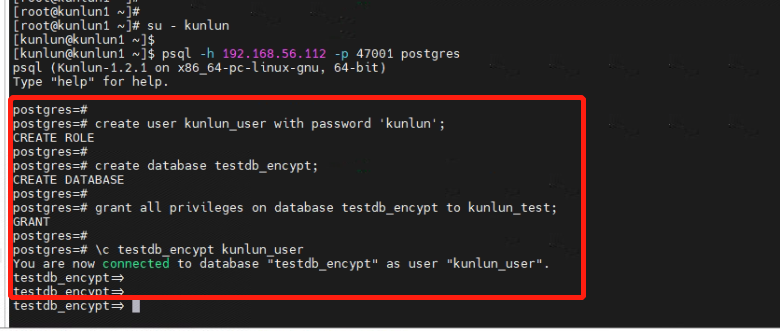
[kunlun@kunlun1 ~]$ psql -h 192.168.56.112 -p 47001 postgres

create user kunlun\_user with password 'kunlun';

create database testdb\_encypt;

grant all privileges on database testdb\_encypt to kunlun\_user;

\c testdb\_encypt kunlun\_user



3.2 创建非加密表product表和加密表product\_encryp表，并给表插入数据。

创建非加密表product表并插入数据：

CREATE TABLE product(

pro\_id int8,

pro\_name varchar(100),

pro\_type varchar(100),

price int8

);

insert into product (pro\_id,pro\_name,pro\_type,price) values (10001,'ipad','padnote',4500);

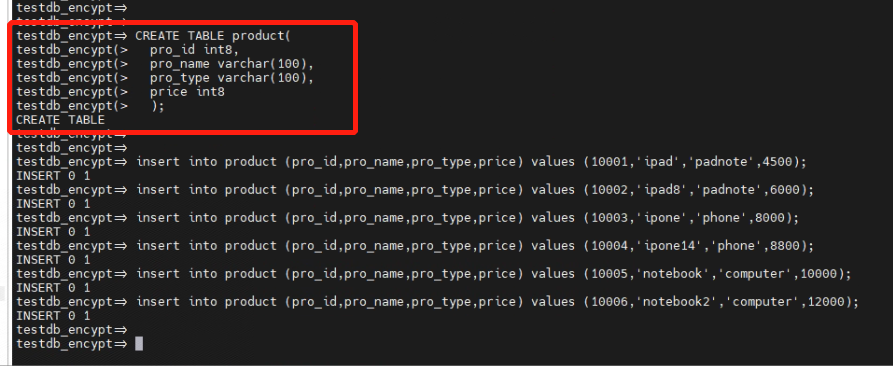
insert into product (pro\_id,pro\_name,pro\_type,price) values (10002,'ipad8','padnote',6000);

insert into product (pro\_id,pro\_name,pro\_type,price) values (10003,'ipone','phone',8000);

insert into product (pro\_id,pro\_name,pro\_type,price) values (10004,'ipone14','phone',8800);

insert into product (pro\_id,pro\_name,pro\_type,price) values (10005,'notebook','computer',10000);

insert into product (pro\_id,pro\_name,pro\_type,price) values (10006,'notebook2','computer',12000);



创建加密表product\_encryp表并插入数据：

CREATE TABLE product\_encryp(

pro\_id int8,

pro\_name varchar(100),

pro\_type varchar(100),

price int8

) with (

engine=innodb,

compression=lz4,

row\_format= COMPACT,

ENCRYPTION = 'Y');

insert into product\_encryp (pro\_id,pro\_name,pro\_type,price) values (10001,'ipad','padnote',4500);

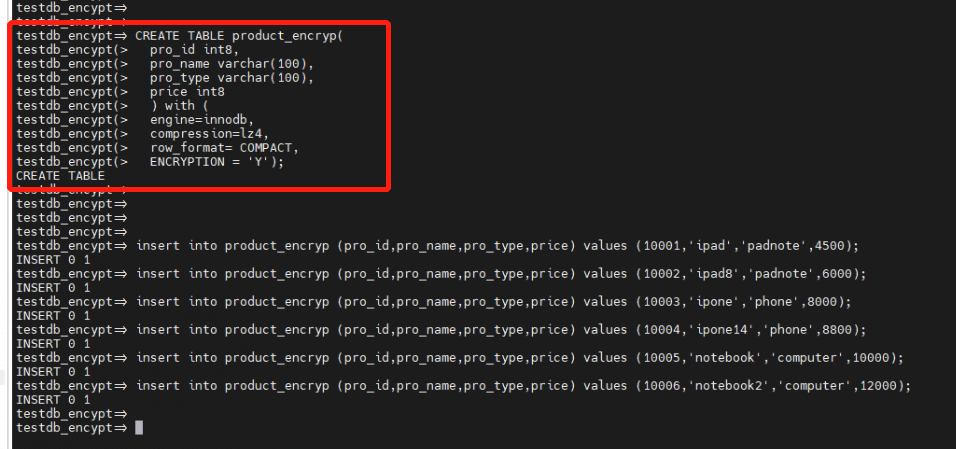
insert into product\_encryp (pro\_id,pro\_name,pro\_type,price) values (10002,'ipad8','padnote',6000);

insert into product\_encryp (pro\_id,pro\_name,pro\_type,price) values (10003,'ipone','phone',8000);

insert into product\_encryp (pro\_id,pro\_name,pro\_type,price) values (10004,'ipone14','phone',8800);

insert into product\_encryp (pro\_id,pro\_name,pro\_type,price) values (10005,'notebook','computer',10000);

insert into product\_encryp (pro\_id,pro\_name,pro\_type,price) values (10006,'notebook2','computer',12000);



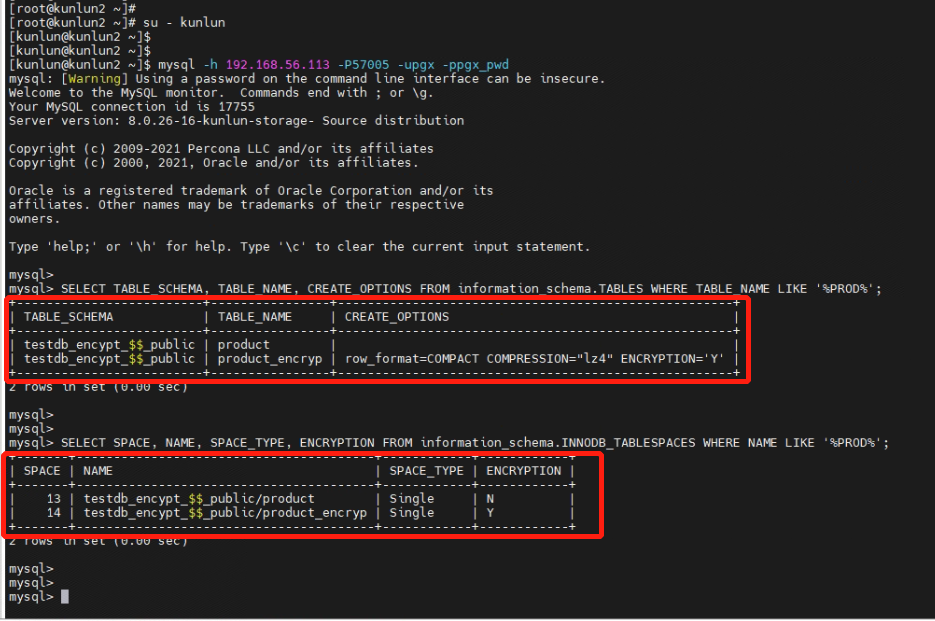
3.3 查看数据库数据字典，对比非加密表product表和加密表product\_encryp表的加密属性，从属性中查看product\_encryp表已加密。

[root@kunlun2 ~]# su – Kunlun

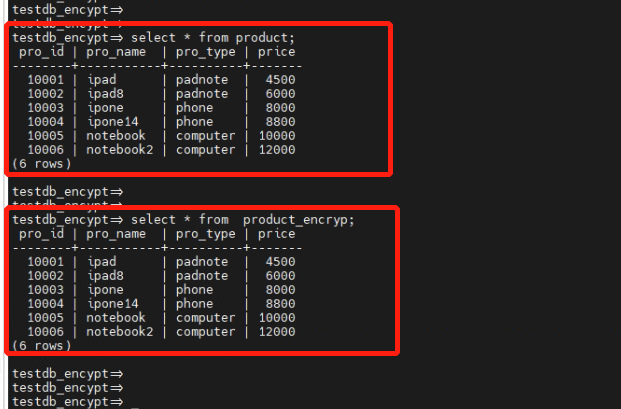
[kunlun@kunlun2 ~]$ mysql -h 192.168.56.113 -P57005 -upgx -ppgx\_pwd

SELECT TABLE\_SCHEMA, TABLE\_NAME, CREATE\_OPTIONS FROM information\_schema.TABLES WHERE TABLE\_NAME LIKE '%PROD%';

SELECT SPACE, NAME, SPACE\_TYPE, ENCRYPTION FROM information\_schema.INNODB\_TABLESPACES WHERE NAME LIKE '%PROD%';



3.4 从客户端直接查看非加密表product表和加密表product\_encryp表的数据，并做数据对比。



从查询的结果可知，加密表对前端的应用查询是透明的，直接使用表加密特性，应用不需要做任何的修改或变更。

**4 在操作系统直接查看加密表文件**

表文件加密后很好的保证了数据的安全性。表文件在操作系统层面上做了加密，从操作系统层面查看文件显示都是密文，或者文件被盗取或被拷贝到其它地方，查看文件的内容都是密文，保证了数据的安全性；这里做从操作系统层面直接查看没有加密的表文件和加密后的表文件的数据内容进行对比。

4.1从操作系统层面直接查看没有加密的表product的系统文件，可以直接查看到文件存放的数据内容的明文信息。

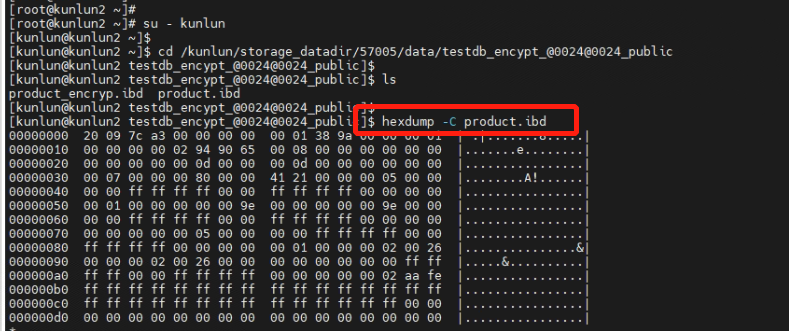
[root@kunlun2 ~]# su - kunlun

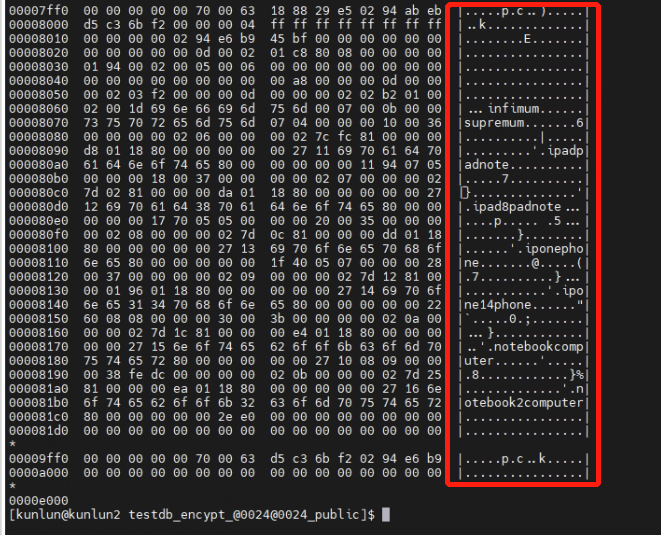
[kunlun@kunlun2 ~]$ cd /kunlun/storage\_datadir/57005/data/testdb\_encypt\_@0024@0024\_public

[kunlun@kunlun2 testdb\_encypt\_@0024@0024\_public]$ ls

product\_encryp.ibd product.ibd

[kunlun@kunlun2 testdb\_encypt\_@0024@0024\_public]$ hexdump -C product.ibd





4.2从操作系统层面直接查看加密的表product\_encryp的系统文件，查看到文件存放的数据已加密，无法获取文件的数据信息。

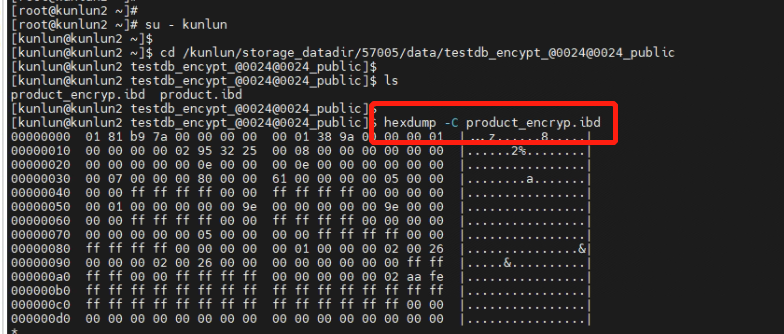
[root@kunlun2 ~]# su - kunlun

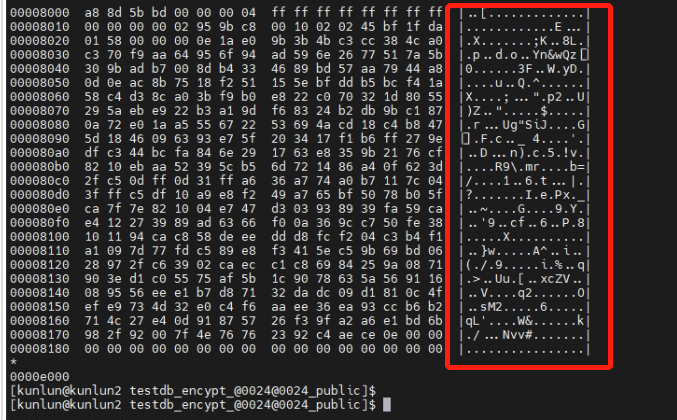
[kunlun@kunlun2 ~]$ cd /kunlun/storage\_datadir/57005/data/testdb\_encypt\_@0024@0024\_public

[kunlun@kunlun2 testdb\_encypt\_@0024@0024\_public]$ ls

product\_encryp.ibd product.ibd

[kunlun@kunlun2 testdb\_encypt\_@0024@0024\_public]$ hexdump -C product\_encryp.ibd





加密后的表product\_encryp表文件查看到的密文，无法查看到文件的明文数据信息。

完成基于数据库表文件加密的测试案例。

【END】