虚谷数据库 V12.7

管理员指南

文档版本 01

发布日期 2025-01-10

虚谷数据库
XUGU DB

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

虚谷数据库 XUGU DB

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

前言

概述

本文档从多方面介绍了配置、管理数据库的详细信息。

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
⚠ 注意	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。
🕮 说明	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	2025-01-10	第一次正式发布。

目录

1 用户管理

1.1 用户介绍

1.1.1 用户概念

用户是数据库中的账号，可以用来登录和操作数据库。用户登录时需要进行身份鉴别，合法有效的用户才能登录成功；同时每个用户拥有一定的权限，可以在权限允许的范围内对数据库进行操作。

数据库的用户可以分为系统用户和普通用户。系统用户是由数据库本身自动创建的用户，每个系统用户拥有已设定好的权限；非系统用户是指由有权用户手动创建的用户。每个用户都隶属于某个特定的（逻辑）库（以下简称“库”）。例如，每个库中都会有系统用户 SYSDBA、SYSAUDITOR、SYSSSO 和 GUEST；而非系统用户被创建后，仅在当前库中有效。可以在不同库中创建同名但不同密码、不同属性、不同权限的用户。

1.1.2 用户属性

每个用户都拥有标识，即名称和 ID 号。其中名称和 ID 号在当前库中唯一，即同一个库中不能有两个及以上同名或同 ID 号的用户。库的标识和用户标识可以唯一确定一个数据库用户。

每个用户还拥有其他属性，包括用户口令、被授予的角色（角色相关内容参见第三章自主访问控制中的权限和角色相关部分）、账号有效期、口令有效期、锁定标志、信任 IP、临时表空间配额等等。属性记载详情查看 2.5 章节相关数据字典。

说明

- 用户可以创建多个实例库，每个实例库下的用户属于当前库，互不影响。
- 单个实例库下的用户标识唯一，不能存在同名的其他对象，否则创建失败。
- 首次启动数据库时自动生成管理员用户（SYSDBA、SYSAUDITOR、SYSSSO）及 GUEST 用户，应及时修改管理员用户信息。
- 每个实例库在创建时自动生成管理员用户，对实例库进行一对一管理。

1.1.3 用户密码管理

用户密码设置要求：

- 数据库账号口令应为无意义的字符组，长度至少八位，并且至少包括数字、英文字母、特殊字符等其中两类字符。可设置相应的策略强制复杂的口令。
- 必须根据安全要求对数据库管理系统的密码策略进行设置和调整，以确保口令符合要求。

应定期或不定期修改数据库管理员口令，在下述几种情况下应修改数据库管理员口令：

- 数据库正式使用之前
- 数据库系统或相关的应用系统遭到入侵
- 数据库管理员轮换
- 数据库管理员口令泄露
- 其它修改口令要求

⚠ 注意

- 对于数据库中的敏感字段，如：口令等，要加密保存。
- 账号口令使用较为复杂的口令策略，同时，口令的变更应通知相关责任人，否则会出现建立连接失败的情况 (口令连续输入错误超过三次会锁定连接访问 IP 三分钟)。
- 在现有数据库设计中，数据库用户对其归属对象 (即对象属主为它) 拥有所有权限，即使不赋予 DDL 权限，其对归属对象仍具有 DDL 操作权限。为避免账号用户越权操作，建议不开放模式属主用户对外使用。

1.2 用户分类

1.2.1 数据库系统用户分类

数据库系统用户可以分为以下三类，针对每个数据库账号按最小权限原则设置其在相应数据库中的权限。

- 系统管理员：具有系统管理权限，能够管理数据库系统中的所有组件及所有数据库，包括账号管理、服务管理、数据库管理等。

- 数据库管理员：具有数据库管理权限，能够管理相关数据库中的账号、对象及数据，包括创建、删除、修改数据库等。
- 数据库用户：具有数据库访问权限，只能以特定的权限访问特定的数据库对象，包括插入、删除、修改数据库特定表记录等，不具有数据库管理权限。

1.2.2 数据库管理员用户分类

数据库管理员的权限体系为“三权分立”，即每个库中 SYSDBA、SYSAUDITOR、SYSSSO 三个用户分别拥有管理权限、审计权限、安全相关权限。

数据库管理员（SYSDBA）

SYSDBA 拥有被系统赋予的默认权限，以行使如下职责：

- 管理各种数据库对象，如用户、角色、表、视图、存储过程等的创建、修改、删除。
- 为创建的用户授予/回收各种权限（除了审计和安全之外的权限）。
- 更改数据库参数，设置数据库资源限额。
- 管理数据库的备份、恢复。
- 管理数据库集群。

审计管理员（SYSAUDITOR）

SYSAUDITOR 拥有被系统赋予的默认权限，以行使如下职责：

- 开启和关闭审计功能。
- 对指定的主体或客体启用/禁用指定类型的审计。
- 查询和管理审计结果。
- 管理审计相关权限，授予其他用户/从其他用户收回权限。

安全管理员（SYSSSO）

SYSSSO 拥有被系统赋予的默认权限，以行使如下职责：

- 创建加密机。
- 对主体或客体定义/应用/删除安全策略。
- 黑白名单规则制定。
- 管理安全相关权限，授予其他用户/从其他用户收回权限。

1.3 用户操作

在数据库系统中包括两类用户，一类是系统用户，一类是普通用户，系统用户是数据库创建时默认自带的用户，他们拥有系统自定义划分的权限；普通用户由数据库管理员创建，并且根据应用需求划分相应的权限。针对普通用户的管理，应采用最小权限的方式进行权限管理，避免用户越权操作数据库，影响数据库安全。

1.3.1 创建用户

1.3.1.1 主要语法结构

语法格式

```
CreateUserStmt::=
CREATE USER user_name [ LOGIN alias_name ] IDENTIFIED BY 'password'
[ DEFAULT ROLE role_name_1[, role_name_2[, ...] ] ]
[ VALID UNTIL date_time_expr ]
[ ACCOUNT { LOCK | UNLOCK } ]
[ PASSWORD EXPIRE ]
[ opt_trust_ip ]
[ opt_user_quotas ]
[ ENCRYPT BY 'encryptor_name' ]
```

参数说明

- user_name：要创建的用户名。
- alias_name：用户的别名。
- password：用户口令字符串。
- role_name_1, role_name_2, ... ：角色名。
- VALID UNTIL date_time_expr：用户有效期截止时间的字符串，格式为日期或日期时间。
- ACCOUNT LOCK | UNLOCK ：账户是否锁定。
- PASSWORD EXPIRE：密码失效。
- opt_trust_ip：IPV4 地址表达式，外部用单引号包裹。表达式可以为如下形式：
  - 单个 IP 地址：例如’192.168.2.21’。
  - 多个 IP 地址：用逗号分隔的多个地址，例如’192.168.2.21,192.168.2.22,192.168.2.105’。
  - IP 地址范围；用减号分隔的地址上下界，例如’192.168.2.20-192.168.2.29’。

- 任意 IP 地址：关键字'ANY'。

- opt_user_quotas：设置用户在数据库中各种资源上的配额限制。
- encryptor_name：加密机（即加密用的密钥）的名称。加密机相关内容请参见《数据加密指南》。

1.3.1.2 资源配额限制 opt_user_quotas

语法格式

```
opt_user_quotas::=
QUOTA int_value quantity_unit ON MEMORY
|    QUOTA UNLIMITED ON MEMORY
|    QUOTA int_value quantity_unit ON TEMP TABLESPACE
|    QUOTA UNLIMITED ON TEMP TABLESPACE
|    QUOTA int_value quantity_unit ON UNDO TABLESPACE
|    QUOTA UNLIMITED ON UNDO TABLESPACE
|    QUOTA int_value ON CURSOR
|    QUOTA UNLIMITED ON CURSOR
|    QUOTA int_value ON SESSION
|    QUOTA UNLIMITED ON SESSION
|    QUOTA int_value ON IO
|    QUOTA UNLIMITED ON IO
|    QUOTA int_value ON PROCEDURE
|    QUOTA UNLIMITED ON PROCEDURE
|    QUOTA int_value quantity_unit ON TABLESPACE
|    QUOTA UNLIMITED ON TABLESPACE
```

参数说明

- int_value：设置的数值，取值为整数。
- quantity_unit：设置数值的单位。例如要设置的资源配额为临时表空间，则单位可以为 M、G 等。
- UNLIMITED：无限制。
- MEMORY：内存配额。
- TEMP TABLESPACE：临时表空间配额。
- UNDO TABLESPACE：回滚表空间配额。
- CURSOR：游标配额。
- SESSION：会话配额。
- IO：I/O 配额。

- PROCEDURE：存储过程配额。
- TABLESPACE：表空间配额。

1.3.1.3 示例

- 创建一个名为 usr_test 的用户，具有以下属性：
  - 登录别名为 ut。
  - 登录密码为 123QWEasd!@。
  - 默认角色为 role_1 和 role_2。
  - 用户有效期截至 2024-12-31 12:00:00。
  - 账户状态为 ACCOUNT LOCK 锁定状态。
  - 临时表空间配额 QUOTA 20 MB ON TEMP TABLESPACE，为用户设置了 20 MB 的临时表空间配额。

```
SQL> CREATE USER usr_test LOGIN ut IDENTIFIED BY '123QWEasd!@'
DEFAULT ROLE role_1, role_2
VALID UNTIL '2021-12-31 12:00:00'
ACCOUNT LOCK
QUOTA 20 M ON TEMP TABLESPACE;
```

- 使用 ACCOUNT LOCK 创建用户，支持对用户进行创建对象、授权操作，切换与登录报错。

```
-- 创建用户
SQL> CREATE USER user_test IDENTIFIED BY '123QWE$$&' ACCOUNT LOCK
   ;

-- 用户下创表
SQL> CREATE TABLE user_test.t1(id INT);

SQL> INSERT INTO user_test.t1 VALUES(1);

SQL> SELECT * FROM user_test.t1;

ID |
-----------------------------------------------------------
1|

-- 授权与回收权限
SQL> GRANT DBA TO user_test;

SQL> REVOKE DBA FROM user_test;

-- 切换用户失败
```

```
SQL> SET SESSION AUTHORIZATION user_test;
[E18063] 切换用户失败，账户已锁定

-- 登录用户失败
SQL> USE SYSTEM USER = user_test PASSWORD = '123QWE$$&';
[E18019] 登录验证失败

-- 解锁
SQL> ALTER USER user_test ACCOUNT UNLOCK;

-- 切换用户成功
SQL> SET SESSION AUTHORIZATION user_test;

-- 切换回SYSDBA
SQL> SET SESSION AUTHORIZATION SYSDBA;

-- 登录用户成功
SQL> USE SYSTEM USER = user_test PASSWORD = '123QWE$$&';

-- 登录回SYSDBA
SQL> USE SYSTEM USER = SYSDBA PASSWORD = 'SYSDBA';
```

- 使用 PASSWORD EXPIRE 创建用户，支持对用户进行创建对象、授权操作，切换与登录报错。

```
-- 创建用户
SQL> CREATE USER user_test_2 IDENTIFIED BY '123qwe###' PASSWORD
   EXPIRE;

-- 用户下创表
SQL> CREATE TABLE user_test_2.t2(id INT);

SQL> INSERT INTO user_test_2.t2 VALUES(1);

SQL> SELECT * FROM user_test_2.t2;

ID |
-----------------------------------------------------------------
1|

-- 授权与回收权限
SQL> GRANT DBA TO user_test_2;

SQL> REVOKE DBA FROM user_test_2;

-- 切换用户失败
SQL> SET SESSION AUTHORIZATION user_test_2;
[E18064] 切换用户失败，密码已失效

-- 登录用户失败
SQL> USE SYSTEM USER = user_test_2 PASSWORD = '123qwe###';
[E18019] 登录验证失败

-- 修改用户密码
```

```
SQL> ALTER USER user_test_2 IDENTIFIED BY 'pass_1234';

-- 切换用户成功
SQL> SET SESSION AUTHORIZATION user_test_2;

-- 切换回SYSDBA
SQL> SET SESSION AUTHORIZATION SYSDBA;

-- 登录用户成功
SQL> USE SYSTEM USER = user_test_2 PASSWORD = 'pass_1234';

-- 登录回SYSDBA
SQL> USE SYSTEM USER = SYSDBA PASSWORD = 'SYSDBA';
```

- 使用 VALID UNTIL 创建用户，支持对用户进行创建对象、授权操作，切换与登录报错。

```
-- 创建用户
SQL> CREATE USER user_test_3 IDENTIFIED BY '123qwe###' VALID
   UNTIL '2008-08-08';

-- 用户下创表
SQL> CREATE TABLE user_test_3.t3(id INT);

SQL> INSERT INTO user_test_3.t3 VALUES(1);

SQL> SELECT * FROM user_test_3.t3;

ID |
-----------------------------------------------------------
1|

-- 授权与回收权限
SQL> GRANT DBA TO user_test_3;

SQL> REVOKE DBA FROM user_test_3;

-- 切换用户失败
SQL> SET SESSION AUTHORIZATION user_test_3;
[E18062] 切换用户失败，时间已过期

-- 登录用户失败
SQL> USE SYSTEM USER = user_test_3 PASSWORD = '123qwe###';
[E18019] 登录验证失败

-- 修改用户时间
SQL> ALTER USER user_test_3 VALID UNTIL '2099-10-1';

-- 切换用户成功
SQL> SET SESSION AUTHORIZATION user_test_3;

-- 切换回SYSDBA
SQL> SET SESSION AUTHORIZATION SYSDBA;

-- 登录用户成功
SQL> USE SYSTEM USER = user_test_3 PASSWORD = '123qwe###';
```

```
-- 登录回SYSDBA
SQL> USE SYSTEM USER = SYSDBA PASSWORD = 'SYSDBA';
```

1.3.2 设置用户

设置用户请注意以下事项：

- 在系统正式使用前，数据库管理员应对不需要的账号进行删除或锁定。
- 数据库管理员具有最高数据库管理权限，其他人员需要访问数据库或需要具有一定数据库操作权限，必须向信息管理部门和业务部门主管领导申请。审批通过后，由数据库管理员提供账号信息，其他人员通过指定账号访问数据库。申请账号名称均统一使用大写，账号应与实际使用责任人相匹配，并做好记录，以便后期跟踪、处理。
- 数据库管理员为每一个数据库用户根据需要的权限建立专门的账号，以区分责任，提高系统的安全性，用户必须使用自己的账号登录数据库。
- 对账号权限的设置遵从最小化原则。
- 普通数据库用户账号与数据库管理员账号分离。
- 对数据文件访问权限进行控制，如：禁止除专用账号外的其它账号访问、修改、删除数据文件。
- 删除不需要的实例数据库，在允许存在的实例数据库中严格控制数据库账号的权限。

1.3.3 修改用户信息

1.3.3.1 主要语法结构

语法格式

```
AlterUserStmt::=
ALTER USER user_name
[ LOGIN alias_name ]
[ IDENTIFIED BY 'password' ]
[ VALID UNTIL date_time_expr ]
[ ACCOUNT { LOCK | UNLOCK } ]
[ PASSWORD EXPIRE ]
[ opt_trust_ip ]
[ opt_user_quotas ]
```

参数说明

- user_name：要修改的用户名。

- alias_name：用户的别名。
- password：用户口令字符串。
- role_name_1, role_name_2, ... ：角色名。
- date_time_expr：用户有效期截止时间的字符串，格式为日期或日期时间。
- opt_trust_ip：IPV4 地址表达式，外部用单引号包裹。表达式可以为如下形式：
  - 单个 IP 地址：例如'192.168.2.21'。
  - 多个 IP 地址：用逗号分隔的多个地址，例如'192.168.2.21,192.168.2.22,192.168.2.105'。
  - IP 地址范围；用减号分隔的地址上下界，例如'192.168.2.20-192.168.2.29'。
  - 任意 IP 地址：关键字'ANY'。
- opt_user_quotas：设置用户在数据库中各种资源上的配额限制。
- quantity_unit：设置数值的单位。例如要设置的资源配额为临时表空间，则单位可以为 M、G 等。
- encryptor_name：加密机（即加密用的密钥）的名称。

1.3.3.2 资源配额限制 opt_user_quotas

语法格式

```
opt_user_quotas::=
QUOTA int_value quantity_unit ON MEMORY
|    QUOTA UNLIMITED ON MEMORY
|    QUOTA int_value quantity_unit ON TEMP TABLESPACE
|    QUOTA UNLIMITED ON TEMP TABLESPACE
|    QUOTA int_value quantity_unit ON UNDO TABLESPACE
|    QUOTA UNLIMITED ON UNDO TABLESPACE
|    QUOTA int_value ON CURSOR
|    QUOTA UNLIMITED ON CURSOR
|    QUOTA int_value ON SESSION
|    QUOTA UNLIMITED ON SESSION
|    QUOTA int_value ON IO
|    QUOTA UNLIMITED ON IO
|    QUOTA int_value ON PROCEDURE
|    QUOTA UNLIMITED ON PROCEDURE
|    QUOTA int_value quantity_unit ON TABLESPACE
|    QUOTA UNLIMITED ON TABLESPACE
```

参数说明

- int_value：设置的数值，取值为整数。
- quantity_unit：设置数值的单位。例如要设置的资源配额为临时表空间，则单位可以为 M、G 等。
- UNLIMITED：无限制。
- MEMORY：内存配额。
- TEMP TABLESPACE：临时表空间配额。
- UNDO TABLESPACE：回滚表空间配额。
- CURSOR：游标配额。
- SESSION：会话配额。
- IO：I/O 配额。
- PROCEDURE：存储过程配额。
- TABLESPACE：表空间配额。

1.3.3.3 示例

修改名为 usr_test 的用户的信息。

```
SQL> ALTER USER usr_test
IDENTIFIED BY 'abcPAS135@#'
VALID UNTIL '2099-12-31 12:00:00'
ACCOUNT UNLOCK
'192.168.2.20-192.168.2.30';
```

1.3.4 删除用户

语法格式

```
DROP USER user_name [alter_behavior];
```

> ⚠ 注意
>
> 删除用户后，则所有属主为它的数据库对象均会被删除，所以删除用户前需慎重考虑或对其相关数据库对象进行备份。

参数说明

- user_name：待删除的用户名。
- alter_behavior：可选关键字 RESTRICT（默认值）或 CASCADE。
  - RESTRICT：删除用户时，只有在该用户及其对象没有被其他用户或模式对象依赖，才能成功删除。如果用户拥有其他被依赖对象，数据库返回错误，提示无法删除用户。
  - CASCADE：删除用户时，无论该用户及其对象是否被其他用户或模式对象依赖，都强制删除用户。

示例

强制删除用户 usr_test。

```
SQL> DROP USER usr_test CASCADE;
```

1.4 相关数据字典

1.4.1 系统表 SYS_USERS

用户系统表 SYS_USERS 记载当前库下所有用户的详细信息，如表1-1所示。可以在数据库中执行 SELECT*FROM sys_users 查看。

表 1-1 字段说明

字段名	字段类型	说明
DB_ID	INTEGER	（逻辑）库 ID
USER_ID	INTEGER	用户 ID
USER_NAME	CHAR(128)	用户名
IS_ROLE	BOOLEAN	该用户是否是角色
PASSWORD	BINARY	用户口令
START_TIME	DATETIME	用户账号有效期开始时间
UNTIL_TIME	DATETIME	用户账号有效期结束时间
LOCKED	BOOLEAN	用户账号是否被锁定
		接下页

字段名	字段类型	说明
EXPIRED	BOOLEAN	是否过期（过期后该用户可登录系统但只能进行口令重设操作）
PASS_SET_TIME	DATETIME	最近一次设置口令的时间
PASS_SET_PERIOD	INTEGER	口令设置周期
ALIAS	CHAR(128)	（逻辑）库 ID
IS_SYS	BOOLEAN	是否为系统用户
TRUST_IP	CHAR(128)	可信任 IP 范围
XLS_PID	INTEGER	安全策略 ID
XLS_LID	INTEGER	安全级别 ID
XLS_CIDS	BIGINT	安全范畴 ID
PRIORITY	INTEGER	执行优先级
TEMP_SPACE_QUOTA	INTEGER	临时表空间资源限额
CURSOR_QUOTA	INTEGER	安全策略 ID
SESSION_QUOTA	INTEGER	SESSION 资源限额
IO_QUOTA	INTEGER	IO 配额（单次命令最大 IO 次数）
PRIORITY	INTEGER	执行优先级
CREATE_TIME	DATETIME	用户创建时间
LAST_MODI_TIME	DATETIME	用户上次被更改时间
ENCRY_ID	INTEGER	加密机 ID

接下页

字段名	字段类型	说明
RESERVED2	CHAR(128)	保留，以备将来使用
RESERVED3	CHAR(128)	保留，以备将来使用

1.4.2 系统表 SYS_ALL_FORBIDDEN_IPS

系统表 SYS_ALL_FORBIDDEN_IPS 记载所有用户登录认证的失败情况息，如表1-2所示。可以在数据库中执行 SELECT*FROM sys_all_forbidden_ips 查看。

表 1-2 字段说明

字段名	字段类型	说明
NODEID	INTEGER	节点 ID
FORBIDDEN_IP	CHAR(128)	禁止登录的 IP 地址
CURR_FAILED_CNT	INTEGER	当前登录失败次数
MAX_FAILED_CNT	INTEGER	最大登录失败次数限制
CURR_T	DATETIME	当前时间
LAST_T	DATETIME	最后一次登录时间
FORBIDDEN_TIME	INTEGER	禁止登录时长
IS_FORBIDDEN	BOOLEAN	是否禁止登录

2 角色管理

2.1 角色介绍

角色是拥有同类数据库权限的集合，在用户权限复杂但可进行归类的系统中，可采用角色的管理方式进行权限管理，这样同类权限的用户可通过授予角色的方式获取权限，避免多用户授权的繁复操作。

用户拥有的权限确定其能对数据库进行哪些操作，权限可分为库级权限、模式级权限、表级权限、列级权限。由于数据库中对象很多，为每个用户设置细致的权限控制项目会显得十分复杂，考虑到一个数据库中的各个用户对数据库的操作权限在大多数应用场景下是可分类的，即某些用户具有相同的权限集合，而另一些用户具有另一种相同的权限集合。因此，用户在数据库中拥有的权限大致可由其在实际工作中担任的角色来确定。

虚谷数据库设立角色机制是为了方便权限管理，在虚谷数据库中可创建多个角色，一个用户可以加入一个或多个角色，从而继承所加入的角色组的权限，用户最终拥有的权限是其拥有的权限以及加入到的一个或多个角色具有的权限的合成权限。

继承权限是从父对象传播到子对象的权限。继承权限可以简化管理权限的任务，能够一次完成权限分配，使子对象拥有父对象的所有权限。比如在虚谷数据库中，需要对多个对象授予相同权限时，可以创建一个角色，对角色授予此权限，然后为角色指定成员用户。其中的角色就是父对象，所有在这个角色下的成员用户即为子对象。同样如需更改所有角色成员的权限，仅更改角色权限即可。如需要单独更改一个用户权限，将此用户从角色中移除，单独授权即可。角色删除后，原角色成员中用户拥有的权限被回收，仅拥有单独授予的权限或者归属于其他角色的权限。

2.2 角色操作

2.2.1 创建角色

语法格式

```
CreateRoleStmt::=
CREATE ROLE role_name [ { INIT | INITIALLY } user_name_1[,
   user_name_2 [, ...] ] ]
```

参数解释

- role_name: 要创建的角色名。
- user_name: 用户名。在创建角色时指定被授予该角色的用户。

示例

创建角色 role_1 并授予用户 usr_1 和用户 usr_2。

```
SQL> CREATE ROLE test_role1 INIT USER test_user1,test_user2;
```

2.2.2 授予角色

语法格式

```
AlterRoleStmt::=
GRANT ROLE role_name TO user_name_1[, user_name_2 [, ...] ]
```

参数解释

- role_name: 要授予的角色名。
- user_name_1, user_name_2, ...：角色被授予的用户名。

示例

角色 test_role1 下的所有用户即 test_user1、test_user2 拥有查询任何表的权限，角色 test_role2 下的所有用户即 test_user2、test_user3 拥有更新任何表的权限。对于用户来说 test_user1 用户仅拥有查询任何表权限；test_user2 用户既拥有查询任何表权限，又拥有更新任何表权限；test_user3 仅拥有更新任何表权限。

```
创建用户：
CREATE USER test_user1 IDENTIFIED BY '1234@ABCD';
CREATE USER test_user2 IDENTIFIED BY '1234@ABCD';
CREATE USER test_user3 IDENTIFIED BY '1234@ABCD';

创建角色：
CREATE ROLE test_role1;
CREATE ROLE test_role2;

将指定用户指定为test_role1角色的成员用户：
GRANT ROLE test_role1 TO test_user1,test_user2;

将指定用户指定为test_role2角色的成员用户：
GRANT ROLE test_role2 TO test_user2,test_user3;

为角色授予权限：
GRANT SELECT ANY TABLE TO test_role1;
GRANT UPDATE ANY TABLE TO test_role2;
```

2.2.3 收回角色

语法格式

```
AlterRoleStmt::=
REVOKE ROLE role_name FROM user_name_1[, user_name_2 [, ...] ]
```

参数解释

- role_name：要收回的角色名。
- user_name_1, user_name_2, ...：角色被收回的用户名。收回用户角色的前提是这些用户已经拥有了该角色，否则会报错。

示例

收回用户 usr_1 和用户 usr_2 的角色 role_1。

```
REVOKE ROLE role_1 FROM usr_1, usr_2;
```

2.2.4 删除角色

语法格式

```
DropRoleStmt::=
DROP ROLE role_name
```

参数解释

role_name: 要删除的角色名。

示例

```
DROP ROLE test_role1;
```

2.2.5 查看角色信息

查看数据库中拥有的角色需要在系统库中查询 sys_users，筛选判断为角色（is_role='T'）的对象名称。

语法格式

```
SELECT db_name,user_name FROM sys_users su,sys_databases sd WHERE
   su.db_id=sd.db_id and su.is_role='T';
```

参数解释

- db_name：角色所在数据库名称。
- user_name：角色名称。
- db_id：角色所在数据库库 ID。
- is_role：判断是否为角色。

2.2.6 查看角色成员信息

系统表 SYS_ROLE_MEMBERS 记载当前库下数据库角色成员相关信息，字段说明如表2-1所示。

表 2-1 字段说明

字段名	字段类型	说明
DB_ID	OID_TYPE	所属库 ID
USER_ID	OID_TYPE	用户 ID
ROLE_ID	OID_TYPE	角色 ID

示例

```
创建用户：
SQL> CREATE USER role_info IDENTIFIED BY '1234@abcd';

将用户加入到角色中：
SQL> GRANT ROLE db_admin TO role_info;

查询角色名称与角色id:
SQL> SELECT su.user_name,srm.role_id FROM sys_role_members srm JOIN
    sys_users su USING(db_id,user_id) WHERE user_name='ROLE_INFO';

USER_NAME | ROLE_ID |
-----------------------------------------------
ROLE_INFO| 5 |
```

以上示例为查询用户所属的角色信息。

3 权限管理

3.1 权限介绍

在虚谷数据库中，权限的最原始来源为天然权限，作为各级对象的拥有者自然具备该对象的天然权限，天然权限几乎等于该对象的一切权限（除数据库拥有者不能同时作为安全员及审计员外），也就是数据库的拥有者作为 DBO（Database Owner）自然拥有数据库的一切权限。即使未对该拥有者赋予任何权限，只要属主为该拥有者就可以对属主是他的对象进行一切操作。例如：新用户 A 未被赋予任何权限，如果用户 B 在用户 A 模式下创建了表对象，则用户 A 拥有对该表的一切操作权限，包括增删改查以及删除表。

通过 DBO 授权方能产生 DBA、审计员、安全管理员、可创建资源的用户等，DBA 或创建资源的用户可以创建表级资源，如：表、视图、存储过程、序列值等对象，创建者自然作为这些对象的拥有者，具有这些对象上的一切权限。DBA 也具有在一切表级对象上的全部权限，当创建对象的用户被删除后，该用户的数据库对象也一并删除。权限的授予者或 DBA 可以从被授权者收回全部或部分权限，若非 DBA，则只能收回其授出的权限，不能收回其他用户授出的权限。

在虚谷数据库系统中，权限继承主要针对用户与角色或角色与角色，用户继承其所属角色所有权限；用户之间不存在权限继承关系。

3.2 权限分类

在虚谷数据库中权限可以分为四类，分别是库级权限、模式级权限、对象级权限、列级权限，其作用范围如下：

- 库级权限：管理粒度为整个逻辑库，拥有库级权限的用户或角色可对该库下所有对象进行权限允许范围内的操作。
- 模式级权限：管理粒度为模式，拥有模式级权限的用户或角色可对该模式下所有对象进行权限允许范围内的操作。
- 对象级权限：管理粒度为对象，拥有对象级权限的用户或角色可对指定对象进行权限允许范围内的操作。
- 列级权限：管理粒度为表列或视图列，拥有列级权限的用户或角色可访问和操作权限允许

范围内的表列或视图列。

3.3 权限操作

3.3.1 库级权限

语法格式

- 授予库级权限

```
GrantStmt::=
GRANT sys_privilege [,sys_privilege] TO grantee_list [WITH GRANT
   OPTION]
```

> 说明
>
> 授权操作类型与操作对象类型应保证匹配，例如针对 PROCEDURE 对象，可为其赋予 CREATE、ALTER、DROP、EXECUTE 等权限，但不能赋予 INSERT、UPDATE、DELETE、SELECT 等权限。

- 回收库级权限

```
RevokeStmt::=
REVOKE GRANT OPTION FOR sys_privilege [,sys_privilege] FROM
   grantee_list
|   REVOKE sys_privilege [,sys_privilege] FROM grantee_list
```

- 权限信息

```
sys_privilege::=
{DBA | SSO | AUDITOR | BACKUP DATABASE | BACKUP | RESTORE |
   RESTORE DATABASE | sys_operation obj_type}

sys_operation::=
{CREATE | CREATE ANY | ALTER ANY | DROP ANY | SELECT ANY | INSERT
    ANY | UPDATE ANY | DELETE ANY | EXECUTE ANY | REFERENCES ANY
    | VACUUM ANY | ENCRYPT ANY}

obj_type::=
{DATABASE | SCHEMA | TABLE | SEQUENCE | INDEX | VIEW | PROCEDURE
    | PACKAGE | TRIGGER | DATABASE LINK | REPLICATION | SYNONYM
    | PUBLIC SYNONYM | USER | ROLE | JOB}

grantee_list::=
{ROLE  UserId | UserId} [{,} {ROLE  UserId | UserId}]...
```

参数说明

- sys_privilege：库级权限。

- grantee_list：被授予/回收权限角色组或用户组，可同时授予/回收多个用户或角色，若角色权限授予/回收则角色下的所有用户均自动授予/回收该权限。
- WITH GRANT OPTION：权限可转授，权限的赋予和回收是关联的，如将 WITH GRANT OPTION 用于对象授权时，被授予的用户也可把此对象权限授予其他用户或角色，权限回收时转授的权限会一同被收回，WITH GRANT OPTION 只能在授予对象级和列级权限时使用。
- REVOKE GRANT OPTION FOR：回收 GRANT 语句中指定的 WITH GRANT OPTION 的转授权限，用户仍然具有该权限，但是不能将该权限转授予其他用户。
- sys_operation：操作类型，包含 DDL、DML 类型，如：CREATE、ALTER、DROP、INSERT、DELETE、UPDATE、SELECT 等，若选择 ANY 参数则表示被授权对象可跨模式进行操作，否则只能在其所有模式进行相应操作。
- obj_type：表示操作对象类型，包含 TABLE、VIEW、PROCEDURE 等。（SYNONYM 为私有同义词、PUBLIC SYNONYM 为全局同义词）

示例

授予 test_user 在当前库下所有模式的创表权限，若无 ANY 关键字则表示，test_user 只能在属于他的模式下进行创表。

```
GRANT CREATE ANY TABLE TO test_user;
```

3.3.2 模式级权限

语法格式

- 授予模式级权限

```
GrantStmt::=
GRANT sys_privilege [,sys_privilege] IN SCHEMA name TO
   grantee_list [WITH GRANT OPTION]
```

- 回收模式级权限

```
RevokeStmt::=
REVOKE GRANT OPTION FOR sys_privilege [,sys_privilege] IN SCHEMA
   name FROM grantee_list
|    REVOKE sys_privilege [,sys_privilege] IN SCHEMA name FROM
   grantee_list
```

- 权限信息

```
sys_privilege::=
{DBA | SSO | AUDITOR | BACKUP DATABASE | BACKUP | RESTORE |
   RESTORE DATABASE | sys_operation obj_type}

sys_operation::=
{CREATE | CREATE ANY | ALTER ANY | DROP ANY | SELECT ANY | INSERT
    ANY | UPDATE ANY | DELETE ANY | EXECUTE ANY | REFERENCES ANY
    | VACUUM ANY | ENCRYPT ANY}

obj_type::=
{DATABASE | SCHEMA | TABLE | SEQUENCE | INDEX | VIEW | PROCEDURE
    | PACKAGE | TRIGGER | DATABASE LINK | REPLICATION | SYNONYM
    | PUBLIC SYNONYM | USER | ROLE | JOB}

grantee_list::=
{ROLE  UserId | UserId} [{,} {ROLE  UserId | UserId}]...
```

参数说明

- sys_privilege：库级权限。
- grantee_list：被授予/回收权限角色组或用户组，可同时授予/回收多个用户或角色，若角色权限授予/回收则角色下的所有用户均自动授予/回收该权限。
- WITH GRANT OPTION：权限可转授，权限的赋予和回收是关联的，如将 WITH GRANT OPTION 用于对象授权时，被授予的用户也可把此对象权限授予其他用户或角色，权限回收时转授的权限会一同被收回，WITH GRANT OPTION 只能在授予对象级和列级权限时使用。
- REVOKE GRANT OPTION FOR：回收 GRANT 语句中指定的 WITH GRANT OPTION 的转授权限，用户仍然具有该权限，但是不能将该权限转授予其他用户。
- sys_operation：操作类型，包含 DDL、DML 类型，如：CREATE、ALTER、DROP、INSERT、DELETE、UPDATE、SELECT 等，若选择 ANY 参数则表示被授权对象可跨模式进行操作，否则只能在其所有模式进行相应操作。
- obj_type：表示操作对象类型，包含 TABLE、VIEW、PROCEDURE 等。（SYNONYM 为私有同义词、PUBLIC SYNONYM 为全局同义词）

示例

授予角色 role_1 在模式 SYSDBA 下的创表权限。

```
GRANT CREATE ANY TABLE IN SCHEMA SYSDBA TO role_1;
```

3.3.3 对象级权限

语法格式

- 授予对象级权限

```
GrantStmt::=
GRANT privileges ON [TABLE | VIEW | PROCEDURE | SEQUENCE]
   name_space TO grantee_list [WITH GRANT OPTION]
```

- 回收对象级权限

```
RevokeStmt::=
REVOKE privileges ON [TABLE | VIEW | PROCEDURE | PACKAGE |
   SEQUENCE] name_space FROM grantee_list
|    REVOKE GRANT OPTION FOR privileges ON name_space FROM
   grantee_list
```

- 权限信息

```
privileges::=
ALL PRIVILEGES
|    ALL
|    operation [,operation]...

operation::=
{SELECT | INSERT | UPDATE | DELETE | EXECUTE | REFERENCES | ALTER
    | DROP | INDEX | TRIGGER | VACUUM}

grantee_list::=
{ROLE  UserId | UserId} [{,} {ROLE  UserId | UserId}]...
```

参数说明

- privileges：授予/回收权限操作类型，包括 SELECT、UPDATE、EXECUTE 等，若要授予/回收对象的所有可操作权限，可使用 ALL 或 ALL PRIVILEGES 代替。
- name_space：此处为指定授予/回收操作权限对象名，对象名与 privileges 操作类型必须匹配，如该对象为 TABLE，则不能授予 EXECUTE 权限。
- grantee_list：被授予/回收权限的用户名或角色名。
- WITH GRANT OPTION：权限可转授，权限的赋予和回收是关联的，如将 WITH GRANT OPTION 用于对象授权时，被授予的用户也可把此对象权限授予其他用户或角色，权限回收时转授的权限会一同被收回，WITH GRANT OPTION 只能在授予对象级和列级权限时使用。
- REVOKE GRANT OPTION FOR：回收 GRANT 语句中指定的 WITH GRANT OPTION 的转授权限，用户仍然具有该权限，但是不能将该权限转授予其他用户。

示例

- 示例 1

  授予用户 U1 对于表 test_permission 的数据插入权限，则用户 U1 对表 test_permisssion 无删除、查询、更改、引用等权限。

```
GRANT CREATE ANY TABLE IN SCHEMA SYSDBA TO role_1;
```

- 示例 2

  授予用户 U1 对于表 test_premission 所有数据库所允许的操作权限，包括：INSERT、UPDATE、DELETE、SELECT、REFERENCES 等。

```
GRANT ALL ON test_permission TO u1;
```

3.3.4 列级权限

语法格式

- 授予列级权限

```
GrantStmt::=
GRANT operation [,operation]... ( name_list ) ON [TABLE]
   name_space TO grantee_list [WITH GRANT OPTION]
```

- 回收列级权限

```
RevokeStmt::=
REVOKE GRANT OPTION FOR operation_commalist ( name_list ) ON
   name_space FROM grantee_list
|    REVOKE operation_commalist ( name_list ) ON [TABLE]
   name_space FROM grantee_list
```

- 权限信息

```
operation::=
{SELECT | INSERT | UPDATE | DELETE | EXECUTE | REFERENCES | ALTER
    | DROP | INDEX | TRIGGER | VACUUM}

grantee_list::=
{ROLE  UserId | UserId} [{,} {ROLE  UserId | UserId}]...
```

> **说明**
>
> 列级权限只包括 SELECT 与 UPDATE 操作权限，且必须指定可操作的对象名，对象类型只能是表或视图。

参数说明

- operation：表示操作类型，包含 DDL、DML 类型，如 CREATE、ALTER、DROP、INSERT、DELETE、UPDATE、SELECT 等。
- name_list：授予/回收可操作的列名。
- name_space：此处指授予/回收可操作的对象名，可为表名或视图名。
- grantee_list：授予/回收权限的用户名或角色名。

示例

授予用户 TU1 针对表 TEST_COL_PRE 的列 ID2 的查询权限与列 ID 的变更权限。

```
CREATE TABLE TEST_COL_PRE(ID INT,ID2 INT,ID3 INT);
CREATE USER TU1 IDENTIFIED BY 'test_123@';
GRANT SELECT(ID2) ON TEST_COL_PRE TO TU1;
GRANT UPDATE(ID) ON TEST_COL_PRE TO TU1;
```

4 对象管理

4.1 表对象管理

数据库核心功能为数据存储，数据存储于数据表中，若数据表设计不规范，不仅用户使用不便，也会导致业务系统访问效率低下，所以在业务系统建设过程中，数据表设计建议遵循以下标准：

- 符合数据库设计第三范式。
- 普通表记录数建议不超过 5000 万，若存储资料数据量超过 5000 万，建议采用分区表进行数据存储，可根据业务需求采用范围分区、列表分区、HASH 分区等，同时若数据基于时间特征，可考虑采用基于时间的自动扩展分区。
- 以时间为分区键的数据表建议采用自动扩展分区，最小自动扩展分区间隔粒度为天，单分区内存储数据建议不超过 5000 万（若 1 天间隔内数据量超过 5000 万则以最小时间间隔粒度为准）。
- 数据表设计时应考虑预留字段，减少字段扩展带来的对象结构变更影响。
- 数值类型建议采用 NUMERIC，尽量避免使用 DOUBLE 与 FLOAT，且应预留 NUMERIC 精度与标度，否则扩展 NUMERIC 精度后，针对已入库数据的变更字段数据修改需采用 DELETE+INSERT，直接使用 UPDATE 可能导致数据异常。
- 时间类型建议采用 DATETIME，若需自动使用系统时间填充空数据，可使用 TIMESTAMP。
- 字符类型建议采用 VARCHAR，后期可直接扩展，若使用 CHAR 类型则在精度扩展时需重整数据，数据表中数据量越大，时间开销越高。

4.2 表对象结构变更管理

变更表需确认是否为同步任务表，如果是同步任务表，变更操作需要同时更改源端对象与目标端对象。

表结构变更注意事项：

- 在表上添加字段，若添加字段只有列名与数据类型信息（无约束与默认值），可以直接使用命令进行添加，此时无数据重整可快速完成变更。
- 若添加字段包含默认值、约束信息，此时添加字段会对历史数据进行重整，历史数据量越大，此操作耗时越长，且 DDL 操作会锁表，不建议进行此类操作，如确有需要，可在协商解决方案后进行（建议数据库表对象设计时考虑增加冗余字段，避免重整表的时间开销）。
- 删除表字段，此操作将对历史数据进行重整，数据量越大，此操作耗时越长，同时 DDL 操作将会锁表，不建议进行此类操作，可考虑将废除字段重命名为备份字段；如删除字段操作确有需要，可由数据库管理员提供相应解决方案后进行。
- 禁止进行表删除与表清理操作（TRUNCATE），如需进行该类操作，必须通过业务系统管理员与数据库管理员审核方可进行。
- 慎重进行表字段数据类型变更操作，非空数据的不兼容数据类型变更不被允许，针对变长字符类型 VARCHAR 数据类型精度变更只能扩大，不能缩小；针对定长字符类型 CHAR 数据类型精度变更只能扩大，同时扩展需重整数据，此操作可考虑将 CHAR 变更为 VARCHAR，既可实现精度扩展也可不重整数据；针对数值类型 NUMERIC 数据类型变更，支持向高精度类型进行扩展，但若对变更字段进行数据值修改，必须对该条记录进行删除后重新插入，不能直接变更该字段为高精度数据，否则会导致数据异常。
- 表字段添加、修改或删除空、非空、默认值约束，可直接进行修改，不会产生数据操作，但需保证操作期间无业务访问和挂起事务操作该对象，否则无法进行对象加锁。
- 可在表上预留部分字段，留待后期添加字段时，直接重命名。

4.3 表对象数据操作管理

对于数据库表对象中表数据的更新、删除操作应当谨慎，执行操作前认真审核执行 SQL 语句，且变更操作必须编写 WHERE 条件语句，并验证条件语句正确性；限制表对象数据全表更新、删除操作，对于大规模数据量的全表操作需提交申请，由数据库协助执行。

表数据操作注意事项：

- 所有表数据操作需根据数据库设计，无论访问还是变更/删除，均建议提供分区键或索引键，避免全表数据扫描操作导致 I/O 消耗过高与响应效率低下。

- 对于大数据量的表操作，需确定操作方式及合理性。如：是否使用 TRUNCATE 分区；使用 DELETE 语句是否合适，WHERE 条件语句如何给定，执行多少条变更进行 COMMIT 操作。
- 禁止 TRUNCATE 整表操作，此操作仅由管理员进行。
- 操作数据时，SQL 语句应指定需要的列，不建议使用 * 或者全表字段进行数据操作。

4.4 表对象索引管理

表索引的建立应当遵循合理、精简的原则，防止无效、冗余索引的建立，降低索引维护开销。对于较大数据量的表，依据应用系统业务逻辑，制定表分区方案，对表数据进行裁剪，提高数据访问效率。

索引使用注意事项：

- 单个索引中的字段数不超过 5 个，避免高精度变长字段作为索引字段。
- 单张表中索引数量不超过 5 个。
- 索引字段的选择应避免使用重复度高的字段，若字段重复度过高无需对其建立索引。
- 分区表的索引的头字段需使用分区键，并且索引类型使用局部索引，避免使用全局索引，否则在清理表分区或删除表分区时需重建全局索引。
- 每张表原则上只使用唯一一个唯一值索引，其他索引均为普通索引；唯一值索引设计一定要准确规范，否则后期数据清洗与重整操作开销代价很大。
- 创建索引时，索引字段数、表数据量不同，创建索引耗时会有较大差别，对于数据量较大的表（超过 5000 万），需评估时效开销。
- 索引字段设计应与访问业务应用最大化匹配，保证能够提供精准的索引裁剪，提升访问效率。若业务系统访问服务接口对数据的访问存在访问条件为 A+B、A+C+D、A+E 的模式，需对应建立匹配的索引，不能直接建立 A+B+C+D+E 的索引，该索引在 A+C 与 A+E 的访问接口下实际仅进行首字段匹配裁剪，无法达到精确定位的效果。
- 索引设计应在业务系统建设初期进行考虑，避免在业务系统服务过程中进行结构变更，所以索引设计除了要考虑数据变更业务，还要考虑查询业务需求。

4.5 连接管理

数据库连接的建立应当做好规划，不使用的数据库连接及时关闭，并对数据库连接做好负载均衡，防止数据库连接集中于某一台工作节点；数据库连接应当遵循事务整体一致性原则，非自动提交连接的事务执行完成后，必须提交或者回滚变更事务。

数据库连接使用注意事项：

- 长时间未使用的数据库连接，配置有连接空闲时间 MAX_IDLE_TIME，超过空闲时间会被清理；若想保持长连接，可使用连接池，通过配置连接活性检测，从而保持长连接。
- 数据库连接应按需申请，不使用的连接应及时关闭，同时，连接上申请的对象资源 Statement、PrepareStatement 等在使用完后也应进行关闭操作，以免占用过多资源，影响系统使用。
- 定期通过虚谷数据库集群监控软件，查看数据库连接变化，分析业务访问情况。
- 需要确定单个 IP 地址允许建立数据库连接的最大值，加强 IP 地址连接资源管理。

4.6 长耗时、统计分析类事务管理

对于长耗时、统计分析类事务，由于其对系统资源的消耗比较大，需制定合理的运行策略，降低对在线业务系统的冲击，原则上应用访问数据范围仅包括所需内容，减少全表、无关数据的访问；对于此类资源消耗较多的事务，可调用系统资源使用数据，结合执行结果分析系统资源瓶颈，在系统资源遇见瓶颈时，及时改善系统资源。

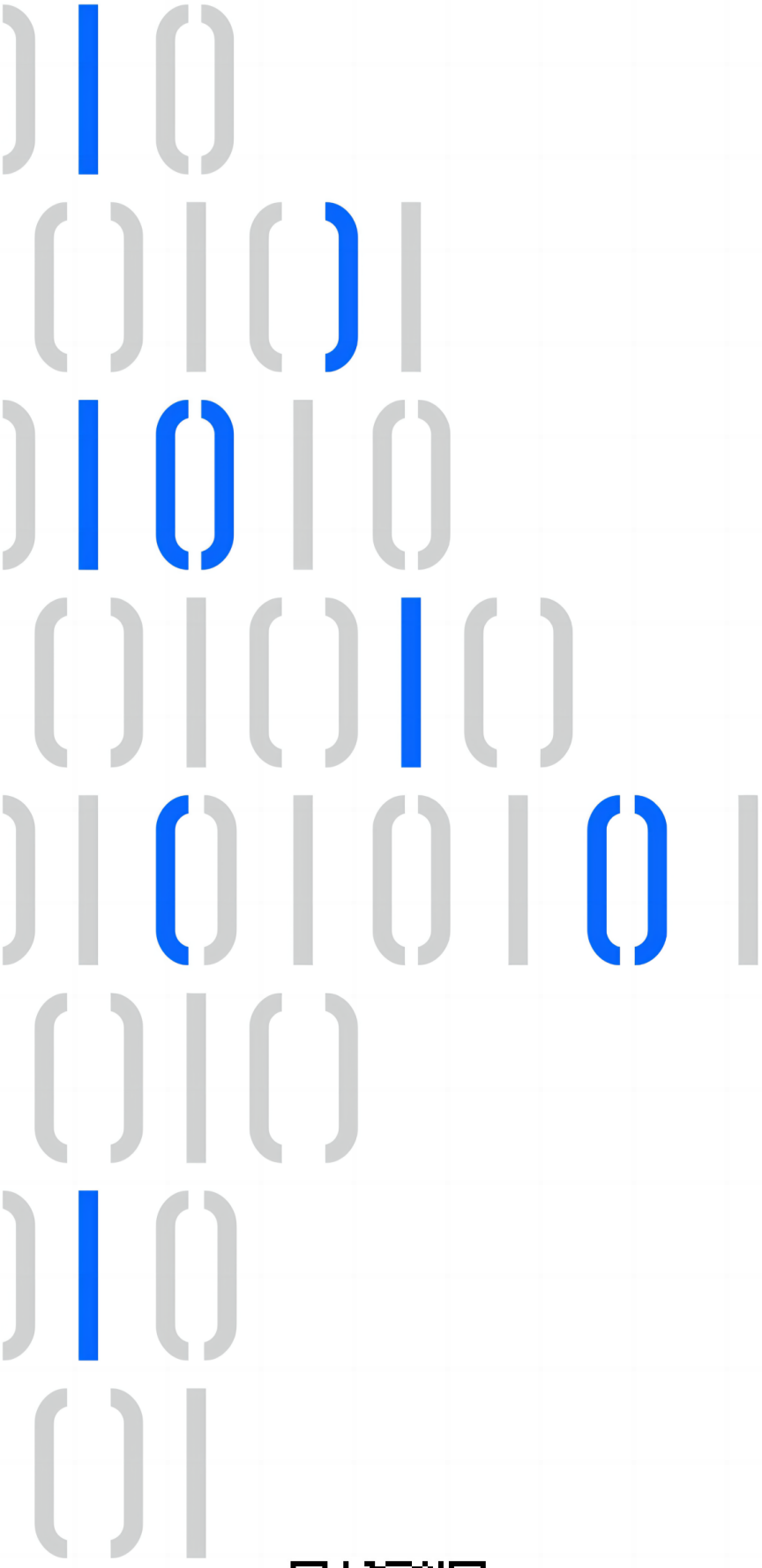
- 对于此类事务建议先做执行路径分析，若未走索引路径，则需要调整 SQL 语句，让该业务走索引查询。
- 对于可预估的长耗时变更事务，可考虑降低数据范围，切片成小范围的多次执行，提高效率。
- 对于查询、统计类等适合数据分析类的场景，建议在分析库上进行，充分利用分析库优势。

4.7 SQL 性能优化

对于一些应用系统响应时长较长的情况，可通过数据库跟踪日志，提取应用系统执行 SQL，分析 SQL 执行路径，优化业务逻辑或 SQL 执行路径，提高 SQL 执行性能。

- 区分 OLTP 与 OLAP 系统：针对 OLTP 与 OLAP 业务场景，建议建立相应的数据库服务系统，避免 OLAP 的高资源消耗影响 OLTP 业务响应效率。
- 使用索引与分区键：编写 SQL 查询时，务必确认查询是否利用了分区键和索引键，并通过 EXPLAIN 命令分析执行计划，确保尽可能多地使用索引扫描而非全表扫描，以提高检索速度。
- JOIN 优化策略：在多表联合查询中，应优先将关联键重复较少的表放在 JOIN 的左侧进行关联，以此减少不必要的运算处理。同时，在应用 WHERE 条件时，优先考虑那些能大幅裁剪结果集的条件，特别是对于数据量较小且选择性高的表，从而进一步优化查询性能。
- 创建适当的索引：涉及多表联合查询时，应在关联键上创建相应的索引，特别是在主外键关系中，对外键也应设计合理的索引。此外，当检索大量数据时，可以通过缩小时间范围并采用多次访问的方式，降低磁盘扫描的可能性，从而提升查询效率。
- 使用表别名：在连接多个表时，推荐使用表的别名并将别名前缀于每个列名之前，这不仅能减少解析时间，还能避免因列名歧义导致的语法错误。
- SELECT 子句优化：SELECT 子句中避免使用 *。在解析的过程中, 数据库通过查询数据字典，将 * 依次转换成所有的列名。这意味着将耗费更多的时间，并且会返回可能不使用的列数据，加重运算和网络开销。
- 限制子查询使用：除非必要，尽量避免使用子查询，子查询通常开销较大。同时，如果业务逻辑允许，减少排序操作也可以显著提高数据响应效率。

遵循以上原则，可以帮助优化 SQL 查询，减少数据库系统的负载，进而提升整体性能和响应速度。



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