

# 虚谷数据库 V12.8

## 数据加密指南

文档版本 01

发布日期 2025-04-15



版权所有 © 2025 成都虚谷伟业科技有限公司。

## 声明

未经本公司正式书面许可，任何企业和个人不得擅自摘抄、复制、使用本文档中的部分或全部内容，且不得以任何形式进行传播。否则，本公司将保留追究其法律责任的权利。

用户承诺在使用本文档时遵守所有适用的法律法规，并保证不以任何方式从事非法活动。不得利用本文档内容进行任何侵犯他人权益的行为。

## 商标声明



为成都虚谷伟业科技有限公司的注册商标。

本文档提及的其他商标或注册商标均非本公司所有。

## 注意事项

您购买的产品或服务应受本公司商业合同和条款的约束，本文档中描述的部分产品或服务可能不在您的购买或使用范围之内。由于产品版本升级或其他原因，本文档内容将不定期进行更新。

除非合同另有约定，本文档仅作为使用指导，所有内容均不构成任何声明或保证。

## 成都虚谷伟业科技有限公司

地址：四川省成都市锦江区锦盛路 138 号佳霖科创大厦 5 楼 3-14 号

邮编：610023

网址：[www.xugudb.com](http://www.xugudb.com)

# 前言

## 概述



本文档主要介绍了虚谷数据库的数据加密方法与示例，包含表加密、文件加密和传输加密三方面的内容。

## 读者对象

数据库管理员

## 符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 <b>注意</b>	用于传递设备或环境安全警示信息，若不避免，可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。
 <b>说明</b>	对正文中重点信息的补充说明。“说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

## 修改记录

文档版本	发布日期	修改说明
01	2025-04-15	第一次正式发布。

# 目录

1	加密概述 .....	1
2	系统包加密 .....	2
2.1	概述 .....	2
2.2	系统包 DBMS_CRYPTO 加密 .....	2
2.3	系统包 DBMS_CRYPTO 解密 .....	6
3	加密机加密 .....	8
3.1	概述 .....	8
3.2	创建加密机 .....	8
3.3	加密库 .....	9
3.4	加密用户 .....	11
3.5	加密表 .....	14
3.6	加密备份 .....	16
3.7	加密机解密恢复 .....	17
4	传输加密 .....	19
4.1	概述 .....	19
4.2	传输加密 .....	19

# 1 加密概述

数据库加密是通过对数据库中存储的数据进行加密处理，用以保护数据的安全性和隐私性，如用户密码、个人身份信息等，常见的加密算法包括对称加密算法（如 AES、DES）、非对称加密算法（如 RSA）、哈希算法等。加密可在数据存储、传输和处理的各个阶段实现，以确保数据在任何时候都处于受保护的状态。

虚谷数据库通过系统包、加密机和驱动（SSL）三种方式，支持对数据库中表数据、存储文件、传输、备份文件等进行加密，加密使用 SM4 对称加密算法。

- 通过使用内置 DBMS\_CRYPT0 加解密系统包，针对新增表数据加密，可对字段进行加解密。
- 通过使用加密机方式，针对存储文件与备份文件加密，可对库、用户、表下存储数据进行加密。
- 通过驱动（SSL）方式，针对传输加密。

加密方式	加密对象
DBMS_CRYPT0 系统包	新增表数据
加密机	存储文件、备份文件
驱动（SSL）	传输

# 2 系统包加密

## 2.1 概述

使用系统包 DBMS\_CRYPTO 对新增表数据进行加密，需注意以下事项。

- 仅支持在对表插入数据与块语句中使用。
- 对表插入数据时，支持加密数据库的所有数据类型。
- 支持插入 SQL 语句对多字段进行加密。
- 密文默认返回为 RAW 类型。

DBMS_CRYPTO 系统包加密	查询是否密文显示
表中新增数据	√
表中原有数据	×
存储文件中密文存储	×

### 📖 说明

不支持对表中原有数据进行加密，仅支持加密新增数据。

## 2.2 系统包 DBMS\_CRYPTO 加密

### 功能说明

对于表中新增数据加密，首先基于原系统包中包体定义创建自定义加密函数，然后将密文转换为十六进制数字返回。

### 参数解释

系统包加密参数如表2-2所示。

**表 2-2** 系统包加密参数

参数名称	参数类型	参数解释
p_text	RAW/VARCHAR	要加密的资源
typ	INTEGER	加密类型/套件，由块加密算法、模式、填充方式组成
v_text	RAW/VARCHAR	用于加密的密码
v_enc	RAW	加密后返回密文

块加密算法说明如表2-3所示。

**表 2-3** 块加密算法说明

名称	对应十进制数字	十六进制数字
ENCRYPT_DES	1	0x0001
ENCRYPT_3DES_2KEY	2	0x0002
ENCRYPT_3DES	3	0x0003
ENCRYPT_AES	4	0x0004
ENCRYPT_PBE_MD5DES	5	0x0005
ENCRYPT_AES128	6	0x0006
ENCRYPT_AES192	7	0x0007
ENCRYPT_AES256	8	0x0008

块加密模式说明如表2-4所示。

**表 2-4** 块加密模式说明

名称	对应十进制数字	十六进制数字
CHAIN_CBC	256	0x0100
CHAIN_CFB	512	0x0200
CHAIN_ECB	768	0x0300
CHAIN_OFB	1024	0x0400

块加密填充方式说明如表2-5所示。

**表 2-5** 块加密填充方式说明

名称	对应十进制数字	十六进制数字
PAD_PKCS5	4096	0x1000
PAD_NONE	8192	0x2000
PAD_ZERO	12288	0x3000
PAD_XUGU	16384	0x4000

流加密算法说明如表2-6所示。

**表 2-6** 流加密算法说明

名称	对应十进制数字	十六进制数字
ENCRYPT_RC4	129	0x0081

## 示例

- 示例 1

```

--利用内置加密系统包创建自定义加密函数：
SQL> CREATE OR REPLACE FUNCTION encrypt_aes (p_text VARCHAR,
      v_text VARCHAR) RETURN VARCHAR IS
  typ INTEGER := DBMS_CRYPTO.ENCRYPT_AES + DBMS_CRYPTO.
    CHAIN_CBC + DBMS_CRYPTO.PAD_PKCS5;
  v_enc VARCHAR;

```



```

decrypted_raw RAW;
BEGIN
decrypted_raw:=DBMS_CRYPTO.ENCRYPT(p_text,typ,v_text);
v_enc := RAWTOHEX(decrypted_raw);
RETURN v_enc;
END;

-- 示例:
SQL> SELECT encrypt_aes('加密内容','123456') FROM DUAL;

EXPR1 |
-----|
B87F341CD02BD2E4460B61E36F0FB4CF|

```

上述示例，创建加密函数后，将“加密内容”加密为“B87F341CD02BD2E4460B61E36F0FB4CF”。

常用加密套件：

- ENCRYPT\_DES + CHAIN\_CBC + PAD\_PKCS5
- ENCRYPT\_3DES + CHAIN\_CBC + PAD\_PKCS5
- ENCRYPT\_AES + CHAIN\_CBC + PAD\_PKCS5

• 示例 2

```

-- 创建加密测试表:
SQL> CREATE TABLE test_encrypt(id INT,name VARCHAR,
    phonenumber VARCHAR);

-- 使用常规非加密方式进行数据插入:
SQL> INSERT INTO test_encrypt VALUES (1,'张大','13123456789')
    (2,'张二','13987654321');

SQL> SELECT * FROM test_encrypt;

ID | NAME | PHONENUMBER |
-----|
1 | 张大 | 13123456789 |
2 | 张二 | 13987654321 |

-- 使用加密方式进行数据插入:
SQL> INSERT INTO test_encrypt VALUES (3,'张三',encrypt_aes('
    18123456789','123456'));

SQL> SELECT * FROM test_encrypt;

ID | NAME | PHONENUMBER |
-----|
1 | 张大 | 13123456789 |
2 | 张二 | 13987654321 |

```

```
3 | 张三 | A44BC022C913BEF9F5ECFC0788310DD9 |
```

上述示例展示了将新插入数据进行加密，对新插入“张三”的“PHONENUMBER”进行查询以密文方式进行返回。

## 2.3 系统包 DBMS\_CRYPTO 解密

### 功能说明

对于表中新增数据解密，首先基于原系统包中包体定义创建自定义解密函数，对密文转换的十六进制数字进行解密，加解密套件必须一致，否则无法解密。

### 参数解释

参数名称	参数类型	参数解释
p_text	RAW/VARCHAR	要解密的资源
typ	INTEGER	解密类型/套件，由块解密算法、模式、填充方式组成
v_text	RAW/VARCHAR	用于解密的密码
v_enc	RAW	解密后返回明文

### 示例

```
--利用内置加密系统包创建自定义解密函数：
SQL> CREATE OR REPLACE FUNCTION decrypt_aes(p_text VARCHAR,
      v_text VARCHAR) RETURN VARCHAR IS
  typ INTEGER := DBMS_CRYPTO.ENCRYPT_AES + DBMS_CRYPTO.CHAIN_CBC
    + DBMS_CRYPTO.PAD_PKCS5;
  v_dec VARCHAR;
BEGIN
  v_dec:=DBMS_CRYPTO.DECRYPT(p_text,typ,v_text);
  RETURN v_dec;
END;

-- 示例：
SQL> SELECT decrypt_aes('B87F341CD02BD2E4460B61E36F0FB4CF',
      123456') FROM DUAL;

EXPR1 |
-----
加密内容 |
```

```
SQL> SELECT id,name,decrypt_aes(phonenumber,'123456') FROM
      test_encrypt WHERE id=3;

ID | NAME | EXPR1 |
-----
3 | 张三 | 18123456789|

--用错误的解密密码进行解密

SQL> SELECT decrypt_aes('B87F341CD02BD2E4460B61E36F0FB4CF','
      12345') FROM DUAL;

EXPR1 |
-----
|xxxxxxxxxx|
```

上述示例，创建解密函数后，将章节系统包 DBMS\_CRYPTO 加密的示例 1 中密文“B87F341CD02BD2E4460B61E36F0FB4CF”解密为“加密内容”；将表“TEST\_ENCRYPT”插入“张三”的“PHONENUMBER”进行解密，以明文方式进行返回。

# 3 加密机加密

## 3.1 概述

虚谷数据库使用加密机对数据库存储文件和备份文件进行加密，以确保文件的安全性和保密性。

使用加密机对数据库存储文件和备份文件进行加密，需注意以下事项。

- 支持三种加密级别：库级、用户级、表级。
- 指定了库级，未指定用户级或表级，则使用库级加密机进行加密。
- 既指定了库级，也指定了用户级，则按用户级加密机进行加密；若指定了表级，则按表级加密机进行加密。
- 支持对备份文件进行加密。

DBMS_CRYPTO 系统包加密	查询是否密文显示
表中新增数据	×
表中原有数据	×
存储文件中密文存储	√
备份文件中密文存储	√

### 📖 说明

- 加密机不支持表中数据密文显示，仅用于加密数据库数据文件。
- 加密机最多支持 254 个。
- 加密机只允许创建，不允许修改、删除。

## 3.2 创建加密机

### 功能说明

创建加密机必须由数据库安全员 **SYSSSO** 或者拥有安全员权限的用户进行创建。

## 语法格式

```
CREATE ENCRYPTOR encry_name BY encry_password
```

## 参数解释

参数名称	参数解释
encry_name	加密机名称
encry_password	加密密码

## 示例

```
-- 使用 SYSSSO 用户
SQL>CREATE ENCRYPTOR 'ENCRYPTOR1' BY '123';

SQL>CREATE ENCRYPTOR 'ENCRYPTOR2' BY '456';

SQL>CREATE ENCRYPTOR 'ENCRYPTOR3' BY '789';

-- 查询加密机编号、加密机名与加密密码
SQL> SHOW ENCRYPTORS;

ID | NAME | KEYS |
--
-----
2 | ENCRYPTOR1 | 123 |
3 | ENCRYPTOR2 | 456 |
4 | ENCRYPTOR3 | 789 |
```

## 3.3 加密库

### 功能说明

加密库执行步骤如下：

1. 登录 **SYSSSO** 用户创建加密机。
2. 登录 **SYSDBA** 用户创建库并在库下创建表。
3. 查看存储文件下存储加密情况和未加密存储情况。
4. 对库进行加密。
5. 查看存储文件下存储加密情况和未加密存储情况。

预期结论：

- 加密前能够明文查询到表中的数据。

- 加密后不再能明文查询到表中的数据。

### 语法格式

```
ENCRYPT DATABASE BY encry_name|encry_id [CASCADE]
```

### 参数解释

参数名称	参数解释
encry_name	加密机名称
encry_id	加密机编号
CASCADE	是否加密原有表

### 示例

- 示例 1

```
--使用SYSDBA用户创建测试库ENC1、ENC2
SQL> CREATE DATABASE ENC1;

SQL> CREATE DATABASE ENC2;

--在ENC1与ENC2库中分别创建表插入数据

SQL> USE ENC1

SQL> CREATE TABLE enc1_tab(enc VARCHAR);

SQL> INSERT INTO enc1_tab VALUES('qwerqwer001');

SQL> USE ENC2

SQL> CREATE TABLE enc2_tab(enc VARCHAR);

SQL> INSERT INTO enc2_tab VALUES('asdfasdf001');

SQL> CHECKPOINT;
```

- 上述示例，在未使用加密机加密存储文件情况下，数据“qwerqwer001”与“asdfasdf001”在数据库存储文件中以明文方式进行存储。
- 使用加密机“ENCRYPTOR1”加密库“ENC1”新增对象数据，使用加密机“ENCRYPTOR1”加密库“ENC2”所有数据。

- 示例 2

```
--使用SYSDBA用户
```

```
SQL> USE ENC1

SQL> ENCRYPT DATABASE BY 'ENCRYPTOR1';

SQL> INSERT INTO enc1_tab VALUES ('qwerqwer002');

SQL> CREATE TABLE enc1_tab_1(enc VARCHAR);

SQL> INSERT INTO enc1_tab_1 VALUES ('qwerqwer003');

SQL> USE ENC2

SQL> ENCRYPT DATABASE BY 'ENCRYPTOR1' CASCADE;

SQL> INSERT INTO enc2_tab VALUES ('asdfasdf002');

SQL> CREATE TABLE enc2_tab_1(enc VARCHAR);

SQL> INSERT INTO enc2_tab_1 VALUES ('asdfasdf003');

SQL> CHECKPOINT;

SQL> USE SYSTEM

SQL> SELECT db_name, encry_id FROM SYS_DATABASES WHERE db_name
       IN ('ENC1', 'ENC2');

DB_NAME | ENCRY_ID |
--
-----
ENC1 | 2 |
ENC2 | 2 |
```

- 上述示例，在数据库存储文件中分别查询“ENC1”和“ENC2”库中插入数据。
- 在“ENC1”库中，表“ENC1\_TAB”中数据“qwerqwer001”、“qwerqwer002”仍然以明文方式存储于数据库存储文件；新建表“ENC1\_TAB\_1”中数据使用密文存储，在数据库存储文件中已无法查到“qwerqwer003”数据。
- 在“ENC2”库中，表“ENC2\_TAB”与表“ENC2\_TAB\_1”的数据均以密文方式存储于数据库存储文件，在数据库存储文件已无法查到插入的数据。

## 3.4 加密用户

### 功能说明

加密用户执行步骤如下：

1. 登录 **SYSSSO** 用户创建加密机。

2. 登录 **SYSDBA** 用户创建表，并插入数据。
3. 查看存储文件下存储加密情况和未加密存储情况。
4. 对用户进行加密。
5. 查看存储文件下存储加密情况和未加密存储情况。

预期结论：

- 加密前能够明文查询到表中的数据。
- 加密后不再能明文查询到表中的数据。

### 语法格式

```
ENCRYPT USER user_name BY encry_name|encry_id [CASCADE]
```

### 参数解释

参数名称	参数解释
user_name	加密的用户
encry_name	加密机名称
encry_id	加密机编号
CASCADE	是否加密原有表

### 示例

- 示例 1

```
--使用SYSDBA用户创建测试库TEST,在库中创建用户U1与U2
SQL> CREATE DATABASE TEST;

SQL> USE TEST

SQL> CREATE USER U1 IDENTIFIED BY '1234@ABCD';

SQL> CREATE USER U2 IDENTIFIED BY '1234@ABCD';

SQL> GRANT DBA TO U1;

SQL> GRANT DBA TO U2;

--在U1与U2中分别创建表并插入数据

SQL> SET SESSION AUTHORIZATION U1

SQL> CREATE TABLE u1_tab(enc VARCHAR);
```



```
SQL> INSERT INTO u1_tab VALUES ('abcabcabc001');
SQL> SET SESSION AUTHORIZATION U2
SQL> CREATE TABLE u2_tab(enc VARCHAR);
SQL> INSERT INTO u2_tab VALUES ('xyzxyzxyz001');
SQL> SET SESSION AUTHORIZATION SYSDBA
SQL> CHECKPOINT;
```

- 上述示例，在未使用加密机加密存储文件情况下，数据“abcabcabc001”与“xyzxyzxyz001”在数据库存储文件中以明文方式进行存储。
- 使用加密机“ENCRYPTOR2”加密用户“U1”新增对象数据，使用加密机“ENCRYPTOR2”加密用户“U2”所有数据。

#### • 示例 2

```
-- 使用 SYSDBA 用户
SQL> ENCRYPT USER u1 BY 'ENCRYPTOR2';
SQL> SET SESSION AUTHORIZATION U1
SQL> INSERT INTO u1_tab VALUES ('abcabcabc002');
SQL> CREATE TABLE u1_tab_1(enc VARCHAR);
SQL> INSERT INTO u1_tab_1 VALUES ('abcabcabc003');
SQL> SET SESSION AUTHORIZATION SYSDBA
SQL> ENCRYPT USER U2 BY 'ENCRYPTOR2' CASCADE;
SQL> SET SESSION AUTHORIZATION U2
SQL> INSERT INTO u2_tab VALUES ('xyzxyzxyz002');
SQL> CREATE TABLE u2_tab_1(enc VARCHAR);
SQL> INSERT INTO u2_tab_1 VALUES ('xyzxyzxyz003');
SQL> SET SESSION AUTHORIZATION SYSDBA
SQL> CHECKPOINT;
SQL> USE SYSTEM
SQL> SELECT user_name, encry_id FROM SYS_USERS WHERE user_name
IN ('U1', 'U2');
```

USER_NAME	ENCRY_ID
U1	3
U2	3

- 上述示例，在数据库存储文件中分别查询“U1”用户和“U2”用户插入数据。
- 在“U1”用户中，表“U1\_TAB”中数据“abcabcabc001”、“abcabcabc002”仍然以明文方式存储于数据库存储文件，新建表“U1\_TAB\_1”中数据使用密文存储，在数据库存储文件已无法查到“abcabcabc003”数据。
- 在“U2”用户中，表“U2\_TAB”与表“U2\_TAB\_1”数据均以密文方式存储于数据库存储文件，在数据库存储文件已无法查到插入的数据。

## 3.5 加密表

### 功能说明

加密表执行步骤如下：

1. 登录 **SYSSSO** 用户创建加密机。
2. 登录 **SYSDBA** 用户创建表，并插入数据。
3. 查看存储文件下存储加密情况和未加密存储情况。
4. 对表进行加密。
5. 查看存储文件下存储加密情况和未加密存储情况。

预期结论：

- 步骤 3 中能够明文查询到表中的数据。
- 步骤 5 加密后不再能明文查询到表中的数据。

### 语法格式

```
ENCRYPT TABLE table_name BY encry_name|encry_id
```

### 参数解释

参数名称	参数解释
table_name	加密的表
encry_name	加密机名称
encry_id	加密机编号

## 示例

### • 示例 1

```

--使用SYSDBA用户创建测试库TEST_USER,在库中创建表TAB1
SQL> CREATE DATABASE test_user;

SQL> USE test_user

SQL> CREATE TABLE TAB1(enc VARCHAR);

SQL> INSERT INTO TAB1 VALUES('wasdwasdwasd001');

SQL> CHECKPOINT;

```

- 上述示例，在未使用加密机加密存储文件情况下，数据“wasdwasdwasd001”在数据库存储文件中以明文方式进行存储。
- 使用加密机“ENCRYPTOR3”加密表“TAB1”中数据。

### • 示例 2

```

--使用SYSDBA用户
SQL> ENCRYPT TABLE TAB1 BY 'ENCRYPTOR3';

SQL> USE SYSTEM

SQL> SELECT table_name,encry_id FROM SYS_TABLES WHERE
    table_name='TAB1';

TABLE_NAME | ENCRY_ID |
--
-----
TAB1 | 4 |

```

上述示例，使用加密机加密存储文件情况下，表中内容在数据库存储文件中以密文方式进行存储，已无法查询到数据“wasdwasdwasd001”。

## 3.6 加密备份

### 功能说明

进行数据备份时，使用加密机对备份数据进行加密存储，保证数据库备份文件的安全，确保不会因为备份文件泄露导致泄密。

加密备份执行步骤如下：

1. 登录 **SYSSSO** 用户创建加密机。
2. 登录 **SYSDBA** 用户创建测试用户、创建模式、创建表。
3. 使用非加密方式备份处理。
4. 查看存储文件下存储未加密情况。
5. 使用加密机进行备份处理。
6. 查看存储文件下存储加密情况。

预期结论：

- 加密前以明文显示备份表结构。
- 加密后以密文显示备份表结构。

### 语法格式

```
BACKUP USER|SCHEMA|TABLE opt_name TO path_name ENCRYPTOR IS  
    encry_name;  
BACKUP DATABASE path_name ENCRYPTOR IS encry_name;
```

### 参数解释

参数名称	参数解释
opt_name	备份对象名称
path_name	备份路径与备份文件名
encry_name	加密机名称

### 示例

```
--使用SYSDBA用户创建测试用户bak_u1、bak_sch1、bak_tab1、bak_db  
SQL> CREATE USER bak_u1 IDENTIFIED BY '1234@ABCD';  
  
SQL> CREATE SCHEMA bak_sch1;  
  
SQL> CREATE TABLE bak_tab1(ID INT);
```

```
SQL> CREATE DATABASE bak_db;

--使用非加密方式进行备份表
SQL> BACKUP TABLE bak_tab1 TO '/BACKUP/BAK_TABLE1.EXP';

--使用加密机ENCRYPTOR3进行加密备份用户下对象
SQL> BACKUP USER bak_u1 TO '/BACKUP/BAK_USER.EXP' ENCRYPTOR IS
    'ENCRYPTOR3';

--使用加密机ENCRYPTOR3进行加密备份模式下对象
SQL> BACKUP SCHEMA bak_sch1 TO '/BACKUP/BAK_SCHEMA.EXP'
    ENCRYPTOR IS 'ENCRYPTOR3';

--使用加密机ENCRYPTOR3进行加密备份表
SQL> BACKUP TABLE bak_tab1 TO '/BACKUP/BAK_TABLE.EXP' ENCRYPTOR
    IS 'ENCRYPTOR3';

--使用加密机ENCRYPTOR3进行加密备份库下所有对象
SQL> USE bak_db

SQL> BACKUP DATABASE TO '/BACKUP/BAK_DATABASE.EXP' ENCRYPTOR IS
    'ENCRYPTOR3';
```

上述示例，使用用户自定义密钥加密备份表，备份文件中以密文显示备份表结构，恢复时会进行密钥检测。

## 3.7 加密机解密恢复

### 功能说明

在数据库中使用加密机进行加密备份后，使用相同加密机进行解密恢复。

### 语法格式

```
RESTORE USER|SCHEMA|TABLE opt_name FROM path_name ENCRYPTOR IS
    encry_name;
RESTORE DATABASE db_name FROM path_name ENCRYPTOR IS encry_name
    ;
```

### 参数解释

参数名称	参数解释
db_name	要恢复的库名
opt_name	备份对象名称
接下一页	

参数名称	参数解释
path_name	备份路径与备份文件名
encry_name	加密机名称

### 示例

```
--不使用加密机进行解密恢复用户下对象，会报错解密口令错误不允许解密恢复
SQL> RESTORE USER bak_u1 FROM '/BACKUP/BAK_USER.EXP';
Error: [E2039] 解密口令错误

--使用错误加密机进行解密恢复用户下对象，会报错解密口令错误不允许解密恢复
SQL> RESTORE USER bak_u1 FROM '/BACKUP/BAK_USER.EXP' ENCRYPTOR IS 'ENCRYPTOR2';
Error: [E2039] 解密口令错误

--使用加密机ENCRYPTOR3进行解密恢复用户下对象
SQL> RESTORE USER bak_u1 FROM '/BACKUP/BAK_USER.EXP' ENCRYPTOR IS 'ENCRYPTOR3';

--使用加密机ENCRYPTOR3进行解密恢复模式下对象
SQL> RESTORE SCHEMA bak_sch1 FROM '/BACKUP/BAK_SCHEMA.EXP' ENCRYPTOR IS 'ENCRYPTOR3';

--使用加密机ENCRYPTOR3进行解密恢复表
SQL> RESTORE TABLE bak_tab1 FROM '/BACKUP/BAK_TABLE.EXP' ENCRYPTOR IS 'ENCRYPTOR3';

--使用加密机ENCRYPTOR3进行解密恢复库下所有对象
SQL> RESTORE DATABASE SYSTEM FROM '/BACKUP/BAK_DATABASE.EXP' ENCRYPTOR IS 'ENCRYPTOR3';
```

### 说明

加密备份在不使用加密机或者使用错误加密机进行解密恢复时，会报解密口令错误，必须使用正确的加密机才能正常恢复。

# 4 传输加密

## 4.1 概述

使用驱动（SSL）进行数据库连接及命令操作时，对传输数据流进行加密，保证传输数据流安全性，防止因数据流被截获而导致数据泄露。

使用 JDBC 驱动进行传输加密需要将动态库加载至客户端工具所在机器：

- Windows 操作系统机器将 `xgssl.dll` 放置于 `C:\Windows\System32` 下。
- Linux 操作系统机器将 `libxgssl.so` 放置于 `/usr/lib` 或 `/usr/lib64` 下。

## 4.2 传输加密

### 功能说明

- JDBC 加密：在动态库加载完毕后，JDBC 在连接串加上“`SSL=SSL`”以采用传输加密方式。
- ODBC 加密：ODBC 传输加密无需加载动态库，只需要在连接串后加上“`UseSSL=TRUE`”，或者在配置 ODBC 数据源 DSN 时选择启用安全连接。

### 语法格式

- JDBC 加密

```
jdbc:xugu://127.0.0.1:5138/SYSTEM?USER=SYSDBA&PASSWORD=SYSDBA  
&SSL=SSL
```

- ODBC 加密

```
Driver={XuguSQL 11.2};UID=SYSDBA;PWD=SYSDBA;SERVER=127.0.0.1;  
Port=5138;Database=SYSTEM;CharSet=GBK;UseSSL=TRUE
```

### 示例

以 Windows 平台下 DBeaver 工具为例。

将 `xgssl.dll` 动态库放置于 `C:\Windows\System32` 下，在 JDBC 连接串上配置 `SSL` 值为 `ssl`。或者在配置 ODBC 数据源 DSN 时选择启用安全连接。

- 设置数据库连接为非传输加密后，使用 DBeaver 连接数据库执行以下 SQL。

```
CREATE TABLE t_int_1(A int);
```

```
INSERT INTO t_int_1 VALUES (-58);
```

使用 Wireshark 工具进行抓包，登录数据库用户名和密码等相关信息、创建的表结构、插入的数据均为明文显示。

- 断开数据库连接，配置 SSL 加密传输后，使用 DBeaver 重新连接数据库执行以下 SQL。

```
INSERT INTO t_int_1 VALUES (-58);
```

使用 Wireshark 工具进行抓包，登录数据库相关信息与插入的数据均为密文显示。





成都虚谷伟业科技有限公司

联系电话：400-8886236

官方网站：[www.xugudb.com](http://www.xugudb.com)