优炫数据库管理员手册 2.1

UXSINO
优 炫 软 件

优炫数据库管理员手册 2.1

版权 © 2016-2023 北京优炫软件股份有限公司

法律声明

优炫数据库管理系统(简称：UXDB) 是由北京优炫软件股份有限公司开发并发布的一款商业性数据库管理系统。

优炫数据库管理系统(UXDB)的一切知识产权以及与该软件产品相关的所有信息内容，包括但不限：文字表述及其组合、图标、图饰、图表、色彩、界面设计、版面框架、有关数据、及电子文档等均属北京优炫软件股份有限公司所有。本软件及其文档的任何使用、复制、修改、出租、传播、销售及分发等行为均须经北京优炫软件股份有限公司书面许可。

凡侵犯北京优炫软件股份有限公司知识产权的行为，北京优炫软件股份有限公司将依法追究其法律责任。

本声明的最终解释权归属于北京优炫软件股份有限公司。

UXSINO和其他优炫公司商标均为北京优炫软件股份有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

由于产品版本安装或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

北京优炫软件股份有限公司（总部）

- 地址：北京市海淀区学院南路62号中关村资本大厦11层（邮编：100081）
- 网址：http://www.uxsino.com
- 邮箱：<uxdb_support@uxsino.com>
- 电话：010-82886998
- 传真：010-82886338
- 服务热线：400-650-7837

目录

前言 vi
1. 文档目的 vi
2. 文档对象 vi
3. 修改记录 vi
1. 产品简介 1
1.1. 概述 1
1.2. 产品优势 1
1.3. 版本分类 1
2. 实例管理 3
2.1. 概述 3
2.2. 创建实例 3
2.2.1. 使用initdb创建实例 3
2.2.2. 使用ux_ctl 4
2.3. 删除实例 4
2.3.1. 使用removedb工具 4
2.4. 查看实例状态 5
2.4.1. 使用ux_ctl status 5
2.4.2. 使用uxdb 5
3. 启动和关闭服务 6
3.1. 启动 6
3.1.1. 使用ux_ctl 6
3.1.2. 使用uxdb 7
3.2. 关闭 8
3.2.1. 使用ux_ctl 8
3.2.2. 使用kill 9
4. 管理数据库 10
4.1. 概述 10
4.2. 创建一个数据库 10
4.3. 模板数据库 11
4.4. 数据库配置 12
4.5. 销毁一个数据库 13
4.6. 表空间 13
5. 用户和权限 15
5.1. 用户 15
5.2. 权限 15
5.3. 通过图形界面管理用户和权限 15
5.3.1. 新建用户 15
5.3.2. “属性”页 16
5.3.3. “角色”页 16
5.3.4. “设置”页 16
5.3.5. “权限”页 17
5.3.6. “源”页 18
5.3.7. 删除用户 18
5.4. 通过SQL管理用户和权限 18
5.4.1. 查看用户账户信息 18
5.4.2. CREATE USER 19
5.4.3. ALTER USER 19
5.4.4. DROP USER 19
5.4.5. 权限 19
5.4.6. createuser和dropuser 21
6. 数据库角色 23

6.1. 数据库角色 23
6.2. 角色属性 24
6.3. 角色成员关系 25
6.4. 删除角色 26
6.5. 默认角色 27
6.6. 函数和触发器安全性 28
7. 模式对象管理 29
7.1. 模式管理 29
7.1.1. 关于模式 29
7.1.2. 基于图形界面的模式管理 29
7.1.3. 基于SQL语句的模式管理 32
7.2. 表管理 34
7.2.1. 表 34
7.2.2. 基于图形界面的表管理 36
7.2.3. 基于SQL语句的表管理 54
7.3. 视图 58
7.3.1. 关于视图 58
7.3.2. 基于图形界面的视图管理 58
7.3.3. 基于SQL语句的视图管理 63
7.4. 索引 64
7.4.1. 关于索引 64
7.4.2. 查看索引 65
7.4.3. CREATE INDEX 65
7.4.4. ALTER INDEX 66
7.4.5. REINDEX 67
7.4.6. DROP INDEX 67
7.5. 存储过程/函数 68
7.5.1. 关于存储过程/函数 68
7.5.2. 基于图形界面的存储过程/函数管理 68
7.5.3. 基于SQL语句的存储过程/函数管理 71
7.6. 触发器 75
7.6.1. 关于触发器 75
7.6.2. 表级触发器 75
7.6.3. 事件触发器 76
7.6.4. 基于图形界面的表级触发器管理 79
7.7. 序列管理 81
7.7.1. 序列发生器 81
7.7.2. 基于图形界面的序列管理 81
7.7.3. SQL语句 83
8. 事务管理 85
8.1. 数据库事务 85
8.1.1. 事务的概念 85
8.1.2. 事务的特性 85
8.1.3. 数据库对事务的管理 86
8.2. 数据的一致性和并发控制 87
8.2.1. 事务的隔离级别 87
8.2.2. 并发控制的实现 89
9. 存储管理 91
9.1. 表空间 91
9.1.1. 关于表空间 91
9.1.2. 基于图形界面的表空间管理 91
9.1.3. SQL语句 93

表格清单

1. 文档更新记录 vi
6.1. 默认角色 27
7.1. 模式管理“基本属性”页参数说明 31
7.2. DROP SCHEMA参数说明 34
7.3. “基本属性”说明 37
7.4. “字段”页说明 38
7.5. “约束”页说明 39
7.6. “分区”页说明 45
7.7. “基本属性”页说明 59
7.8. 物化视图的“基本属性”页说明 62
7.9. DROP INDEX 参数说明 68
7.10. 过程/函数参数说明 69
7.11. 存储过程“基本属性”页参数说明 70
7.12. 表级触发器“基本属性”页参数说明 80
7.13. 序列管理“基本属性”页参数说明 82
8.1. 事务的隔离级别 87
8.2. 锁 89

前言

1. 文档目的

本文档主要为数据库管理员的日常工作提供必要信息。

2. 文档对象

- 技术支持工程师
- 维护工程师

3. 修改记录

修改记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

表 1. 文档更新记录

工具版本	发布日期	修改说明
2.1.1.5C	2022-09-21	第一次正式发布。

第 1 章 产品简介

1.1. 概述

UXDB是北京优炫软件技术股份有限公司（简称：北京优炫）经过多年努力自主研发的、具有自主知识产权的商用关系型数据库管理系统（DBMS）。该产品支持严格的ACID特性，结合多核架构的极致性能，行业最高的安全标准，以及完备的高可用方案，并提供可覆盖迁移、开发及运维管理全使用周期的智能便捷工具。产品融合了北京优炫在数据库领域几十年的产品研发和企业级应用经验，可满足各行业用户多种场景的数据处理需求。

优炫数据库系统主要包含：优炫数据库（以下简称UXDB）、优炫数据库管理平台（以下简称UXDBWeb）和优炫数据库图形化开发软件（以下简称UXDBAdmin）。UXDB是一款数据库软件，包含数据处理引擎（DB）、高可用工具、数据迁移工具等。 UXDBWeb和UXDBAdmin是两款独立的软件，两者的不同在于使用对象和管理内容不同，UXDBWeb安装在管理端，供管理人员使用，主要实现对UXDB的DB部分、系统、安全部分的监控和管理；UXDBAdmin一般安装在客户端，供开发人员使用，主要通过sql语句，实现对DB部分基本操作。

1.2. 产品优势

UXDB产品系列在系统的可靠性、可用性、性能和兼容性等方面进行了重大改进。

- 高度容错，稳定可靠

  针对企业级关键业务应用的可持续服务需求，UXDB提供可在电力、金融、电信等核心业务系统中久经考验的容错功能体系，通过如数据备份、恢复、同步复制、多数据副本等高可用技术，确保数据库7×24小时不间断服务，实现99.999%的系统可用性。

- 应用迁移，简单高效

  针对从异构数据库将应用迁移到UXDB的场景，UXDB一方面通过智能便捷的数据迁移工具，实现无损、快速数据迁移；另一方面，UXDB还提供高度符合标准（如SQL、ODBC、JDBC等）、并兼容主流数据库（支持Oracle等主流数据库的部分语法，针对MySQL等各类数据源都能实现无损、平滑、快速迁移）语法的服务器端、客户端应用开发接口，可最大限度地降低迁移成本。此外，UXDB迁移工具能够自动选择迁移失败的对象，实现再次迁移。

- 性能强劲，扩展性强

  针对企业业务增长带来的数据库并发处理压力，该版本提供了包括并行计算、索引覆盖等技术在内的多种性能优化手段，此外提供了基于读写分离的负载均衡技术，让企业能从容应对高负载大并发的业务。

1.3. 版本分类

UXDB支持多种操作系统和硬件平台：支持Unix、Linux和Windows等数十个操作系统产品版本；支持X86、X86_64及国产龙芯、飞腾、申威、银河麒麟、UOS等CPU硬件体系结构。并具备与这些版本服务器和管理工具之间的无缝互操作能力。

针对不同类型的客户需求，UXDB设计并实现了标准版、企业版等多类版本，满足各种业务场景对通用数据库管理系统的技术需求。UXDB数据库应用程序可从笔记本电脑扩展到台式机、大型数据库服务器，以至整个企业网络，而无需重新设计。此外，当用户业务发展需更大的数据处理能力时，UXDB还支持各个版本之间的平滑升级。

- 标准版

  UXDB标准版是向政府部门、中小型企业及互联网应用数据处理需求提供的通用数据库管理系统，具备数据存储、处理、管理、开发支持等基本功能，产品具备高标准符合性，支持多用户并发访问，能满足各种中小型应用需要，具有高性价比。

- 企业版

  UXDB　企业版具备完整的数据库组件，面向政府部门、中大型企业及复杂互联网应用的联机事务处理、决策分析和数据仓库等应用需求，提供高性能、高安全、高可靠可用、高兼容、易使用易管理的企业级大型通用数据库管理系统，并具有海量数据管理和大规模并发处理能力，可支持多用户并发处理、　支持集群、支持多维数据分析、全文检索、地理信息系统、图片搜索等复杂功能。并在标准版基础上提供物理同步、逻辑同步、性能优化、运行分析诊断等高级功能。

第 2 章 实例管理

2.1. 概述

优炫数据库管理系统UXDB，由数据库文件和UXDB实例组成。

- 数据库文件：数据库文件为存储用户数据以及元数据的一组磁盘文件。
- 元数据：是描述数据库结构、配置和控制有关的信息。

UXDB数据库实例管理的所有数据在物理上都以操作系统文件的方式存放在磁盘上。

UXDB实例具有如下特点。

1. 包含若干对存储的数据进行操作的数据库服务进程，还包括分配和管理内存，统计各种信息，以及实现各种协调工作的后台进程。一台设备上，可以同时运行多个实例。
2. 实例注册成功后，会有唯一的名字标志一个实例。
3. 一个UXDB实例在操作系统上表现为一个UXDB进程，它可以由控制器启动，也可以单独用命令行启动。
4. 一个UXDB实例管理多个逻辑上的数据库。启动一个UXDB实例后，使用客户端可以访问到这个实例管理的任意一个数据库。
5. 正常情况下，优炫数据库管理系统UXDB在安装过程中，会自行初始化注册一个UXDB实例。但如果数据库安装过程中初始化失败，请参见第 2.2 节 “创建实例”，手动注册UXDB实例。

2.2. 创建实例

2.2.1. 使用initdb创建实例

打开控制台工具，在命令工具中执行命令进入UXDB服务器的目录uxdbinstall/uxsql/bin，执行initdb的命令注册实例。

- 用法

  initdb [*option...*] [--uxdata | -D] *directory*

- 常用选项

  -D directory

  指定数据库集群应该存放的目录。这是initdb要求的唯一信息，但是你可以通过设定UXDBTA环境变量来避免书写它，这很方便因为之后数据库服务器（uxdb）可以使用同一个变量来找到数据库目录。

  [--pwfile=]*filename*

  让initdb从一个文件读取数据库超级用户的口令。该文件的第一行被当作口令。

  -W

  提示设置一个口令，该口令会被程序以MD5默认加密算法加密后存储至系统表ux_authid中的rolepassword字段中。

-U username

选择数据库超级用户的用户名。这个的默认值是实际运行initdb的用户的名称。超级用户的名字不重要，但是你可以选择保留常用的名字uxdb，即使操作系统的用户名不同。

详细请参见《优炫数据库 2.1 工具使用手册》initdb部分。

- 示例

使用命令initdb创建一个数据库实例，数据库实例的数据目录位置由-D选项指定（此处指定数据目录位置为/opt/uxdbinstall/dbsql/data）。

$ initdb -D /opt/uxdbinstall/dbsql/data

- 注意

1. 如果数据目录不存在，initdb命令将尝试创建-D选项指定的数据目录，但该操作可能因为权限不足而失败（若数据目录的父目录属于root用户）。
2. 不能以root用户运行initdb。

2.2.2. 使用ux_ctl

- 用法

ux_ctl init[db] [-D *datadir*] [-o *initdb-options*]

init或initdb模式将创建一个新的UXDB数据库实例，此模式调用initdb命令。

- 常用选项

-D datadir

指定数据库配置文件的文件系统位置。

-o initdb-options

指定要直接传递给initdb命令的选项。

详细请参见《优炫数据库 2.1 工具使用手册》ux_ctl部分。

- 示例

$ ux_ctl initdb -D /opt/uxdbinstall/dbsql/data -o '-W '

2.3. 删除实例

删除实例即移除UXDB数据库集群目录。

2.3.1. 使用removedb工具

当不再需要某服务时，数据库管理员可以将它移除。

- 用法

removedb［*option*...］［ --uxdata ｜ -D ］*directory*

- 常用选项

  -D directory

  　　指定数据库集群的数据目录示例。

  -F

  　　强制删除给定的数据库集群，小心使用这个选项。

- 示例

  $ **removedb -D /opt/uxdbinstall/dbsql/data**

- 注意

  在UXDB运行期间，禁止删除集群目录。

2.4.　查看实例状态

2.4.1.　使用ux_ctl status

status模式检查一个服务器是否运行在指定的数据目录中。如果存在正在运行的服务区，其PID和用来调用它的命令行选项将被显示。如果没有在运行的服务器，ux_ctl将返回退出状态3。如果没有指定一个可以访问的数据目录，ux_ctl将返回退出状态4。

- 用法

  ux_ctl status［-D *datadir*］

- 常用选项

  -D datadir

  　　指定数据库配置文件的文件系统位置。

- 示例

  显示服务器状态。

  $ **ux_ctl status**

2.4.2.　使用uxdb

可通过如下命令，查看数据库服务器当前状态。

$ **ps -ef|grep uxdb**

uxdb进程存在，即为正常开启状态，否则数据库服务器未启动。

第 3 章 启动和关闭服务

3.1. 启动

3.1.1. 使用ux_ctl

服务器管理工具ux_ctl将在后台启动服务器，并将输出存储在指定的日志文件中。

- 用法

  ux_ctl start [-D *datadir*] [-l *filename*] [-W] [-t *seconds*] [-s] [-o *options*] [-p *path*] [-c]

- 常用选项

  -D datadir

  指定数据库配置文件的文件系统位置。如果这个选项被忽略，将使用环境变量UXDATA。

  -l filename

  追加服务器日志输出到filename。如果该文件不存在，它会被创建。umask被设置成077，这样默认情况下不允许其他用户访问该日志文件。

  -W

  不等待操作完成。

  -t seconds

  指定等待一个操作完成时要等待的最大秒数（见选项-w）。默认为UXCTLTIMEOUT环境变量的值，如果该环境变量没有设置则默认为60秒。

  -o options

  指定被直接传递给uxdb命令的选项。-o可以被指定多次，所有给定的选项都会被传递。

  -p path

  指定uxdb可执行程序的位置。默认情况下，uxdb可执行程序可以从ux_ctl相同的目录得到，或者如果没有在那里找到，则在硬写的安装目录中获得。

  在init模式中，这个选项类似于指定了initdb可执行程序的位置。

  详细请参见《优炫数据库 2.1 工具使用手册》ux_ctl部分。

- 示例

  start模式启动一个新的服务器。

  $ ux_ctl start -D /opt/uxdbinstall/dbsql/data -l logfile

  当服务器启动后运行时，其PID将被保存在数据目录中的uxmaster.pid文件，防止多个服务器实例运行在同一个数据目录中。

3.1.2. 使用uxdb

- 用法

  uxdb [option...]

- 常用选项

  -B NBUFFERS

  共享缓冲区的数量。

  -c NAME=VALUE

  设置运行时间参数。

  -C NAME

  打印运行时参数的值。

  -d 1-5

  调试级别。

  -D DATADIR

  数据库目录。

  -e

  使用欧洲日期输入格式（DMY）。

  -F

  关闭fsync。

  -h HOSTNAME

  主机名或IP地址进行监听。

  -i

  启用TCP/IP连接。

  -k DIRECTORY

  Unix域套接字位置。

  -l

  启用SSL连接。

  -N MAX-CONNECT

  允许的最大连接数

详细请参见《优炫数据库 2.1 工具使用手册》uxdb部分。

- 示例

  - 示例一

    通过-D选项来指定数据库配置文件在系统中的位置（/opt/uxdbinstall/dbsql/data），启动uxdb。

    $ **uxdb -D /opt/uxdbinstall/dbsql/data**

    > **注意**
    >
    > 1. 上述命令必须以数据库用户登录后进行操作。
    > 2. 如果没有-D选项，服务器将尝试使用环境变量UXDBDATA命名的目录。如果环境变量未被提供，则此启动服务器操作将失败。

  - 示例二

    使用Unix shell语法，在后台启动uxdb。

    $ **uxdb -D /opt/uxdbinstall/dbsql/data >logfile 2>&1 &**

    该命令会将服务器的stdout和stderr输出存储到某个地方，这将有益于审计和诊断问题。

3.2. 关闭

3.2.1. 使用ux_ctl

ux_ctl的stop模式关闭运行在指定数据目录中的服务器。

- 用法

  ux_ctl stop [-D *datadir*] [-m s[mart] | f[ast] | i[mmediate]] [-W] [-t *seconds*] [-s]

- 常用选项

  -m

  指定关闭模式，如下所示。

  1. “smart”模式：将等待所有客户端断开连接，并等待所有在线备份结束。
  2. “fast”模式（默认）：不会等待客户端断开连接，并且将终止进行中的在线备份。所有活动事务都被回滚，并且客户端被强制断开连接，然后服务器被关闭。
  3. “immediate”模式：将立刻中止所有服务器进程。该模式将导致下一次重启时，数据库进行一次从崩溃中的恢复。

  -D datadir

  指定数据库配置文件的文件系统位置。如果这个选项被忽略，将使用环境变量UXDATA。

  详细请参见《优炫数据库 2.1 工具使用手册》ux_ctl部分。

- 示例

使用ux_ctl关闭运行在/opt/uxdbinstall/dbsql/data中的服务器。

$ ux_ctl stop -D /opt/uxdbinstall/dbsql/data

3.2.2. 使用kill

在非 Windows系统上可以直接用kill发送关闭信号。可用ps程序或从数据目录的uxmaster.pid文件中找到uxdb进程的PID，执行快速关闭。假如数据目录为/opt/uxdbinstall/dbsql/data，执行如下命令。

$ kill -INT `head -1 /opt/uxdbinstall/dbsql/data/uxmaster.pid`

第 4 章 管理数据库

每个正在运行的UXDB服务器实例都管理着一个或多个数据库。因此，在组织SQL对象（“数据库对象”）的层次中，数据库位于最顶层。本章描述数据库的属性，以及如何创建、管理、删除它们。

4.1. 概述

一个数据库是一些SQL对象（“数据库对象”）的命名集合。通常每个数据库对象（表、函数等）属于并且只属于一个数据库（不过有几个系统表如ux_database属于整个集簇并且对集簇中的每个数据库都是可访问的）。更准确地说，一个数据库是一个模式的集合， 而模式包含表、函数等等。因此完整的层次是这样的：服务器、数据库、模式、表（或者某些其他对象类型，如函数）。

当连接到数据库服务器时，客户端必须在它的连接请求中指定它要连接的数据库名。每次连接不能访问超过一个数据库。不过，一个应用能够在同一个或者其他数据库上打开的连接数并没有受到限制。数据库是物理上相互隔离的，并且访问控制是在连接层面进行管理的。如果一个UXDB服务器实例用于承载那些应该分隔并且相互之间并不知晓的用户和项目，那么我们建议把它们放在不同的数据库里。如果项目或者用户是相互关联的， 并且可以相互使用对方的资源，那么应该把它们放在同一个数据库里，但可能在不同的模式中。模式只是一个纯粹的逻辑结构并且谁能访问某个模式由权限系统管理。

数据库是使用CREATE DATABASE（见第 4.2 节 “创建一个数据库”，并且用DROP DATABASE命令删除（见第 4.5 节 “销毁一个数据库”。要确定现有数据库的集合，可以检查系统目录ux_database，如下所示。

```
SELECT datname FROM ux_database;
```

uxsql程序的\l元命令和-l命令行选项也可以用来列出已有的数据库。

> **注意**
>
> SQL标准把数据库称作“目录”，不过实际上没有区别。

4.2. 创建一个数据库

为了创建一个数据库，UXDB服务器必须启动并运行（请参见《优炫数据库 V2.1 用户手册》“服务器设置和操作”章节）。

数据库用SQL命令CREATE DATABASE创建。

```
CREATE DATABASE name;
```

其中*name*遵循SQL标识符的一般规则。当前角色自动成为该新数据库的拥有者。以后删除这个数据库也是该拥有者的特权（同时还会删除其中的所有对象，即使那些对象有不同的拥有者）。

创建数据库是一个受限的操作。如何授权请参见《优炫数据库 V2.1 用户手册》“数据库角色”章节。

因为你需要连接到数据库服务器来执行CREATE DATABASE命令，那么还有一个问题是任意给定站点的第一个数据库是怎样创建的？第一个数据库总是由initdb命令在初始化数据存储区域时创建的（请参见《优炫数据库 V2.1 用户手册》“服务器设置和操作”章节）。这个数据库被称为uxdb。因此要创建第一个“普通”数据库时，你可以连接到uxdb。

在数据库集簇初始化期间也会创建第二个数据库template1，。当在集簇中创建一个新数据库时，实际上就是克隆了template1。这就意味着你对template1所做的任 何修改都会体现在所有随后创建的数据库中。因此应避免在template1中创建对象，除非你想把它们传播到每一个新创建的数据库中。详见第 4.3 节 “模板数据库”。

为了方便，你还可以用一个程序来创建新数据库：createdb。

createdb *dbname*

它连接到uxdb数据库并且发出CREATE DATABASE命令，和前面介绍的完全一样。注意不带任何参数的createdb将创建一个使用当前用户名的数据库。

> **注意**
>
> 《优炫数据库 V2.1 用户手册》“客户端认证”章节包含有关如何限制谁能连接到一个给定数据库的信息。

有时候你想为其他人创建一个数据库，并且使其成为新数据库的拥有者，这样他们就可以自己配置和管理这个数据库。要实现这个目标，使用下列命令之一。

- 用于 SQL 环境

  CREATE DATABASE *dbname* OWNER *rolename*;

- 用于 shell

  createdb -O *rolename dbname*

只有超级用户才被允许为其他人（即为一个你不是其成员的角色）创建一个数据库。

4.3. 模板数据库

CREATE DATABASE实际上通过拷贝一个已有数据库进行工作。默认情况下，它拷贝名为template1的标准系统数据库。所以该数据库是创建新数据库的“模板”。 如果你为template1数据库增加对象，这些对象将被拷贝到后续创建的用户数据库中。 这种行为允许对数据库中标准对象集合的站点本地修改。例如，如果你把过程语言PL/Perl安装到 template1中，那么你在创建用户数据库后不需要额外的操作就可以使用该语言。

系统里还有名为template0的第二个标准系统数据库。这个数据库包含和template1初始内容一样的数据，也就是说，只包含你的UXDB版本预定义的标准对象。在数据库集簇被初始化之后，不应该对template0做任何修改。通过指示CREATE DATABASE使用template0取代template1进行拷贝，你可以创建一个“纯净的”用户数据库，它不会包含任何template1中的站点本地附加物。 这一点在恢复一个ux_dump转储时非常方便：转储脚本应该在一个纯净的数据库中恢复以确保我们重建被转储数据库的正确内容，而不和任何现在可能已经被加入到template1中的附加物相冲突。

另一个从template0而不是template1复制的常见原因是，可以在复制template0时指定新的编码和区域设置，而一个template1的副本必须使用和它相同的设置。这是因为的template1可能包含编码相关或区域相关的数据，而template0中没有。

要通过拷贝template0来创建一个数据库，使用以下命令之一。

- 用于 SQL 环境

  CREATE DATABASE *dbname* TEMPLATE template0;

- 用于 shell

  createdb -T template0 *dbname*

可以创建额外的模板数据库，并且实际上可以通过将集簇中任意数据库指定为CREATE DATABASE的模板来从该数据库拷贝。不过，我们必需明白，这个功能并不是设计作为一般性的“COPY DATABASE”功能。主要的限制是当源数据库被拷贝时，不能有其他会话连接到它。如果在CREATE DATABASE开始时存在任何其它连接，那么该命令将会失败。在拷贝操作期间，到源数据库的新连接将被阻止。

对于每一个数据库在ux_database中存在两个有用的标志：datistemplate和datallowconn列。datistemplate可以被设置来指示该数据库是不是要作为CREATE DATABASE的模板。如果设置了这个标志，那么该数据库可以被任何有 CREATEDB权限的用户克隆；如果没有被设置，那么只有超级用户和该数据库的拥有者可以克隆它。如果datallowconn为假，那么将不允许与该数据库建立任何新的连接（但已有的会话不会因为把该标志设置为假而被中止）。template0通常被标记为datallowconn = false来阻止对它的修改。template0和template1通常总是被标记为datistemplate = true。

> **注意**
>
> 除了template1是CREATE DATABASE的默认源数据库名之外，template1和template0没有任何特殊的状态。例如，我们可以删除template1然后从template0重新创建它而不会有任何不良效果。如果我们不小心在template1中增加了一堆垃圾，那么我们就会建议做这样的操作（要删除template1，它必须有ux_database.datistemplate = false）。
>
> 当数据库集簇被初始化时，也会创建uxdb数据库。这个数据库用于做为用户和应用连接的默认数据库。它只是template1的一个拷贝，需要时可以删除并重建。

4.4. 数据库配置

UXDB服务器提供了大量的运行时配置变量。你可以为其中的许多设置数据库相关的默认值。

例如，如果由于某种原因，你想禁用指定数据库上的GEQO优化器，正常情况下你不得不对所有数据库禁用它，或者确保每个连接的客户端小心地发出了SET geqo TO off。要令这个设置在一个特定数据库中成为默认值，执行如下命令。

ALTER DATABASE mydb SET geqo TO off;

这样将保存该设置（但不是立即设置它）。在后续建立的到该数据库的连接中它将表现得像在会话开始后马上调用SET geqo TO off;。注意用户仍然可以在该会话中更改这个设置，它只是默认值。要撤消这样的设置，使用ALTER DATABASE *dbname* RESET *varname*。

4.5. 销毁一个数据库

数据库用命令删除。

DROP DATABASE *name*;

只有数据库的拥有者或者超级用户才可以删除数据库。删除数据库会移除其中包括的所有对象。数据库的删除不能被撤销。

你不能在与目标数据库连接时执行DROP DATABASE命令。不过，你可以连接到任何其它数据库，包括template1数据库。template1也是你删除一个给定集簇中最后一个用户数据库的唯一选项。

为了方便，有一个在 shell 程序可以删除数据库，如下所示。

dropdb *dbname*

（和createdb不同，删除当前用户名的数据库不是默认动作）。

4.6. 表空间

UXDB中的表空间允许数据库管理员在文件系统中定义用来存放表示数据库对象的文件的位置。一旦被创建，表空间就可以在创建数据库对象时通过名称引用。

通过使用表空间，管理员可以控制一个UXDB安装的磁盘布局。 这么做至少有两个用处。首先，如果初始化集簇所在的分区或者卷用光了空间，而又不能在逻辑上扩展或者做别的什么操作，那么表空间可以被创建在一个不同的分区上，直到系统可以被重新配置。

其次，表空间允许管理员根据数据库对象的使用模式来优化性能。例如，一个很频繁使用的索引可以被放在非常快并且非常可靠的磁盘上，如一种非常贵的固态设备。同时，一个很少使用的或者对性能要求不高的存储归档数据的表可以存储在一个便宜但比较慢的磁盘系统上。

> **警告**
>
> 即便是位于主要的UXDB数据目录之外，表空间也是数据库集簇的一部分并且不能被视作数据文件的一个自治集合。它们依赖于包含在主数据目录中的元数据，并且因此不能被附加到一个不同的数据库集簇或者单独备份。类似地，如果丢失一个表空间（文件删除、磁盘失效等），数据库集簇可能会变成不可读或者无法启动。把一个表空间放在一个临时文件系统（如一个内存虚拟盘）上会带来整个集簇的可靠性风险。

要定义一个表空间，使用CREATE TABLESPACE命令，如下所示。

CREATE TABLESPACE fastspace LOCATION '/ssd1/uxsino/data';

这个位置必须是一个已有的空目录，并且属于UXDB操作系统用户。所有后续在该表空间中创建的对象都将被存放在这个目录下的文件中。该位置不能放在可移动或者瞬时存储上，因为如果表空间丢失会导致集簇无法工作。

> **注意**
>
> 通常在每个逻辑文件系统上创建多于一个表空间没有什么意义，因为你无法控制在一个逻辑文件系统中特定文件的位置。不过，UXDB不强制任何这样的限制，并且事实上它不会注意你的系统上的文件系统边界。它只是在你告诉它要使用的目录中存储文件。

表空间的创建本身必须作为一个数据库超级用户完成，但在创建完之后之后你可以允许普通数据库用户来使用它。要这样做，给数据库普通用户授予表空间上的CREATE权限。

表、索引和整个数据库都可以被分配到特定的表空间。想这么做，在给定表空间上有CREATE权限的用户必须把表空间的名字以一个参数的形式传递给相关的命令。例如，在表空间space1中创建一个表，如下所示。

```
CREATE TABLE foo(i int) TABLESPACE space1;
```

另外，还可以使用default_tablespace参数，如下所示。

```
SET default_tablespace = space1;
CREATE TABLE foo(i int);
```

当default_tablespace被设置为非空字符串，那么它就为没有显式TABLESPACE子句的CREATE TABLE和CREATE INDEX命令提供一个隐式TABLESPACE子句。

还有一个temp_tablespaces参数，它决定临时表和索引的位置，以及用于大数据集排序等目的的临时文件的位置。这可以是一个表空间名的列表，而不是只有一个。因此，与临时对象有关的负载可以散布在多个表空间上。每次要创建一个临时对象时，将从列表中随机取一个成员来存放它。

与一个数据库相关联的表空间用来存储该数据库的系统目录。此外，如果没有给出TABLESPACE子句并且没有在default_tablespace或temp_tablespaces（如适用）中指定其他选择，它还是在该数据库中创建的表、索引和临时文件的默认表空间。如果一个数据库被创建时没有指定表空间，它会使用其模板数据库相同的表空间。

当初始化数据库集簇时，会自动创建两个表空间。ux_global表空间被用于共享系统目录。ux_default表空间是template1和template0数据库的默认表空间（并且，因此也将是所有其他数据库的默认表空间，除非被一个CREATE DATABASE中的TABLESPACE子句覆盖）。

表空间一旦被创建，就可以被任何数据库使用，前提是请求的用户具有足够的权限。这也意味着，一个表空间只有在所有使用它的数据库中所有对象都被删除掉之后才可以被删掉。

要删除一个空的表空间，使用DROP TABLESPACE命令。

要确定现有表空间的集合，可检查ux_tablespace系统目录，如下所示。

```
SELECT spcname FROM ux_tablespace;
```

uxsql程序的\db元命令也可以用来列出现有的表空间。

UXDB使用符号连接来简化表空间的实现。这就意味着表空间只能在支持符号连接的系统上使用。

$UXDATA/ux_tblspc目录包含指向集簇中定义的每个非内建表空间的符号连接。　尽管我们不推荐，但还是可以通过手工重定义这些连接来调整表空间布局。在服务器运行时，绝不要这样做。

第 5 章 用户和权限

5.1. 用户

用户是一个数据对象，是一系列数据库对象和权限的统称。用户所有的操作默认在可使用的模式下进行，模式是一个用户所拥有的数据库对象的集合，每个用户都有自己的默认模式，管理用户默认模式的名字与用户名相同，普通用户默认模式为public。UXDB数据库为了防止权限滥用问题，引入了三权分立的安全机制，只有安全功能开关开启时，才会具备三权分立。默认提供了系统管理员（uxsmo）、安全管理员（uxsso）、审计管理员（uxsao），将DBA的权限拆解出来，按最小授权原则分别授予它们各自为完成自己承担任务所需的最小权限，并形成相互制约的关系。

5.2. 权限

数据库用户能够创建自己的数据库对象（例如表），并可以把对这些数据库对象的操纵权限授予其他用户。

当创建一个数据库用户时，可以同时为该用户创建一个对应的模式（模式为数据库对象的集合，如表、视图、过程和函数等)。当该用户连接数据库时，就可存取该模式中的全部对象。一个用户可以创建多个模式，用户 缺省使用的模式为PUBLIC。

权限是执行一种特殊类型的SQL语句或存取某一用户的对象的权力，包括系统权限、对象权限和列级权限。

- 系统权限

  是执行特定操作的权限。一类是以用户或角色的属性存在的系统权限，这些权限包括：CREATE DATABASE、CREATE USER、CREATE ROLE的权限，具体分为SUPERUSER、CREATEDB和CREATEROLE系统权限；一类是通过GRANT/REVOKE语句来授予和回收的系统权限，如ANY系统权限。

- 对象权限

  是对给定的用户授予在给定对象（例如表）上执行的操作集。这些操作可以指明为INSERT 、SELECT等，具体各类对象具有的权限类型可参见GRANT和REVOKE语句的说明。

- 列级权限

  是对给定的用户授予在给定表或视图上某些列执行操作集。此动作只能为INSERT、UPDATE和REFERENCES。

  当前用户是其所创建的对象的拥有者，对象的拥有者在其上具有所有特权。只有当用户有适当系统权限或对象权限时，才能执行相应操作，否则执行失败，并返回权限不足的错误提示信息。

5.3. 通过图形界面管理用户和权限

5.3.1. 新建用户

通过uxdbadmin数据库对象管理工具可视化图形界面的引导，创建用户。

鼠标定位在导航树上角色或“角色名称”所在节点，右键单击角色或“角色名称”；在弹出式菜单中选择“新建角色”菜单项，新建用户。

5.3.2. “属性”页

属性页，可以定义或查看用户的主要属性。

“属性”包含以下信息。

- 用户名

  待创建的或者当前查看的用户的名称。在“新建角色”操作过程中，可以指定一个有效的数据库标识符作为用户名。用户名不能重复。

- 对象id

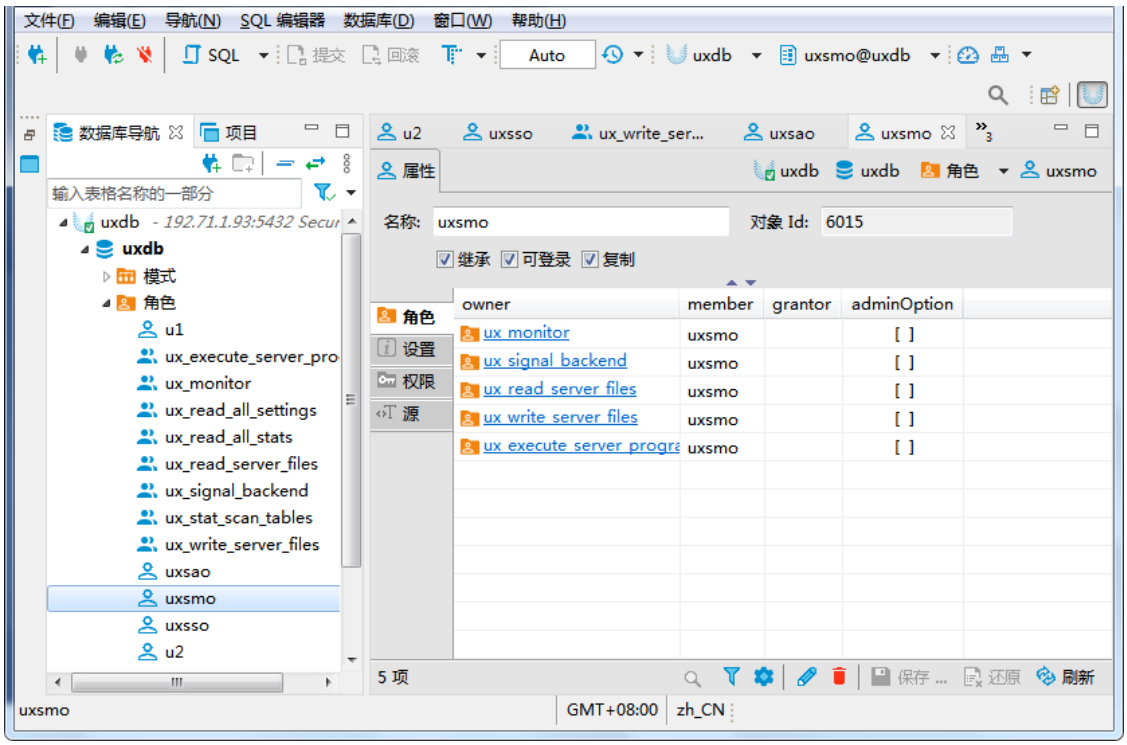
  唯一标识。

- 用户的权限

  继承、可登录、复制、绕过所有的行级别安全策略、超级用户、创建角色、创建数据库。

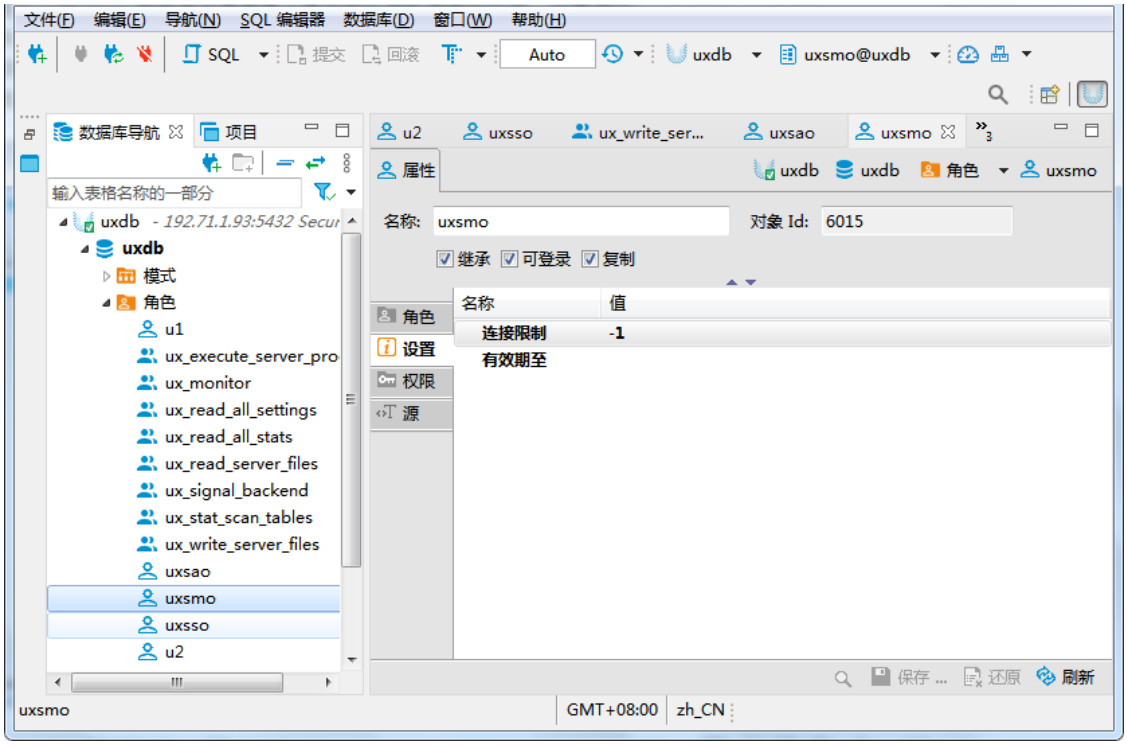
5.3.3. “角色”页

角色页说明当前角色/用户，继承的角色，授予的权限。



5.3.4. “设置”页

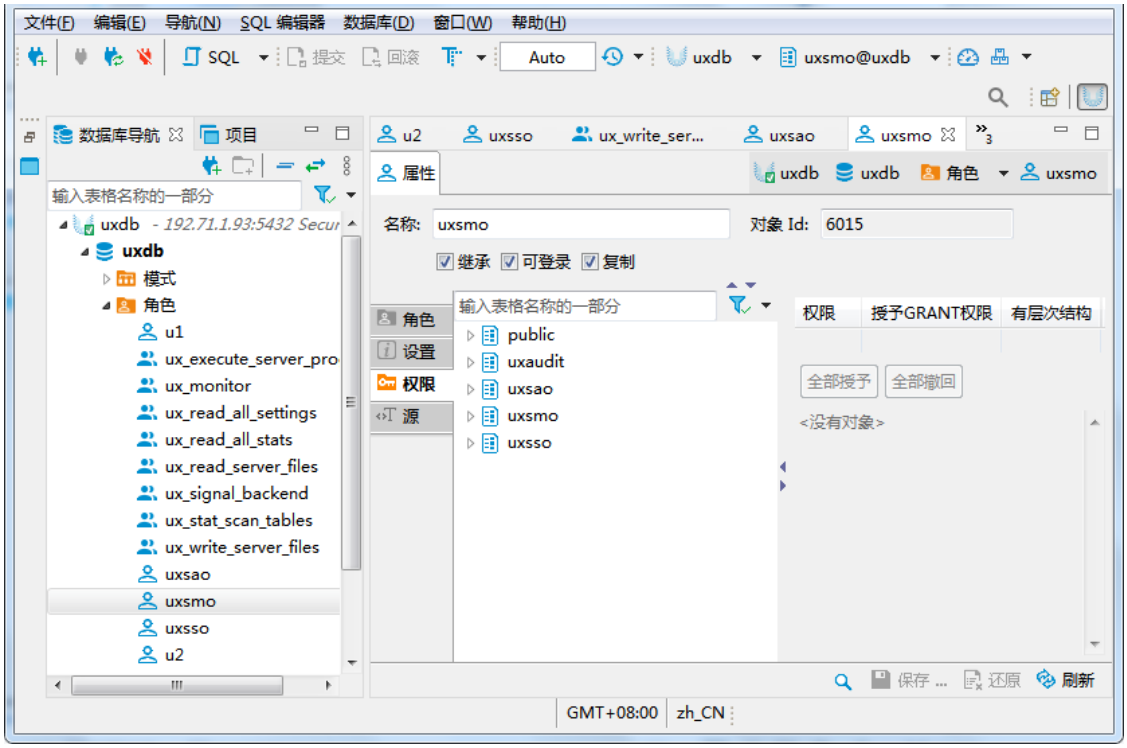
设置页说明连接限制和有效期。连接限制指用户能够建立的最大连接数。有效期当前不允许设置，即用户一直有效。



5.3.5. “权限”页

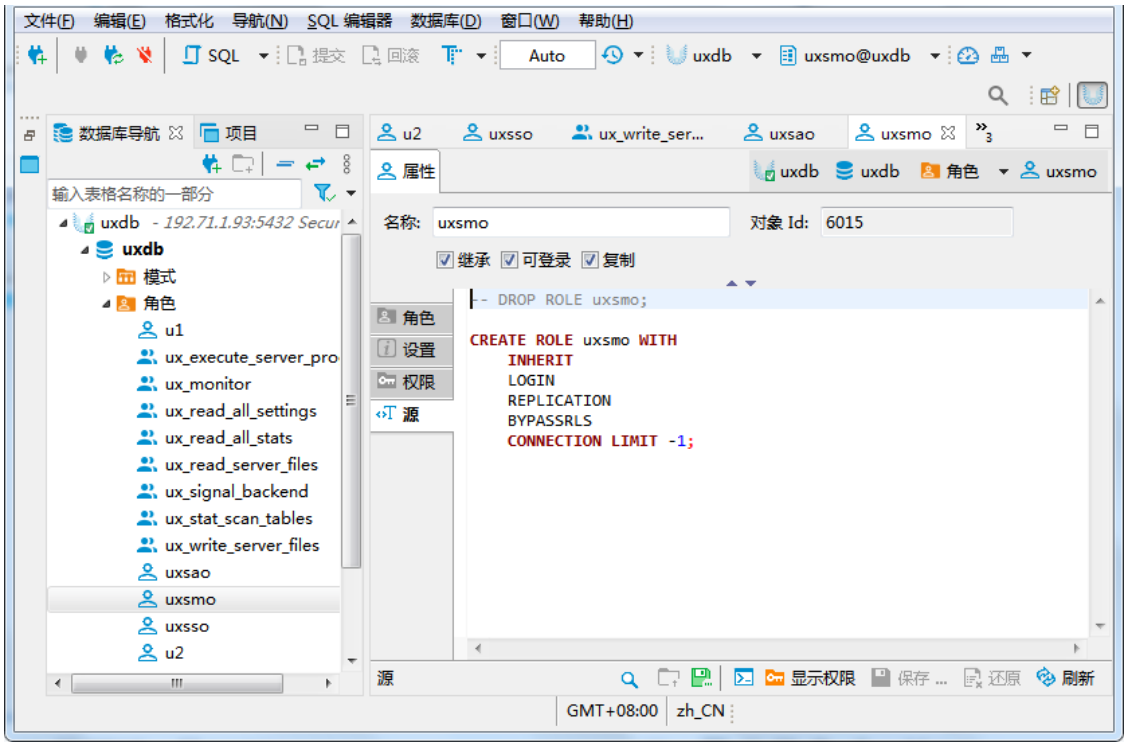
对象权限是对给定的用户授予/撤回在给定对象（例如表）上执行的操作集。这些操作可以指明为INSERT、SELECT等。

对象权限属性页，可以定义或查看数据库对象的权限。



5.3.6. “源”页

通过信息浏览区SQL对话框，用户可以查看对应的SQL语言。



5.3.7. 删除用户

通过uxdbadmin数据库对象管理工具可视化图形界面的引导，删除用户。

鼠标定位在导航树上“角色名称”所在节点，右键单击“角色名称”；在弹出式菜单中选择“删除”菜单项，或按快捷键Delete。

5.4. 通过SQL管理用户和权限

5.4.1. 查看用户账户信息

在uxdb数据库中，可通过查询系统表或系统视图查看用户账户信息。用户账户对应的系统表和视图为ux_authid、ux_shadow和ux_user，如下所示。

```
uxdb=# select * from ux_user;
     username      | usesysid | usecreatedb | usesuper | userepl | usebypassrls |  passwd  | valuntil |
 useconfig
--------------------+----------+-------------+----------+---------+--------------+----------+----------+-----------
 uxdb               |       10 | t           | t        | t       | t            | ******** |          |
 UX_SECURITY_OFFICER |     6019 | f           | f        | f       | f            | ******** |          |
 UX_AUDIT_OFFICER   |     6016 | f           | f        | f       | f            | ******** |          |
 UX_MANAGE_OFFICER  |     6015 | t           | f        | t       | t            | ******** |          |
(4 rows)
```

5.4.2. CREATE USER

- 大纲

  CREATE USER name [[WITH] option [...]]

- 描述

  使用CREATE USER语句创建新的用户。

- 示例

  创建用户名为test1、登录口令为“123456”的用户。

  uxdb=# **create user test1 with password '123456';**

5.4.3. ALTER USER

- 描述

  使用ALTER USER语句修改数据库认证用户的口令。

- 示例

  修改名为test1的用户登录口令为“654321”。

  uxdb=# **alter user test1 with password '654321';**

5.4.4. DROP USER

- 描述

  使用DROP USER语句删除用户。

- 示例

  删除名为test1的用户。

  uxdb=# **drop user test1;**

5.4.5. 权限

对象在创建时会被分配一个所有者，所有者通常是执行创建语句的角色。对于大部分类型的对象，初始状态下只有所有者（或者数据库管理员）能够对该对象做任何事情。为了允许其他角色使用它，必须分配权限。

数据库存在多种不同的权限：
SELECT、INSERT、UPDATE、DELETE、TRUNCATE、REFERENCES、TRIGGER、CREATE、CONNECT、TEMPORARY、EXECUTE以及USAGE。可以应用于一个特定对象的权限随着对象的类型（表、函数等）不同而不同。

下面将简单介绍如何使用这些权限。

> **注意**
>
> 通常只有对象的所有者才可以修改或删除一个对象的权限。

1. 一个对象可以通过该对象类型相应的ALTER命令来重新分配所有者。

   重新分配对象，示例如下所示。

   ALTER TABLE table_name
   OWNER TO new_owner;

   数据库管理员总是可以做到这点，对于普通用户而言，当且仅当该用户同时是对象的当前所有者时才能做同样的事。

2. 可使用GRANT命令分配权限。

   示例如下所示。

   如果joe是一个已有角色，而accounts是一个已有表，更新该表的权限可以按如下方式授权。

   GRANT UPDATE ON accounts TO joe;

   用ALL取代特定权限会把与对象类型相关的所有权限全部授权。

   一个特殊的名为PUBLIC的"角色"可以用来向系统中的每一个角色授予一个权限。同时，在数据库中有很多用户时可以设置"组"角色来帮助管理权限。详见数据库角色。

3. 使用REVOKE命令，撤销权限。

   示例如下所示。

   撤销表accounts的权限。

   REVOKE ALL ON accounts FROM PUBLIC;

   对象拥有者的特殊权限（即执行DROP、GRANT、REVOKE等的权力）总是隐式地属于拥有者，并且不能被授予或撤销。但是对象拥有者可以选择撤销他们自己的普通权限。

   一般情况下，只有对象拥有者（或者超级用户）可以授予或撤销一个对象上的权限。但是可以在授予权限时使用"with grant option"来允许接收人将权限转授给其他人。如果授予被撤销，则所有从接收人那里获得的权限（直接或者通过授权链获得）都将被撤销。

   更多信息请见GRANT和REVOKE参考页。

   示例如下所示。

   新建用户liming和liming_rw，创建表mytable，并将表mytable分配给用户liming。

   create user miriam with password '123456';
   create user miriam_rw with password '123456';
   create table mytable(col1 text, col2 integer);
   alter table mytable owner to miriam;

   \dp命令显示表mytable的存取权限。

   uxdb=# **\dp mytable**

```
                        Access privileges
 Schema |  Name   | Type  | Access privileges | Column privileges | Policies
--------+---------+-------+-------------------+-------------------+----------
 public | mytable | table |                   |                   |
(1 row)
```

重新授予权限。

```
grant select on mytable to public;
grant select, update, insert on mytable to uxdb;
grant select (col1), update (col1) on mytable to liming;
```

显示。

```
uxdb=# \dp mytable
                           Access privileges
 Schema |  Name   | Type |   Access privileges   | Column privileges  | Policies
--------+---------+-------+----------------------+--------------------+----------
 public | mytable | table | liming=arwdDxt/liming+| col1:              +|
        |         |       | =r/liming            +|   liming=rw/liming |
        |         |       | uxdb=arw/liming       |                    |
(1 row)
```

5.4.6. createuser和dropuser

createuser创建一个新的uxdb数据库用户。

createuser [connection-option...] [option...] [username]

connection-option选项如下所示。

-h host

指定运行服务器的机器的主机名。

-p port

指定服务器正在监听连接的TCP端口或本地Unix域套接字文件扩展。

-U username

指定连接的用户。

option选项如下所示。

-d

新用户将被允许创建数据库。

详细参数见createuser。

> **注意**
>
> 1. 只有数据库管理员和具有CREATEROLE特权的用户才能创建新用户。

2. 如果希望创建一个新的数据库管理员用户，必须作为一个数据库管理员连接，而不仅仅只具有CREATEROLE特权。数据库管理员拥有绕过数据库中所有访问权限检查的能力，因此数据库管理员权限不能轻易被授予。

dropuser删除一个已有的uxdb用户。

dropuser [connection-option...] [option...] [username]

connection-option选项如下所示。

-h host

指定运行服务器的机器的主机名。

-p port

指定服务器正在监听连接的TCP端口或本地Unix域套接字文件扩展。

-U username

指定连接的用户。

option选项如下所示。

-e

显示dropuser生成并发送给服务器的命令。

详细参数见dropuser。

注意

只有数据库管理员以及具有CREATEROLE特权的用户能够删除uxdb数据库用户。

第 6 章 数据库角色

UXDB使用角色的概念管理数据库访问权限。一个角色可以被看成是一个数据库用户或者是一个数据库用户组，这取决于角色被怎样设置。角色可以拥有数据库对象（例如，表和函数）并且能够把那些对象上的权限赋予给其他角色来控制谁能访问哪些对象。此外，还可以把一个角色中的成员资格授予给另一个角色，这样允许成员角色使用被赋予给另一个角色的权限。

角色的概念把“用户”和“组”的概念都包括在内。任意角色都可以扮演用户、组或者两者。

本章描述如何创建和管理角色。

6.1. 数据库角色

数据库角色从概念上与操作系统用户是完全无关的。在实际使用中把它们对应起来可能比较方便，但这不是必须的。数据库角色在整个数据库集群中是全局的（而不是独立数据库内）。要创建一个角色，可使用CREATE ROLE，如下所示。

CREATE ROLE *name*;

*name*遵循SQL标识符的规则：要么完全没有特殊字符，要么用双引号包围（实际上通常会给命令增加额外的选项，例如LOGIN。更多细节可见下文）。要移除一个已有的角色，使用DROP ROLE命令，如下所示。

DROP ROLE *name*;

为了方便，createuser和dropuser程序被提供作为这些SQL命令的封装，可以从shell命令行调用。

createuser *name*
dropuser *name*

要检查现有角色的集合，检查ux_roles系统表，如下所示。

SELECT rolname FROM ux_roles;

uxsql程序的\du元命令也可以用来列出现有角色。

为了引导数据库系统，一个刚刚被初始化好的系统总是包含一个预定义角色。这个角色总是一个“superuser”，并且默认情况下（除非在运行initdb时修改）它的名称和初始化数据库集群的操作系统用户名相同。习惯上，这个角色将被命名为uxdb。为了创建更多角色，你首先必须以初始角色的身份连接。

每一个和数据库的连接都必须用一个角色身份进行，这个角色决定在该连接上的初始权限。要使用一个特定数据库连接的角色名由客户端指示，该客户端以一种应用相关的风格发起连接请求。例如，uxsql程序使用-U命令行选项来指定要以哪个角色连接。很多应用以当前操作系统的用户名为默认角色名（包括createuser和uxsql）。因此在角色和操作系统用户之间维护一个名称对应关系通常是很方便的。

一个给定客户端连接能够用来连接的数据库角色的集合由该客户端的认证设置决定。因为角色身份决定一个已连接客户端可用的权限集合，在设置一个多用户环境时要小心地配置权限。

6.2. 角色属性

一个数据库角色可以有一些属性，它们定义角色的权限并且与客户端认证系统交互。

login privilege

只有具有LOGIN属性的角色才能被用于一个数据库连接的初始角色名称。一个带有LOGIN属性的角色可以被认为和一个“数据库用户”相同。要创建一个带有登录权限的角色，使用下列两个命令之一。

CREATE ROLE *name* LOGIN;
CREATE USER *name*;

（除了CREATE USER默认假定有LOGIN之外，CREATE USER和CREATE ROLE等效）。

superuser status

一个数据库超级用户会绕开所有权限检查，除了登录的权限。这是一个危险的权限并且应该小心使用，最好用一个不是超级用户的角色来完成你的大部分工作。要创建一个新数据库超级用户，使用CREATE ROLE *name* SUPERUSER。你必须作为一个超级用户来完成这些。

database creation

一个角色必须被显式给予权限才能创建数据库（除了超级用户，因为它们会绕开所有权限检查）。要创建这样一个角色，使用CREATE ROLE *name* CREATEDB。

role creation

一个角色必须被显式给予权限才能创建更多角色（除了超级用户，因为它们会绕开所有权限检查）。要创建这样一个角色，使用CREATE ROLE *name* CREATEROLE。一个带有CREATEROLE权限的角色也可以修改和删除其他角色，还可以授予或回收角色中的成员关系。然而，要创建、修改、删除一个超级用户角色的成员关系，需要以超级用户的身份操作，CREATEROLE不足以完成这一切。

initiating replication

一个角色必须被显式给予权限才能发起流复制（除了超级用户，因为它们会绕开所有权限检查）。一个被用于流复制的角色必须也具有LOGIN权限。要创建这样一个角色，使用CREATE ROLE *name* REPLICATION LOGIN。

password

只有当客户端认证方法要求用户在连接数据库时提供一个口令时，口令才有意义。password和md5认证方法使用口令。数据库口令与操作系统命令独立。在角色创建时指定一个口令：CREATE ROLE *name* PASSWORD '*string*'。

在创建后可以用ALTER ROLE修改一个角色属性。

> **提示**
>
> 推荐创建一个具有CREATEDB和CREATEROLE权限的角色，而不是创建一个超级用户，并且然后用这个角色来完成对数据库和角色的例行管理。这种方法避免了在非必要时作为超级用户操作任务的风险。

运行时配置设置，一个角色也可以针对自身为运行时的配置设置默认项。例如，如果出于某些原因你希望在每次连接时禁用索引扫描（不建议这样操作），可以使用如下命令。

ALTER ROLE myname SET enable_indexscan TO off;

这将保存设置（但是不会立刻设置）。等同于在这个角色的后续连接中，会话开始之前执行SET enable_indexscan TO off。你也可以在会话期间改变该设置，它将只是作为默认值。要移除一个角色相关的默认设置，使用ALTER ROLE *rolename* RESET *varname*。注意附加到没有LOGIN权限的角色的角色相关默认值没有意义，因为它们从不会被调用。

6.3. 角色成员关系

把用户分组在一起来便于管理权限常常很方便：那样，权限可以被授予一整个组或从一整个组回收。在UXDB中通过创建一个表示组的角色来实现，并且然后将在该组角色中的成员关系授予给单独的用户角色。

要建立一个组角色，首先创建该角色，如下所示。

CREATE ROLE *name*;

通常被用作一个组的角色不需要有LOGIN属性，也可以进行设置。

一旦组角色存在，你可以使用GRANT和REVOKE命令增加和移除成员，如下所示。

GRANT *group_role* TO *role1*, ... ;
REVOKE *group_role* FROM *role1*, ... ;

你也可以为其他组角色授予成员关系（因为组角色和非组角色之间其实没有任何区别）。唯一的制约是不能建立循环的成员关系。另外，不允许把一个角色中的成员关系授予给PUBLIC。

组角色的成员可以以两种方式使用角色的权限。第一，一个组的每一个成员可以显式地做SET ROLE来临时“称为”组角色。在这种状态中，数据库会话可以访问组角色而不是原始登录角色的权限，并且任何被创建的数据库对象被认为属于组角色而不是登录角色。第二，有INHERIT属性的成员自动地具有它们所属角色的权限，包括任何组角色继承得到的权限。作为一个例子，假设有如下角色。

CREATE ROLE joe LOGIN INHERIT;
CREATE ROLE admin NOINHERIT;
CREATE ROLE wheel NOINHERIT;
GRANT admin TO joe;
GRANT wheel TO admin;

在作为角色joe连接后，一个数据库会话将立即拥有直接授予给joe的权限，外加任何授予给admin的权限，因为joe“继承了”admin的权限。然而，授予给wheel的权限不可用，因为即使joe是wheel的一个间接成员，但是该成员关系是通过带NOINHERIT属性的admin得到的。如下所示。

SET ROLE admin;

之后，该会话将只拥有授予给admin的权限，但是没有授予给joe的权限。执行如下命令。

SET ROLE wheel;

之后，该会话将只拥有授予给wheel的权限，但是没有授予给joe或admin的权限。初始的权限状态可以使用下面命令之一恢复。

SET ROLE joe;
SET ROLE NONE;
RESET ROLE;

> **注意**
>
> SET ROLE命令总是允许选择原始登录角色的直接或间接组角色。因此，在上面的例子中，在成为wheel之前不必先成为admin。

> **注意**
>
> 在SQL标准中，用户和角色之间的区别很清楚，并且用户不会自动继承权限而角色会继承。这种行为在UXDB中也可以实现：为要用作SQL角色的角色给予INHERIT属性，而为要用作SQL用户的角色给予NOINHERIT属性。不过，UXDB默认给所有的角色INHERIT属性。

角色属性LOGIN、SUPERUSER、CREATEDB和CREATEROLE可以被认为是一种特殊权限，但是它们从来不会像数据库对象上的普通权限那样被继承。要使用这些属性，你必须实际SET ROLE到一个有这些属性之一的特定角色。继续上述例子，我们可以选择授予CREATEDB和CREATEROLE给admin角色。然后一个以joe角色连接的会话将不会立即有这些权限，只有在执行了SET ROLE admin之后才会拥有。

要销毁一个组角色，使用DROP ROLE，如下所示。

DROP ROLE *name*;

任何在该组角色中的成员关系会被自动撤销（但是成员角色不会受到影响）。

6.4. 删除角色

由于角色可以拥有数据库对象并且能持有访问其他对象的特权，删除一个角色常常并非一次DROP ROLE就能解决。任何被该用户所拥有的对象必须首先被删除或者转移给其他拥有者，并且任何已被授予给该角色的权限必须被收回。

对象的拥有关系可以使用ALTER命令一次转移出去，如下所示。

ALTER TABLE bobs_table OWNER TO alice;

此外，REASSIGN OWNED命令可以被用来把要被删除的角色所拥有的所有对象的拥有关系转移给另一个角色。由于REASSIGN OWNED不能访问其他数据库中的对象，有必要在每一个包含该角色所拥

有对象的数据库中运行该命令（注意第一个这样的REASSIGN OWNED将更改所有在数据库间共享的该角色拥有的对象的拥有关系，即数据库或者表空间）。

一旦任何有价值的对象被转移给新的拥有者，任何由被删除角色拥有的剩余对象就可以用DROP OWNED命令删除。再次，由于这个命令不能访问其他数据库中的对象，有必要在每一个包含该角色所拥有对象的数据库中运行该命令。DROP OWNED不会删除整个数据库或者表空间，因此如果该角色拥有任何还没有被转移给新拥有者的数据库或者表空间，有必要手工删除它们。

DROP OWNED也会注意移除为不属于目标角色的对象授予给目标角色的任何特权。因为REASSIGN OWNED不会触碰这类对象，通常有必要运行REASSIGN OWNED和DROP OWNED（按照描述的先后顺序）以完全地移除要被删除对象的从属物。

总之，移除曾经拥有过对象的角色的方法如下所示。

```
REASSIGN OWNED BY doomed_role TO successor_role;
DROP OWNED BY doomed_role;
-- 在集群中的每一个数据库中重复上述命令
DROP ROLE doomed_role;
```

如果不是所有的拥有对象都被转移给了同一个后继拥有者，最好手工处理异常然后执行上述步骤直到结束。

如果在依赖对象还存在时尝试了DROP ROLE，它将发出消息标识哪些对象需要被重新授予或者删除。

6.5. 默认角色

UXDB提供一组默认角色，他们可以访问特定的的特权功能和信息。管理员可以将这些角色GRANT给用户或其环境中的其他角色，为这些用户提供对指定功能和信息的访问权限。

默认的角色在表 6.1 “默认角色”中描述。请注意，每个默认角色的特定权限可能会因为将来添加额外的功能而发生变化。管理员应监控功能授予记录以进行更改。

表 6.1. 默认角色

角色	允许的权限
ux_read_all_settings	阅读所有配置变量，包含通常只对超级用户可见的配置变量。
ux_read_all_stats	阅读所有ux_stat_*视图并使用各种统计相关的扩展，包含通常只对超级用户可见的扩展。
ux_stat_scan_tables	执行监视功能，该功能可能需要对表进行很长时间的ACCESS SHARE锁定
ux_signal_backend	给其他后端发送信号(比如: 取消查询、终止)。
ux_monitor	读取/执行各种监视视图和函数。 此角色是ux_read_all_settings、 ux_read_all_stats和ux_stat_scan_tables的成员。

ux_monitor、ux_read_all_settings、ux_read_all_stats和ux_stat_scan_tables角色旨在允许管理员轻松配置角色以监视数据库服务器，允许读取通常仅限于超级用户的各种有用的配置设置，统计信息和其他系统信息。

应小心授予这些角色，以确保只在需要执行所需监视的情况下才会使用这些角色。

管理员可以使用GRANT命令给这些用户授予访问权限，如下所示。

GRANT ux_signal_backend TO admin_user;

6.6. 函数和触发器安全性

函数和触发器允许用户在后端服务器中插入代码，其他用户不会注意到这些代码的执行。因此，两种机制允许用户相对容易地为其他人设置“特洛伊木马”。唯一真正的保护是严格控制对能定义函数的用户的控制。

在后端服务器进程中运行的函数带有数据库服务器守护进程的操作系统权限。如果用于函数的编程语言允许非检查的内存访问，它就可能改变服务器的内部数据结构。因此，这些函数可能绕开任何系统访问控制。允许这种访问的函数语言被认为是“不可信的”，并且UXDB只允许超级用户创建这种语言编写的函数。

第 7 章 模式对象管理

7.1. 模式管理

7.1.1. 关于模式

一个数据库可以包含一个或多个命名的模式（SCHEMA），一个模式内可以包含多个表。不同的模式中的表名可以相同，例如，schema1和schema2中可以分别包含名称为mytable的表，但是同一个模式中的表不能同名。模式类似于操作系统级的目录，但模式不能嵌套。

用户只要有权限就可以访问所连接数据库中的任何模式中的对象。

在创建数据库时，UXDB会默认创建两个模式：ux_catalog和public。

ux_catalog模式用于存放系统表和所有内置的数据类型、函数和操作符。

当用户没有自己的模式并且其创建或使用数据库对象没有指定模式时，默认使用public模式。

模式通常在如下情况中使用。

1. 多个用户使用同一个数据库而不会相互影响。
2. 对数据库中的对象进行逻辑分组，更便于管理。
3. 各个应用分别使用各自的模式，以避免命名冲突。

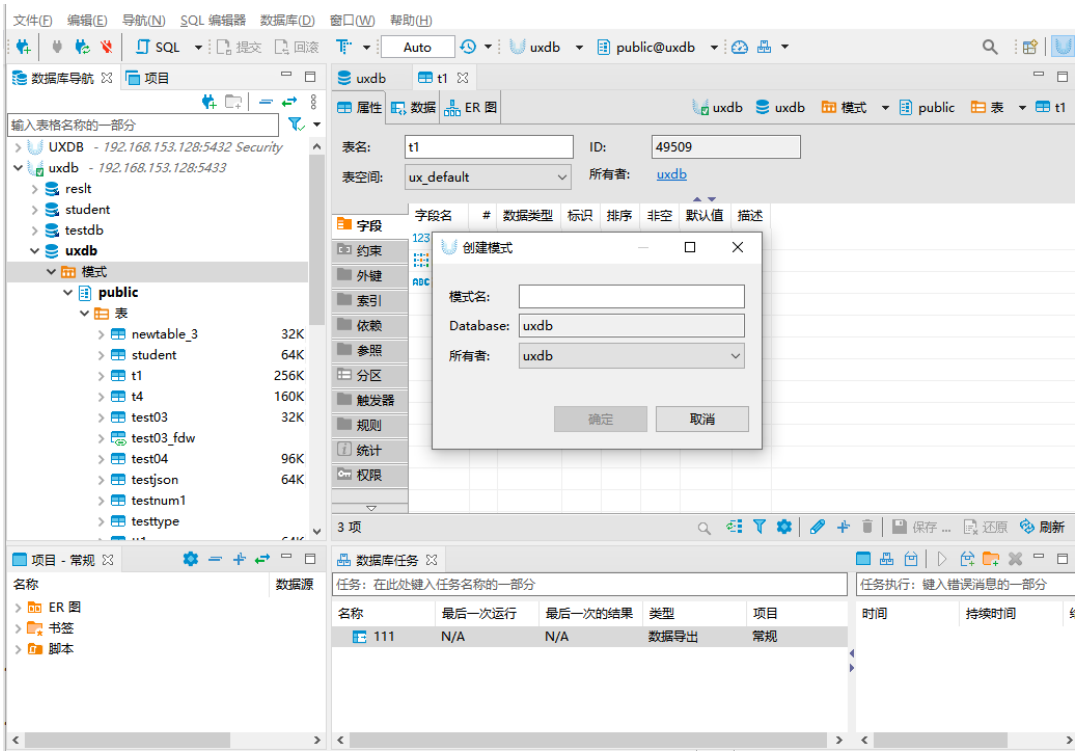
7.1.2. 基于图形界面的模式管理

7.1.2.1. 创建模式

通过UXDBAdmin数据库对象管理工具提供的可视化图形界面的引导，创建模式。如下所示。

1. 鼠标定位在导航树上模式或“模式名称”所在节点，右键单击模式或“模式名称”。
2. 在弹出式菜单中选择“新建模式”菜单项。

3. 数据库对象管理工具右侧信息浏览区将弹出“新建模式”的向导界面所示。



4. 填写图中“模式名称”、“所有者”、“数据库”等信息后，单击确定，即可新建模式。

7.1.2.2. “基本属性”页

“基本属性”页可以查看模式的主要属性，比如模式名称、属主。

“基本属性”页包含如下信息。

表 7.1. 模式管理“基本属性”页参数说明

参数	说明
名称	待创建的或者当前查看的模式的名称。在“创建模式”操作过程中，用户可以指定一个有效的UXDB数据库标识符作为模式名。模式名不能重复。
所有者	该模式的所有者名称，所有者应该是数据库系统的用户之一。若该模式是创建用户时同时创建的，该模式的所有者就是相应用户，其所有关系不能更改。单独创建的模式的所有者默认为当前登录的用户，但是用户可以将模式的所有者指派给其他的用户。 在“更改模式属性”操作过程中，可以将当前模式的所有者指派给其他用户。

7.1.2.3. “权限”页

权限页，可以定义或查看数据库的权限，切换“用户”/“角色”时，权限将随之更新。

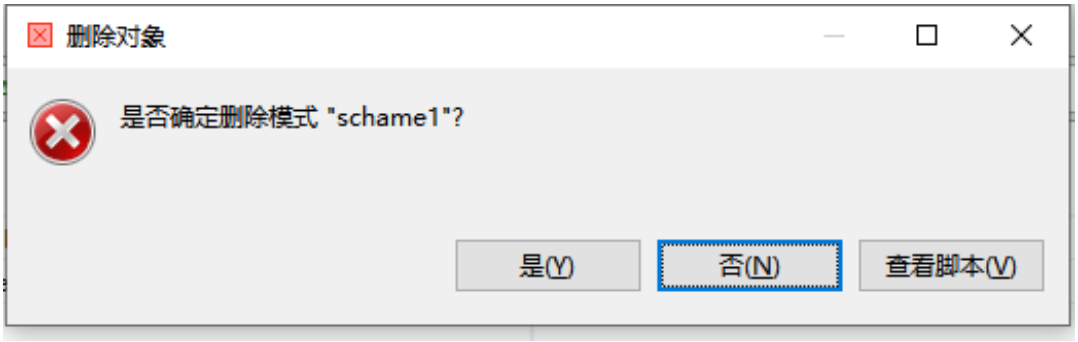
7.1.2.4. “源”页

通过信息浏览区，用户可以查看模式的DDL语言。

7.1.2.5. 删除模式

通过UXDBAdmin数据库对象管理工具可视化图形界面的引导，删除模式。

1. 鼠标定位在导航树上“模式名称”（例如“schame1”）所在节点，右键单击“模式名称”
2. 在弹出式菜单中选择“删除”菜单项，或按快捷键Delete。



7.1.3. 基于SQL语句的模式管理

7.1.3.1. 查看模式

通过检查系统表sys_namespace，可以确定现有模式的集合。

```
uxdb=# select nspname from ux_namespace ;
    nspname
-----------------
 ux_toast
 ux_temp_1
 ux_toast_temp_1
 ux_catalog
 public
 schame1
(6 行记录)
```

uxsql程序的dn命令行选项也可以用来列出已有的模式。

```
uxdb=# \dn
  架构模式列表
  名称  | 拥有者
---------+--------
 public  | uxdb
 schame1 | uxdb
(2 行记录)
```

7.1.3.2. CREATE SCHEMA

- 大纲

  CREATE SCHEMA

- 描述

  CREATE SCHEMA创建一个新模式到当前数据库中。

  > **注意**
  >
  > 1. 要创建一个模式，创建用户必须拥有当前数据库的CREATE特权，或者是数据库管理员。
  > 2. 所创建模式的模式名必须与当前数据库中任何现有模式的名称不同。

  如果希望创建属于UserName的模式SchemaName，可以使用下面的语句。

  CREATE SCHEMA SchemaName AUTHORIZATION UserName;

  使用CREATE SCHEMA语句创建模式，在模式中创建数据库对象或访问模式中的对象时，对象名前使用指定的模式名。

- 示例

  创建一个模式并且在其中创建一个表和视图，如下所示。

  ```
  CREATE SCHEMA hollywood
  CREATE TABLE films (title text, release date, awards text[])
  CREATE VIEW winners AS
  SELECT title, release FROM films WHERE awards IS NOT NULL;
  ```

7.1.3.3. ALTER SCHEMA

- 大纲

  ```
  ALTER SCHEMA SchemaName RENAME TO new_ SchemaName
  ALTER SCHEMA SchemaName OWNER TO { new_owner | CURRENT_USER |
   SESSION_USER }
  ```

- 描述

  ALTER SCHEMA更改一个模式的定义。

  > **注意**
  >
  > 1. 使用ALTER SCHEMA，必须拥有该模式。
  > 2. 重命名一个模式，必须拥有该数据库的CREATE特权。
  > 3. 更改拥有者，必须是新拥有角色的一个直接或者间接成员，并且该角色必须具有该数据库上的CREATE特权。

7.1.3.4. DROP SCHEMA

- 大纲

  ```
  DROP SCHEMA [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]
  ```

- 描述

  DROP SCHEMA从数据库中删除模式。

- 参数

  表 7.2. DROP SCHEMA参数说明

参数	说明
IF EXISTS	如果该模式不存在，将不会抛出一个错误，而是发出一个提示。
name	模式的名称。
CASCADE	自动删除包含在该模式中的对象（表、函数等），然后删除所有依赖于那些对象的对象。
RESTRICT	如果该模式含有任何对象，则拒绝删除，是默认值。

> **注意**
>
> 1. 一个模式只能由其拥有者或一个超级用户删除。
> 2. 即使拥有者不拥有该模式中的某些对象，也能删除该模式（以及所有含有的对象）。

- 示例

  删除一个空模式（模式中的所有数据库对象已经删除），如下所示。

  DROP SCHEMA SchemaName;

  如果模式中还有数据库对象，则使用CASCADE子句，删除模式及其中的所有数据库对象，如下所示。

  DROP SCHEMA SchemaName CASCADE;

7.2. 表管理

7.2.1. 表

表是为数据库中数据存储的基本单位，其数据按行、列存储。

表中列的个数和顺序是固定的。在创建基本表时，要对表中的所有列进行说明，要说明它的名称、数据类型、宽度或精度等。已经说明的列也可以使用SQL语句修改。

列的数据类型限制了该列的可能取值以及说明了该列进行计算时，存储在该列上的数据的语义。例如，一个列声明为INTEGER类型，则其中就不能存放字符串数据，该列的数据可以进行数学运算。同样，一个列声明为字符串类型，则其中可以存放任何数据，但该列的数据不可以进行数学运算。

列的值是可以随时发生变化的。同一列中的所有值具有相同的数据类型，而且这些值必须属于同一个表。列中的一个值是在表中进行检索和更新的最小数据单位。

行是对应单个记录的列信息的集合。同一表中每一行包含表中每一列的一个值，表中每行的第i个值即是该表第i列的一个值。行是在表中进行插入和删除操作的最小数据单位。

表中的数据行是无序的。因此，对表进行查询时，数据行的顺序是随机的，除非查询语句中具有ORDER BY子句。

示例

1. 通过ux_tables，可以查看数据库中每个表的信息。

```
uxdb=# select tablename from ux_tables;
       tablename
-------------------------
 t
 ux_statistic
 ux_statistic_ext_data
 ux_user_mapping
 ux_type
 ux_authid
 .
 .
 .
```

2. 通过uxsql程序的dt命令行选项查看表。

```
uxdb=# \dt
       List of relations
 Schema |  Name   | Type  | Owner
--------+---------+-------+-------
 public | student | table | uxdb
 public | t       | table | uxdb
 public | test    | table | uxdb
(3 rows)
```

3. 通过uxsql程序的d命令行选项查看某个表的结构。

   示例：查看表ux_type的结构(加 ″+″ 获取更多信息：d+)。

```
uxdb=# \d+ ux_type
                                Table "ux_catalog.ux_type"
    Column     |     Type     | Collation | Nullable | Default | Storage  | Stats target | Description
---------------+--------------+-----------+----------+---------+----------+--------------+-------------
 oid           | oid          |           | not null |         | plain    |              |
 typname       | name         |           | not null |         | plain    |              |
 typnamespace  | oid          |           | not null |         | plain    |              |
 typowner      | oid          |           | not null |         | plain    |              |
 typlen        | smallint     |           | not null |         | plain    |              |
 typbyval      | boolean      |           | not null |         | plain    |              |
 typtype       | "char"       |           | not null |         | plain    |              |
 typcategory   | "char"       |           | not null |         | plain    |              |
 typispreferred | boolean     |           | not null |         | plain    |              |
 typisdefined  | boolean      |           | not null |         | plain    |              |
 typdelim      | "char"       |           | not null |         | plain    |              |
 typrelid      | oid          |           | not null |         | plain    |              |
 typelem       | oid          |           | not null |         | plain    |              |
```

```
 typarray      | oid          |          | not null |        | plain    |             |
 typinput      | regproc      |          | not null |        | plain    |             |
 typoutput     | regproc      |          | not null |        | plain    |             |
 typreceive    | regproc      |          | not null |        | plain    |             |
 typsend       | regproc      |          | not null |        | plain    |             |
 typmodin      | regproc      |          | not null |        | plain    |             |
 typmodout     | regproc      |          | not null |        | plain    |             |
 typanalyze    | regproc      |          | not null |        | plain    |             |
 typalign      | "char"       |          | not null |        | plain    |             |
 typstorage    | "char"       |          | not null |        | plain    |             |
 typnotnull    | boolean      |          | not null |        | plain    |             |
 typbasetype   | oid          |          | not null |        | plain    |             |
 typtypmod     | integer      |          | not null |        | plain    |             |
 typndims      | integer      |          | not null |        | plain    |             |
 typcollation  | oid          |          | not null |        | plain    |             |
 typdefaultbin | ux_node_tree | C        |          |        | extended |             |
 typdefault    | text         | C        |          |        | extended |             |
 typacl        | aclitem[]    |          |          |        | extended |             |
Indexes:
    "ux_type_oid_index" UNIQUE, btree (oid)
    "ux_type_typname_nsp_index" UNIQUE, btree (typname, typnamespace)
Access method: heap)
```

4. 系统初始化的时候会自动在ux_catalog模式下创建dual表。该表只有一列，列名是dummy，数据类型是varchar(1)，并且该表只包含一条记录'X'。该表可以被所有的用户访问，但只有数据库管理员才可以删除、增加、修改表dual的内容，普通用户只能查询该表。可以通过查询表dual来计算一个常量表达式的值，因为dual只有一条记录，所以只返回一次结果。

   查询dual表,如下所示。

```
uxdb=# select * from dual;
 dummy
-------
 X
(1 row)
```

7.2.2. 基于图形界面的表管理

7.2.2.1. 创建表

通过UXDBAdmin数据库对象管理工具可视化图形界面的引导，创建表。

1. 鼠标定位在导航树上表或“表名称”所在节点，右键单击表或“表名称”。
2. 在弹出式菜单中选择“新建表”菜单项，新建表。

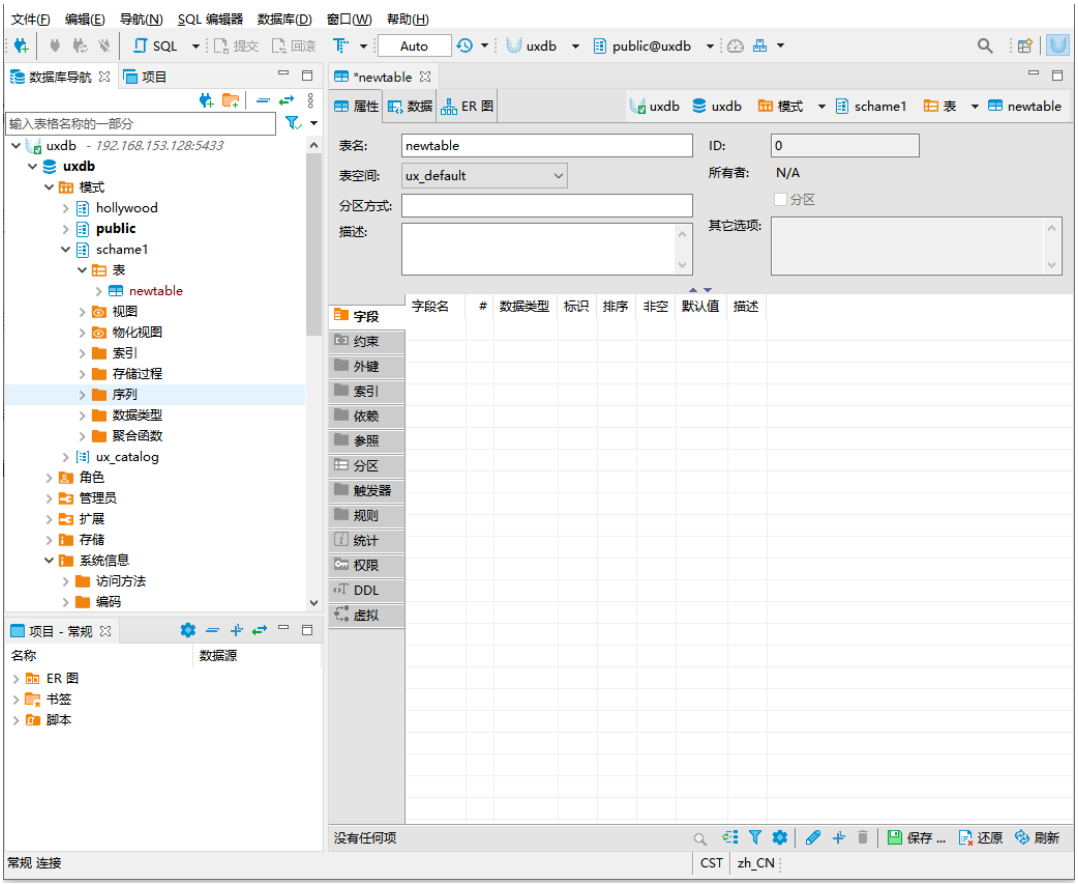


表 7.3. “基本属性”说明

名称	说明
表名称	使用名称字段为表添加描述性名称。表不能具有与同一架构中任何现有表、序列、索引、视图、外表或数据类型相同的名称。指定的名称将显示在UXDBAdmin树控件中。这个字段是必需的。
ID	表的身份ID号，一般为自动生成，用户无需对此进行设置。
表空间	可以通过下拉框选择表空间，即将表创建到指定的表空间。
所有者	表的所有者，默认情况下，表的所有者是创建表的角色。
分区方式	分区表对应的分区类型有三种，分别是范围分区(RANGE)、列表分区(LIST)和哈希分区(HASH)。
描述	允许用户添加对该表所反映的具体信息，便于了解此表的数据特性。

7.2.2.2. “字段”页

“字段”页可以定义或查看字段的基本属性，可以通过鼠标单击导航树字段，右键鼠标弹出菜单，单击新建字段完成字段的创建。或通过字段页右下角的”新建字段“快捷按钮实现字段的快速创建。

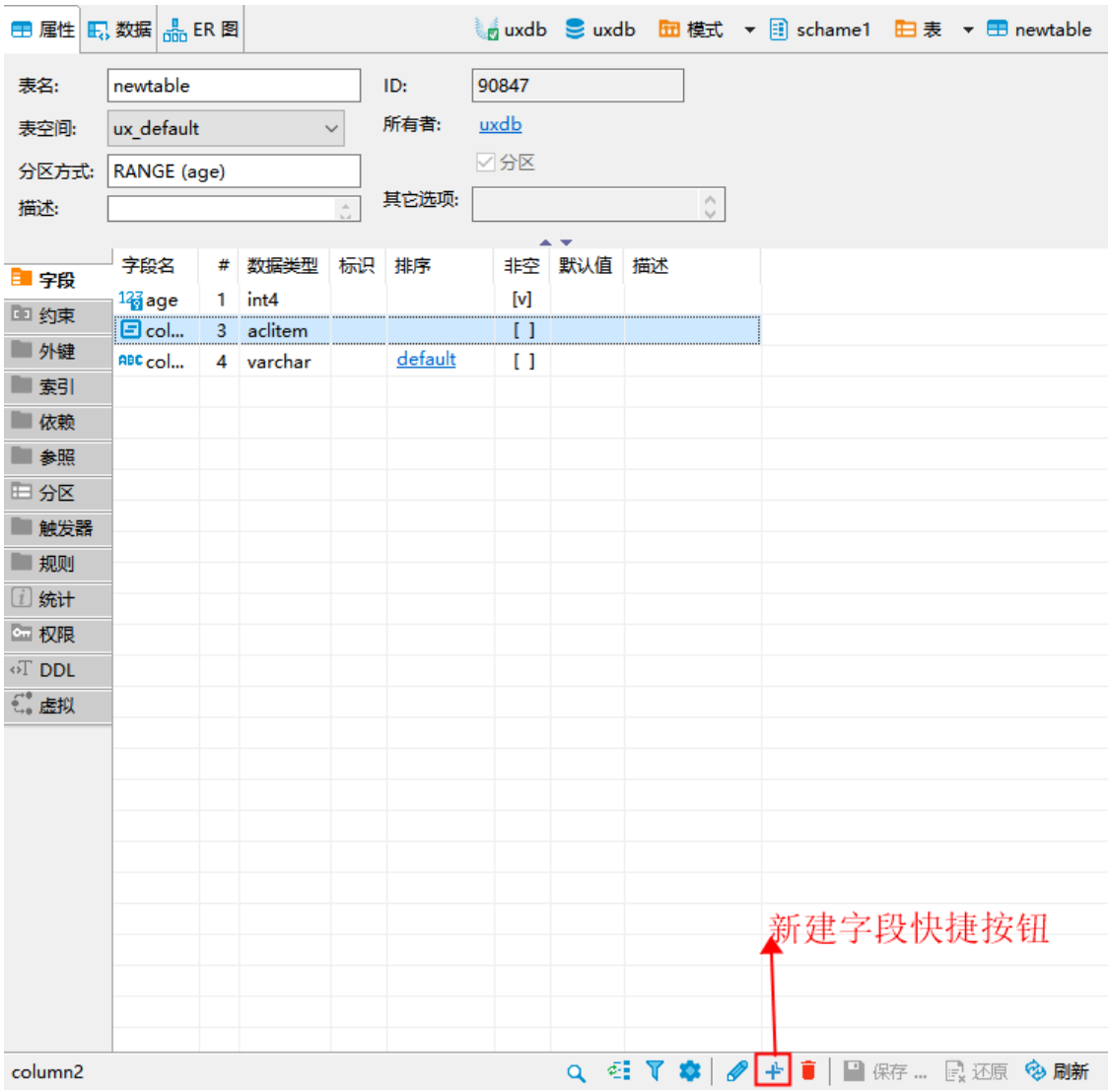


表 7.4. “字段”页说明

名称	说明
字段名称	自定义描述性名称，允许用户根据自己的习惯自定义名称或者定义一个具有代表性的名称。
数据类型	属性的数据类型名称。在“创建”操作过程中，可以指定如下列表中的任一类型作为当前列的数据类型。数据类型可以为整数类型、浮点类型、序数类型、货币类型、字符类型、二进制数据类型、日期/时间类型、布尔类型、枚举类型等。详情请参见《优炫数据库 V2.1 参考手册》“SQL语言”章节中的“数据类型”小节。
标识	下拉框中有ALWAYS和BY DEFAULT两种标识。ALWAYS指示UXDB对标识列总是生成值。如果尝试插入或更新GENERATED ALWAYS AS IDENTITY列，操作会报错。BY DEFAULT也指示UXDB对标识列总是生成值。但允许你对标识列插入或更新。

名称	说明
排序	可以通过下拉框内容选择排序方式，实现排序。
非空	为字段设置非空约束，默认为空白标识，不设置非空约束，单击后的v表示设置非空约束。
默认值	添加字段时填写的默认数据，添加或修改字段时，可以填写默认值。

7.2.2.3. “约束”页

约束属性页，可以定义或查看表的约束，支持新增和删除。

1. 鼠标定位在导航树上约束或“约束名称”所在节点，右键单击约束或“约束名称”。
2. 在弹出式菜单中选择“新建约束”菜单项，然后按照向导创建约束。

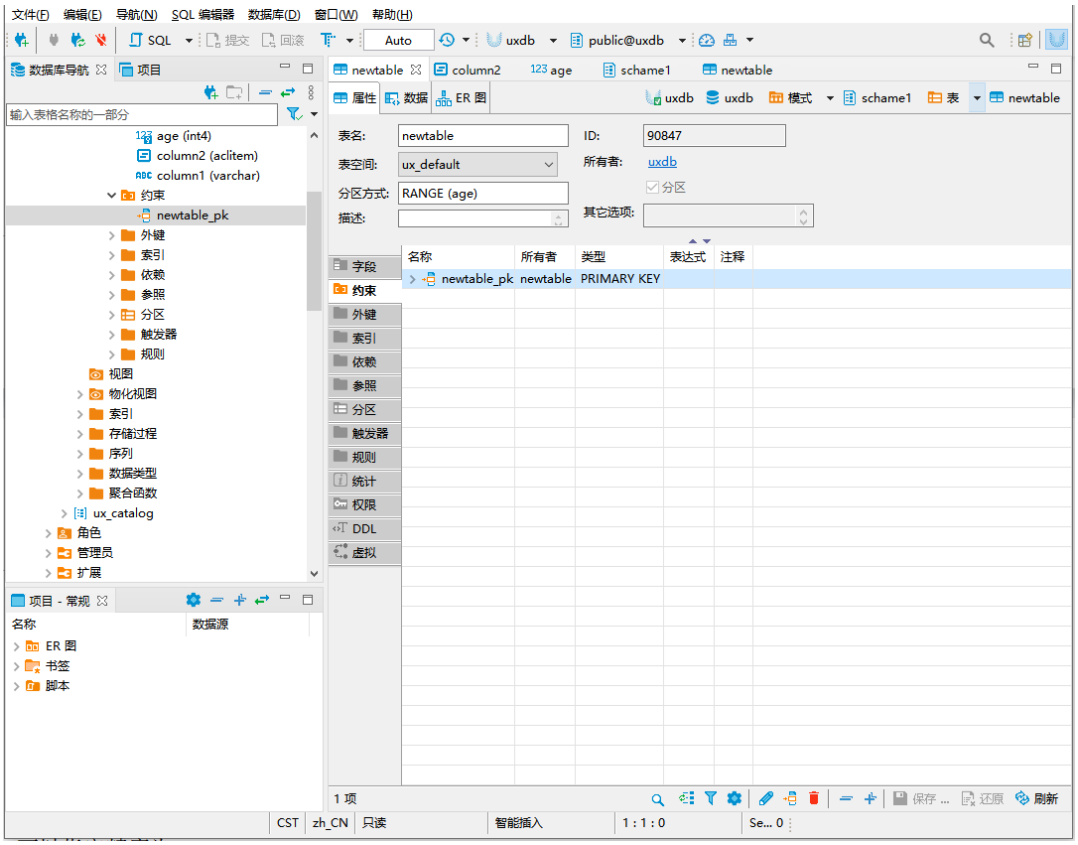


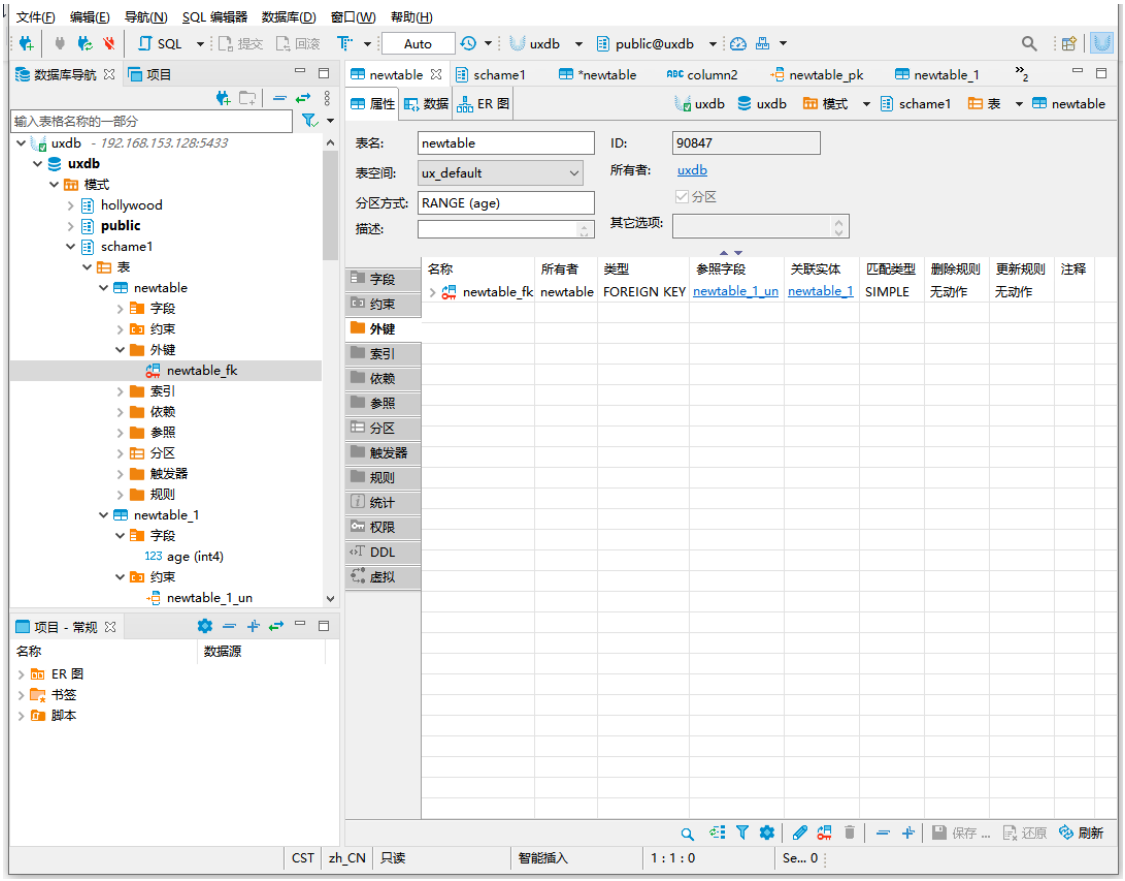
表 7.5. “约束”页说明

名称	说明
名称	待创建的或者当前查看的约束的名称。在“创建”和“修改”操作过程中，用户可以指定一个有效的数据库标识符作为约束名。
所有者	约束的属主，通常是创建约束的表。
类型	类型属于可选项，可选主键、唯一键、检查三者之一作为类型。 1. 主键

名称	说明
	选中后，指示相应列为“主键约束”的成员。 “主键约束”可以理解为“非空约束”和“唯一约束”的组合。它表示一个字段或者是若干个字段的组合，可以用于唯一标识表中的某数据行该信息只在创建表的过程中可见。 2. 唯一键 选中后，指示相应列的数据在系统中需要满足“唯一约束”。 “唯一约束”保证相应列的数据与表中其它行相应列的数据相比是唯一的。但是允许多行有空值（NULL）。 3. 检查 用于定义相应列上的“检查约束”，在插入或修改之前，要求数据满足表达式或条件。 利用“检查约束”可以为需要的字段定义必须满足的表达式。“检查约束”的表达式应该包含受约束的列名。
表达式	显示创建检查约束时，所输入并生效的表达式，支持界面修改。
注释	用户可以添加需要描述的内容。

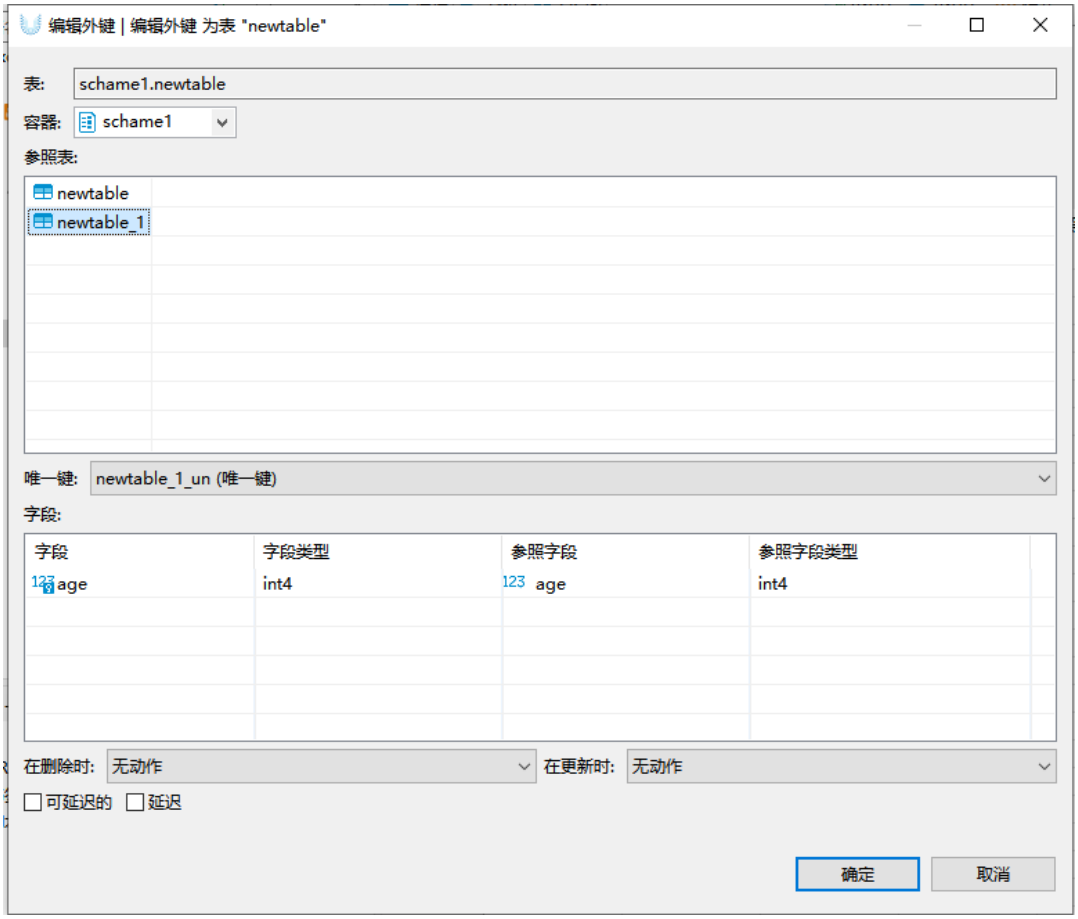
7.2.2.4. “外键”页

外键属性页，可以定义或查看外键的主要属性，可以通过导航树右键菜单实现外键的创建、删除。用于定义相应列上的“外键约束”，“外键约束”用于声明当前列的数值必须匹配另外一个表中的某列出现的数值。系统会给当前外键自动起一个外键名，用户也可以进行修改，在“外键”属性页中，用户可以通过下拉列表，选择当前字段的相关信息。



1. 鼠标定位在导航树上外键或“外键名称”所在节点，右键单击外键或“外键名称”。

2. 在弹出式菜单中选择“新建外键”菜单项，然后按照向导创建外键。



3. 按照上图的向导，选择并填写对应的信息后，单击确定即可进入下个界面，然后根据引导单击保存按钮。

4. 若需删除外键，在左侧导航树的外键名称上鼠标右键，单击删除即可，或在外键页的外键项页面上，右键鼠标，在弹出的菜单中选择删除，实现外键的删除。

7.2.2.5. “索引”页

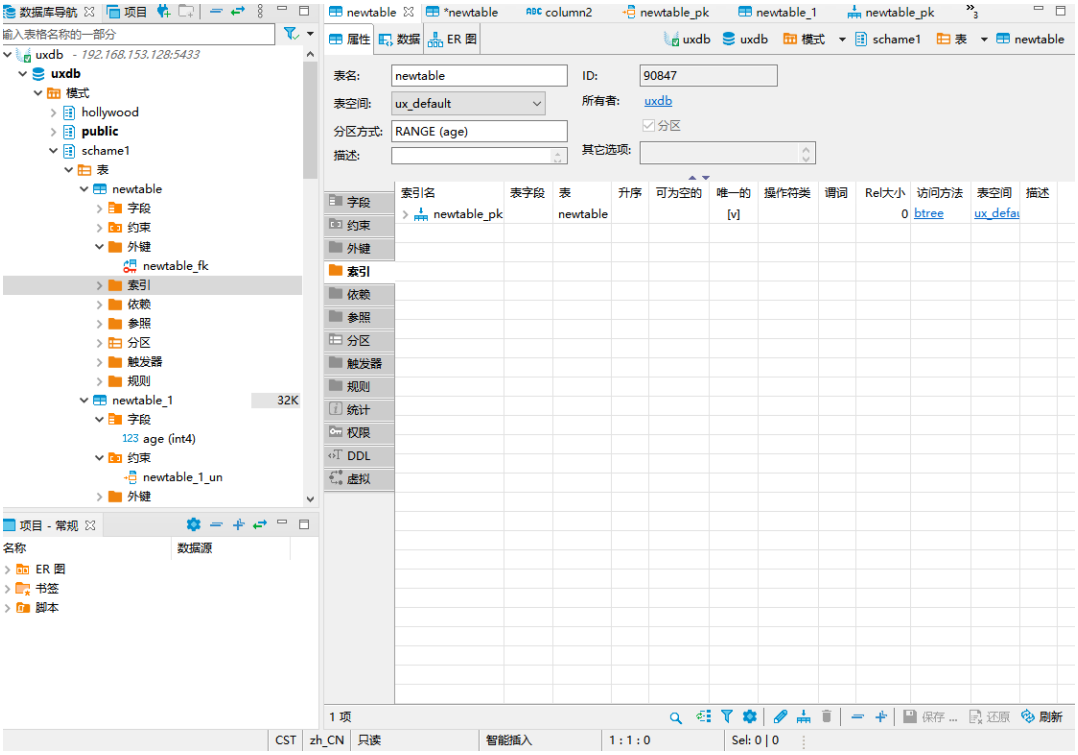
索引属性页，可以定义或查看索引的主要属性，支持索引的创建、删除。

1. 鼠标定位在导航树上索引或“索引名称”所在节点，右键单击索引或“索引名称”。

2. 在弹出式菜单中选择“新建索引”菜单项，然后按照向导创建索引。

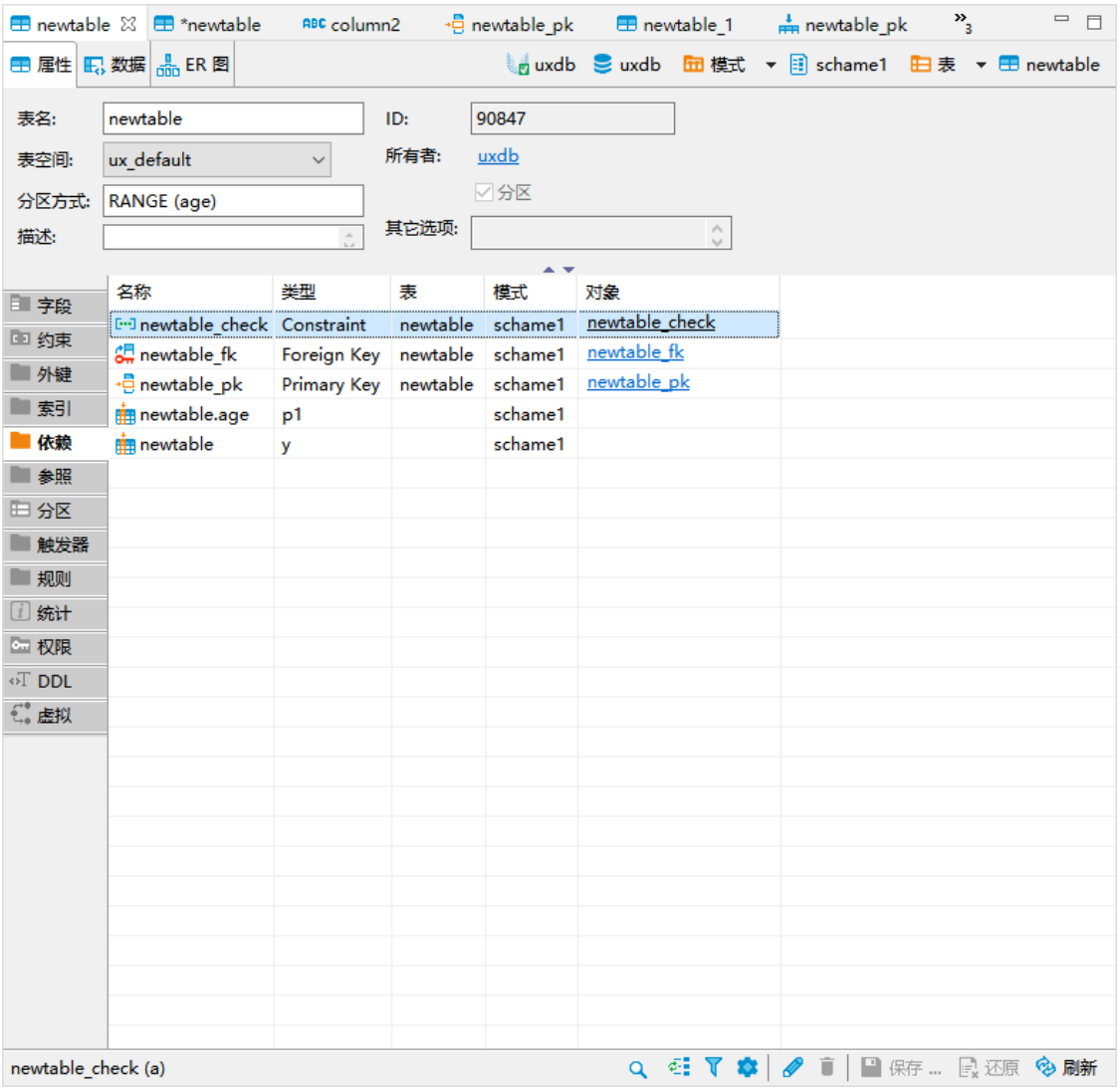
3. 在索引页的右侧功能区，鼠标右键，然后选择新建索引，也可以是实现索引的创建。

4. 索引删除，鼠标单击导航树需要删除的索引项或者在右侧工作表中选中索引，单击鼠标右键，在弹出的右键菜单中，选择删除，就可以删除索引项。



7.2.2.6. “依赖”页

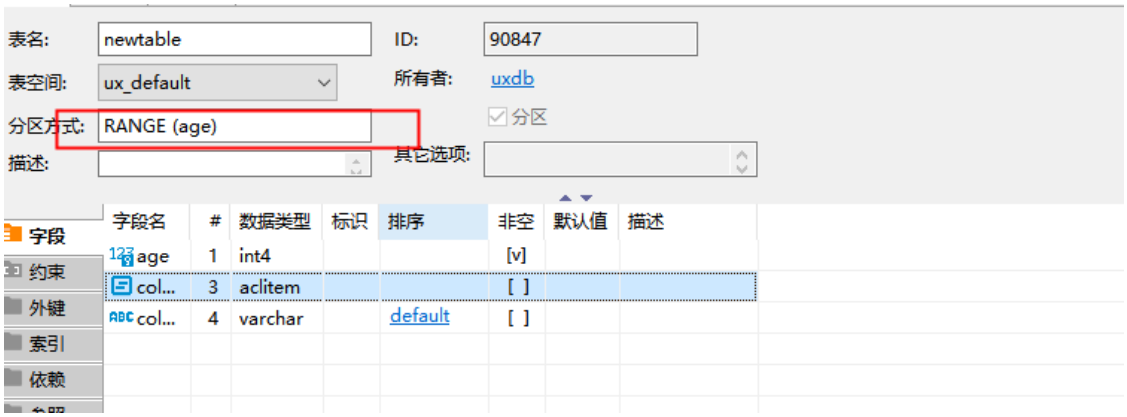
依赖属性页，主要描述数据表的依赖关系，可以查看依赖项的主要属性。



7.2.2.7. “分区”页

分区属性页，可以定义或查看分区的主要属性， 分区表就是根据分区策略，将数据分散到不同的子表中，并通过父表建立关 联关系，从而实现数据物理上的分区。分区表对应的分区类型有三种，分别是范 围分区(RANGE)、列表分区(LIST)和哈希分区(HASH)。

前提条件：在创建表时，需要在分区方式上指定分区类型及分区键。



1. 鼠标定位在导航树上分区或“分区名称”所在节点，右键单击分区或“分区名称”。

2. 在弹出式菜单中选择“新建分区”菜单项，然后按照向导创建分区。

3. 创建分区表，填写如下所示对应信息，然 后单击右下角保存按钮，弹出sql语句框，单击执行按钮，分区表创建成功。

4. 若要删除分区，鼠标定位在导航树上“分区名称”所在节点，右键单击“分区名称”。

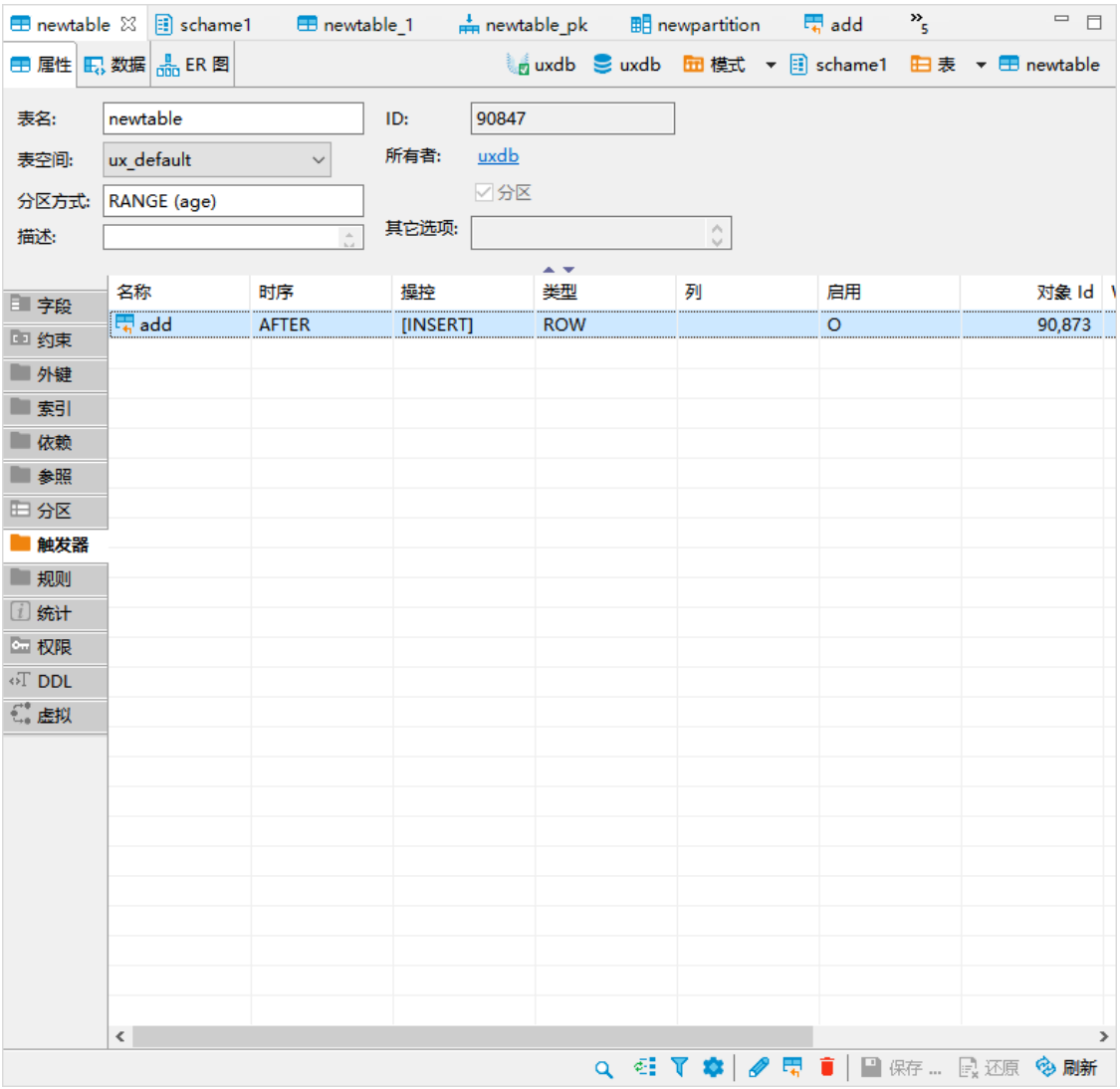
5. 在弹出式菜单中选择“删除”菜单项即可删除分区，也可以使用delete快捷键删除。

表 7.6. “分区”页说明

名称	说明
表名	这里是指创建分区的名字，即分区的标识。
分区表达式	使用操作符来进行定义分区，也被称为分区表达式。
分区方式	分区方式分为三种，主要包括Range：范围分区、List：列表分区及Hash：哈希分区。

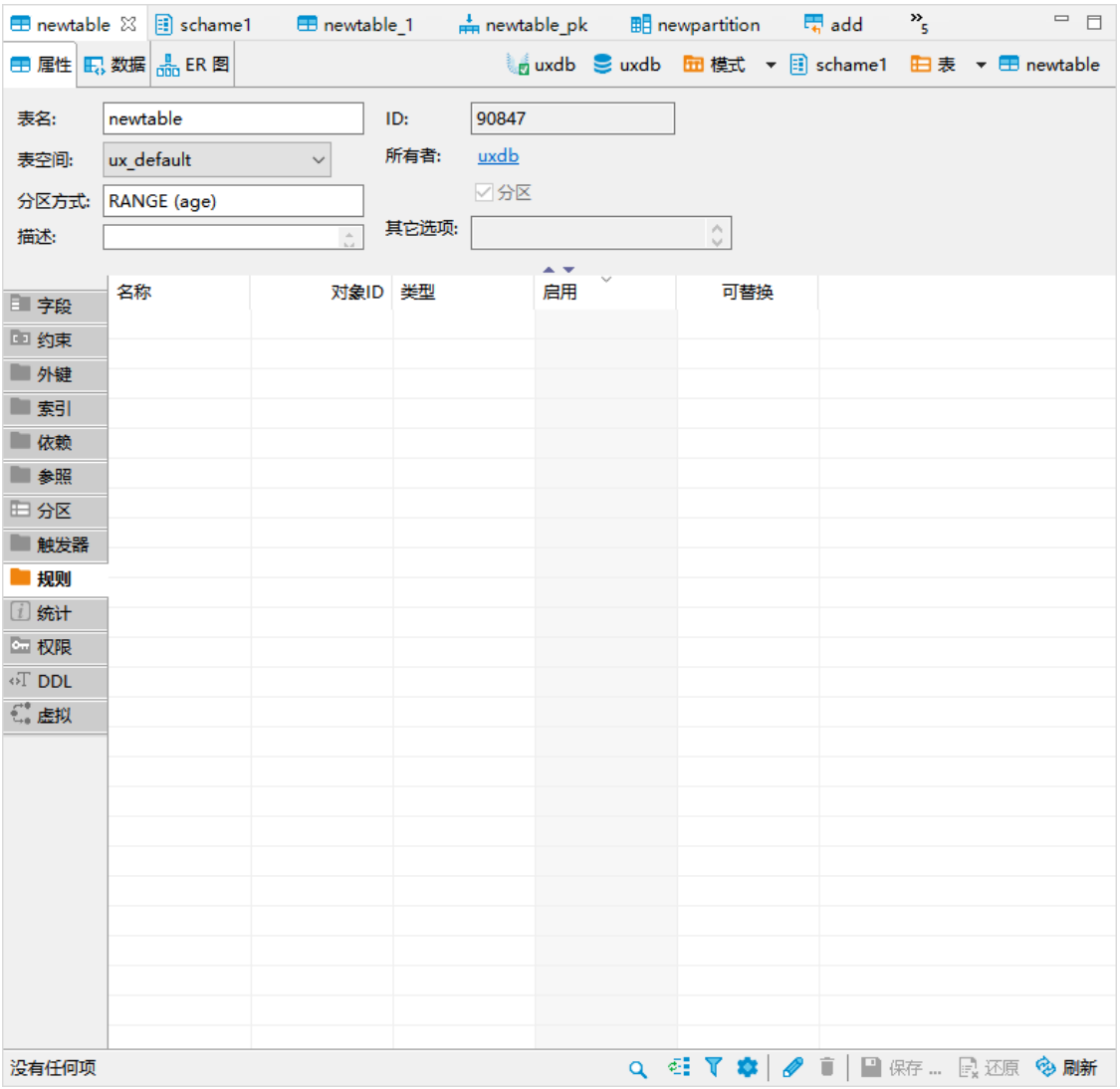
7.2.2.8. “触发器”页

触发器属性页，可以定义或查看触发器的主要属性。



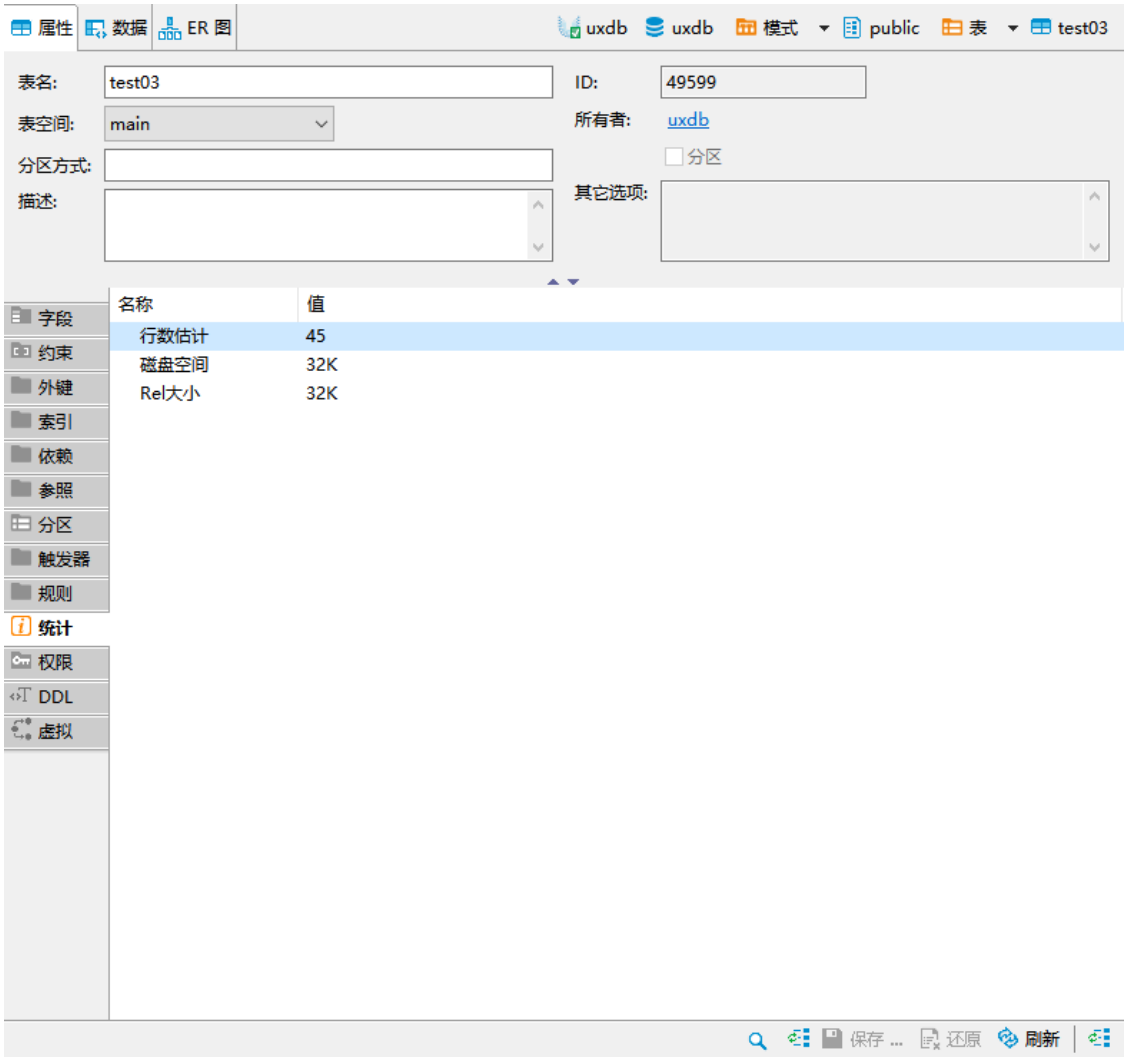
7.2.2.9. “规则”页

规则属性页，可以查看规则的主要属性。



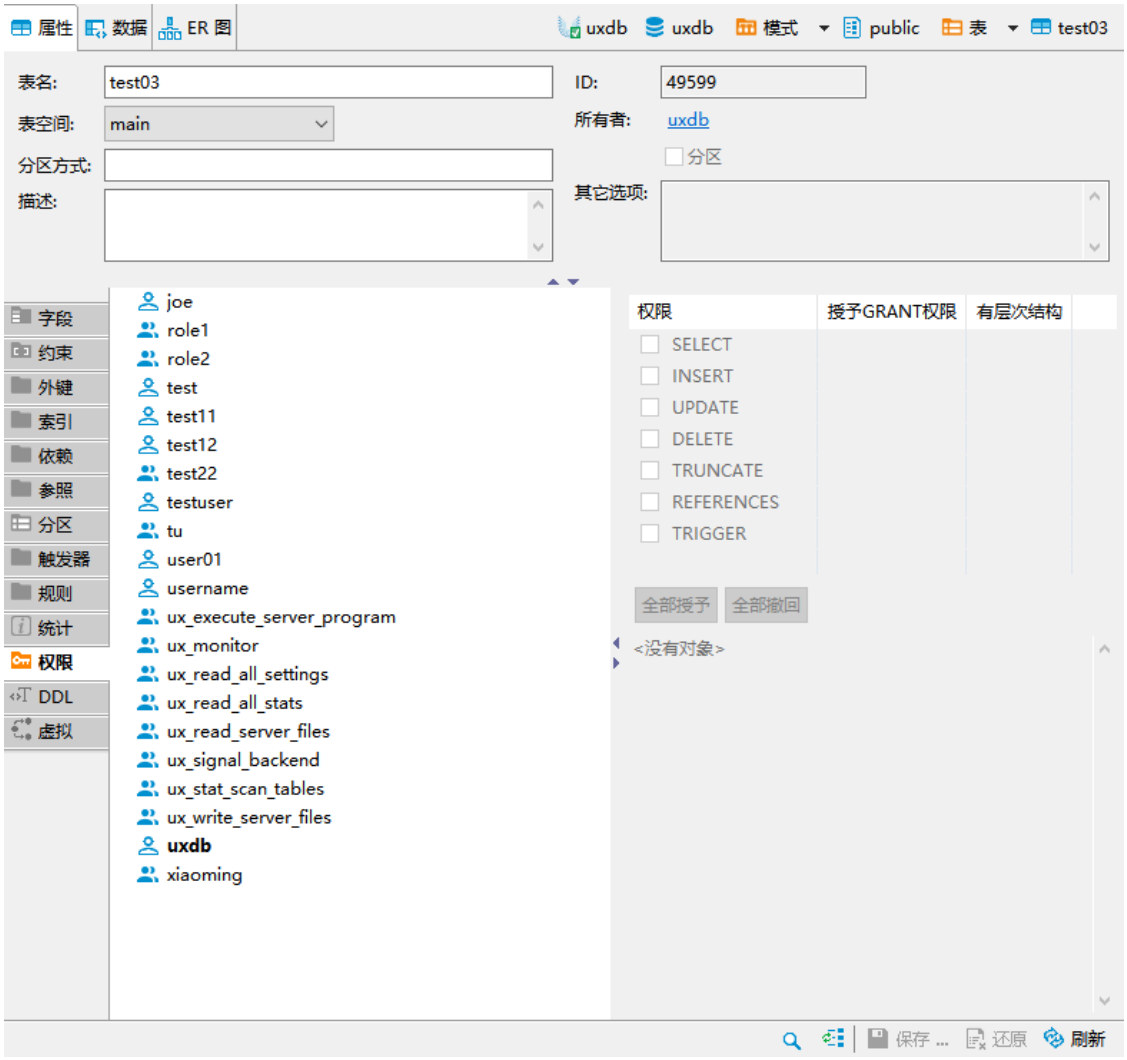
7.2.2.10. “统计”页

统计属性页，可以查看表数据的行数估计、磁盘空间及Rel大小。



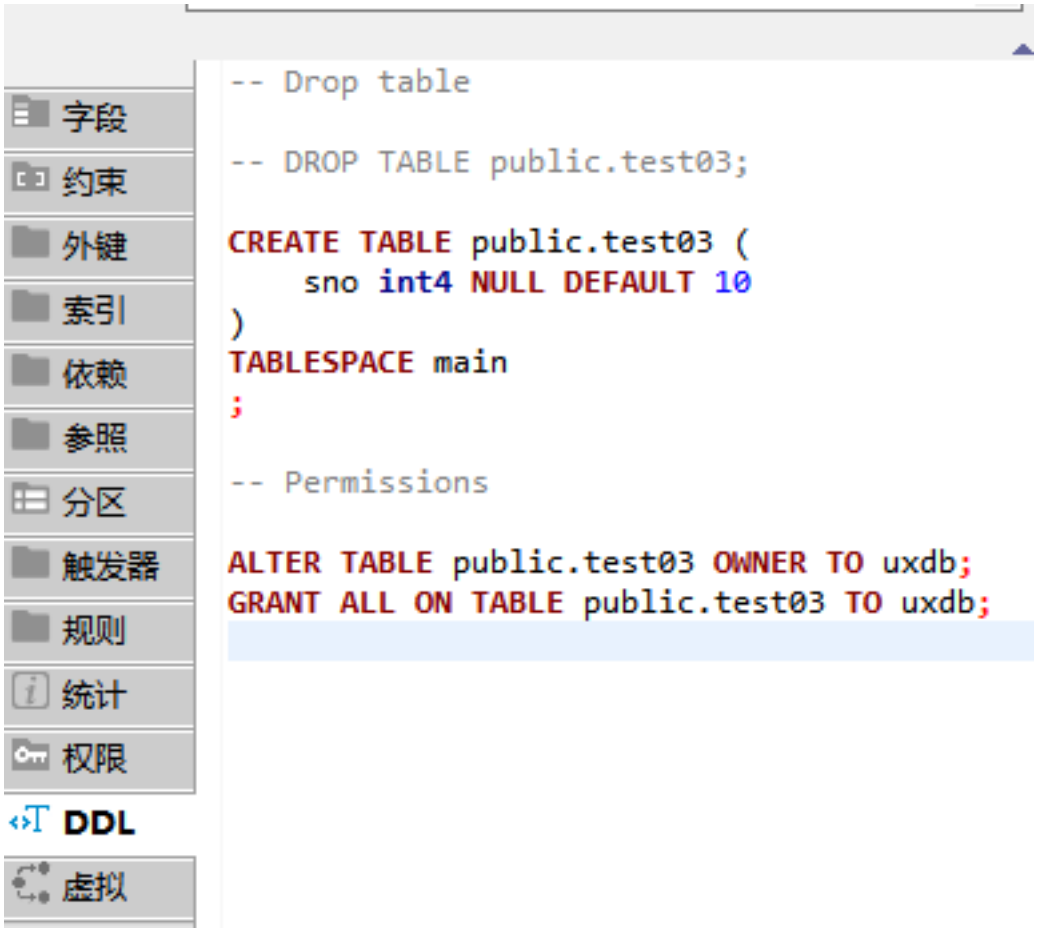
7.2.2.11. “权限”页

权限属性页，可以定义或查看表的权限，切换“用户”/“角色”时，权限将随之更新。通过该权限页可以实现权限的修改，如授予权限、权限撤回。



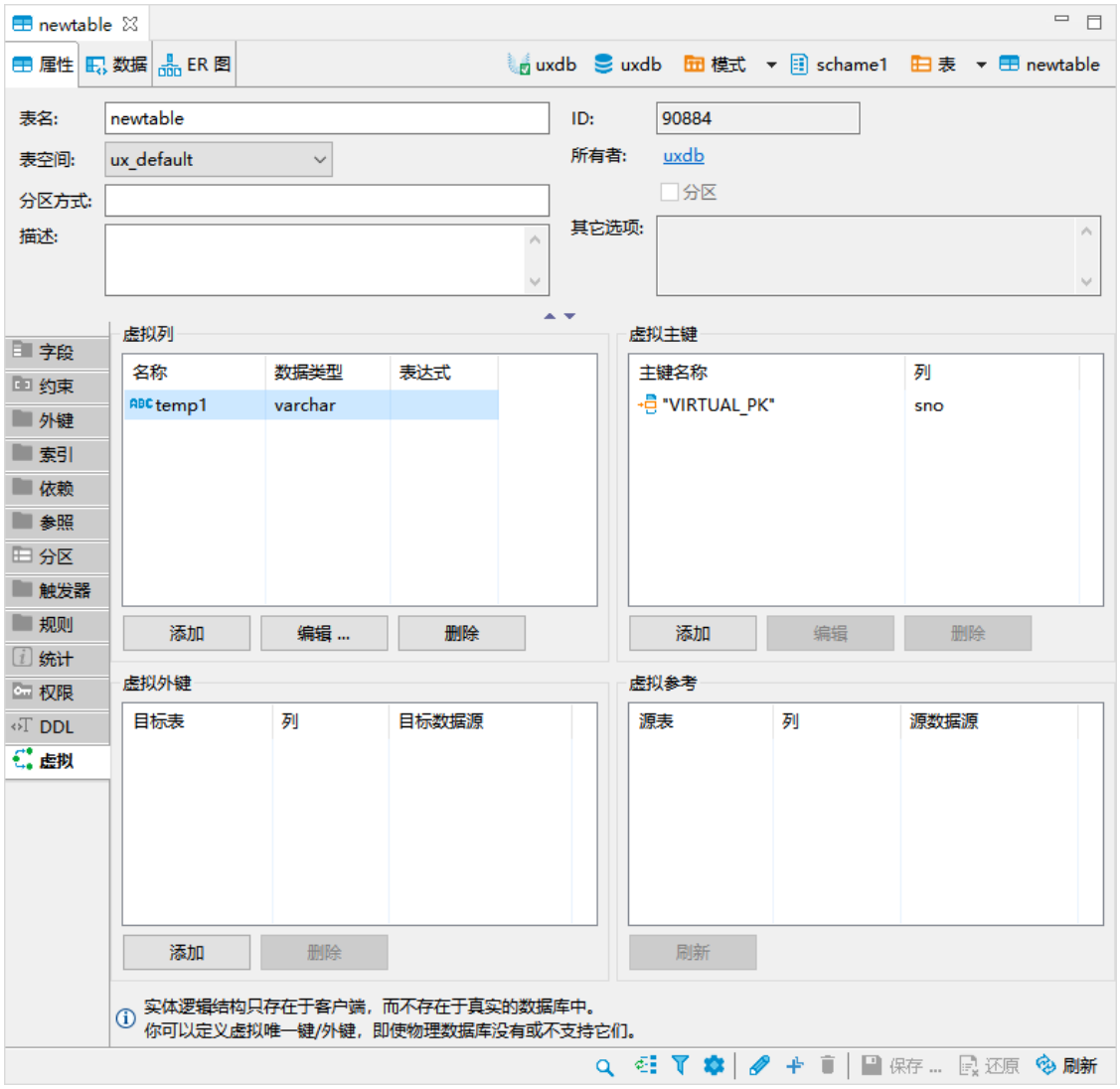
7.2.2.12. “DDL”页

通过信息浏览区DDL对话框，用户可以查看表的 DDL 语言。DDL命令用于构建一些数据，例如它可以创建和更改数据库和表。



7.2.2.13. “虚拟”页

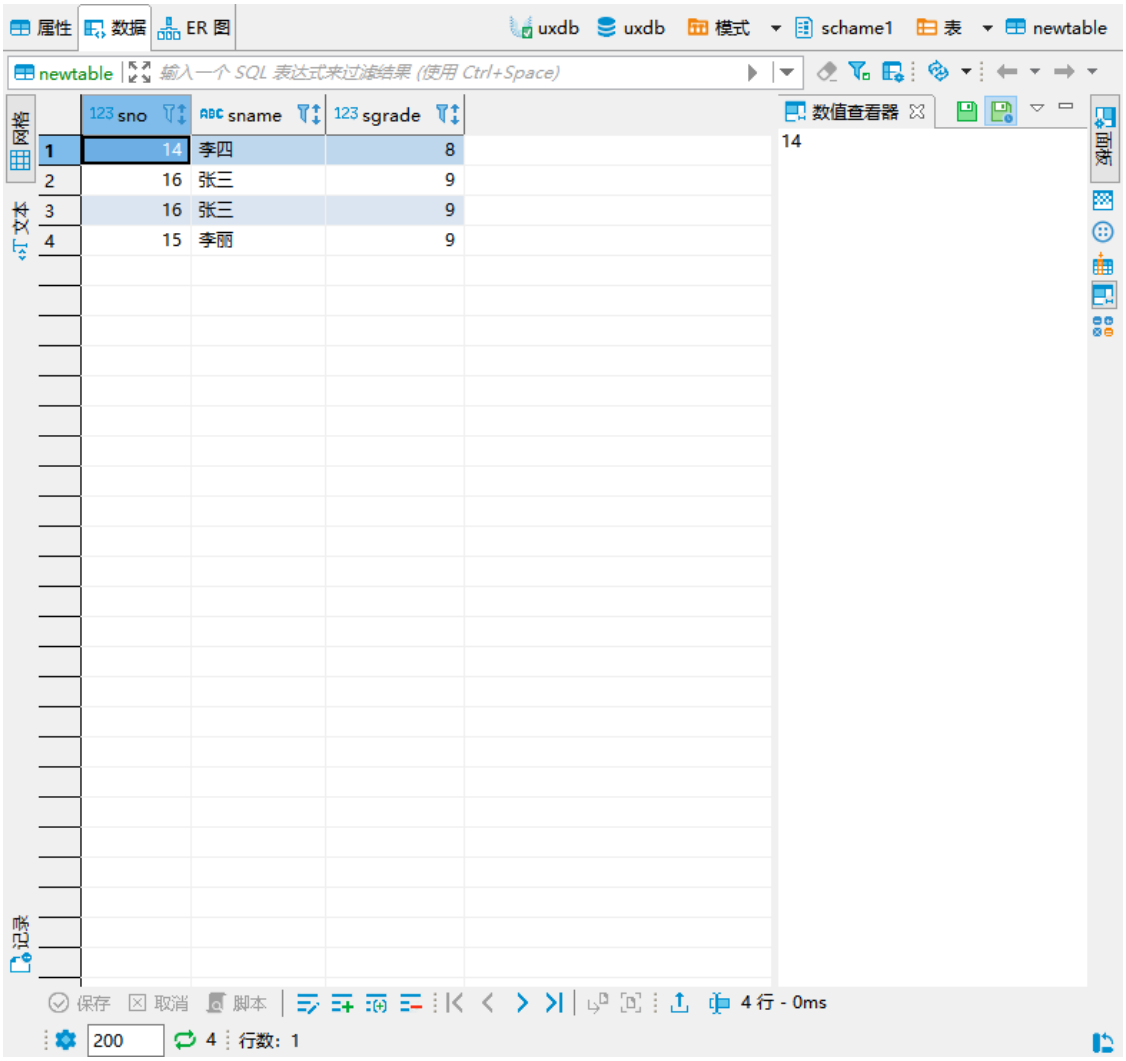
虚拟属性页，可以定义查看虚拟列、虚拟主键、虚拟外键及虚拟参考。虚拟的功能项只存在于客户端，并不存在真实的数据库中，主要辅助用户使用，提升用户的操作性。



7.2.2.14. “数据”页

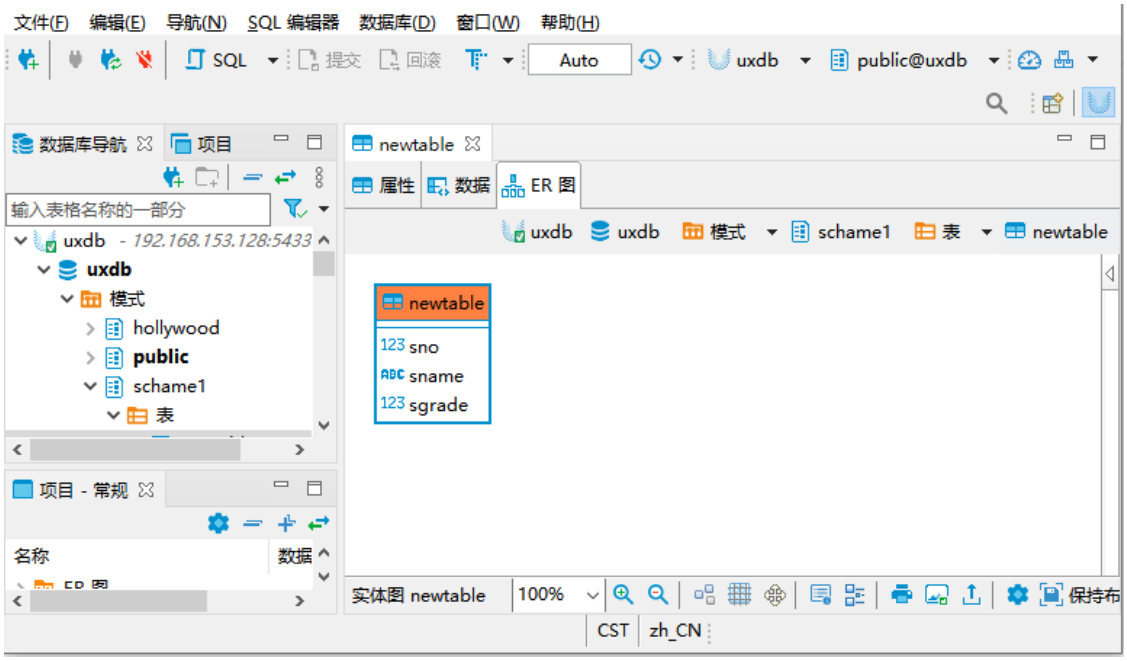
数据页提供增加、删除、修改、查看表数据的功能。通过使用界面上的快捷按钮实现对应的功能，如单击”+”号，能够快速实现数据的添加。

若需删除数据，通过选中需要删除的数据行，单击界面上的删除按钮，可以实现数据的删除，使用此功能也可以实现多项删除。



7.2.2.15. “ER图”页

实体关系图 （ERD） 是数据库实体及其之间关系的图形表示。“ER图”页允许用户查看现有表的关系图。同时，在该界面提供了放大、缩小、图表打印、图表导出等功能。

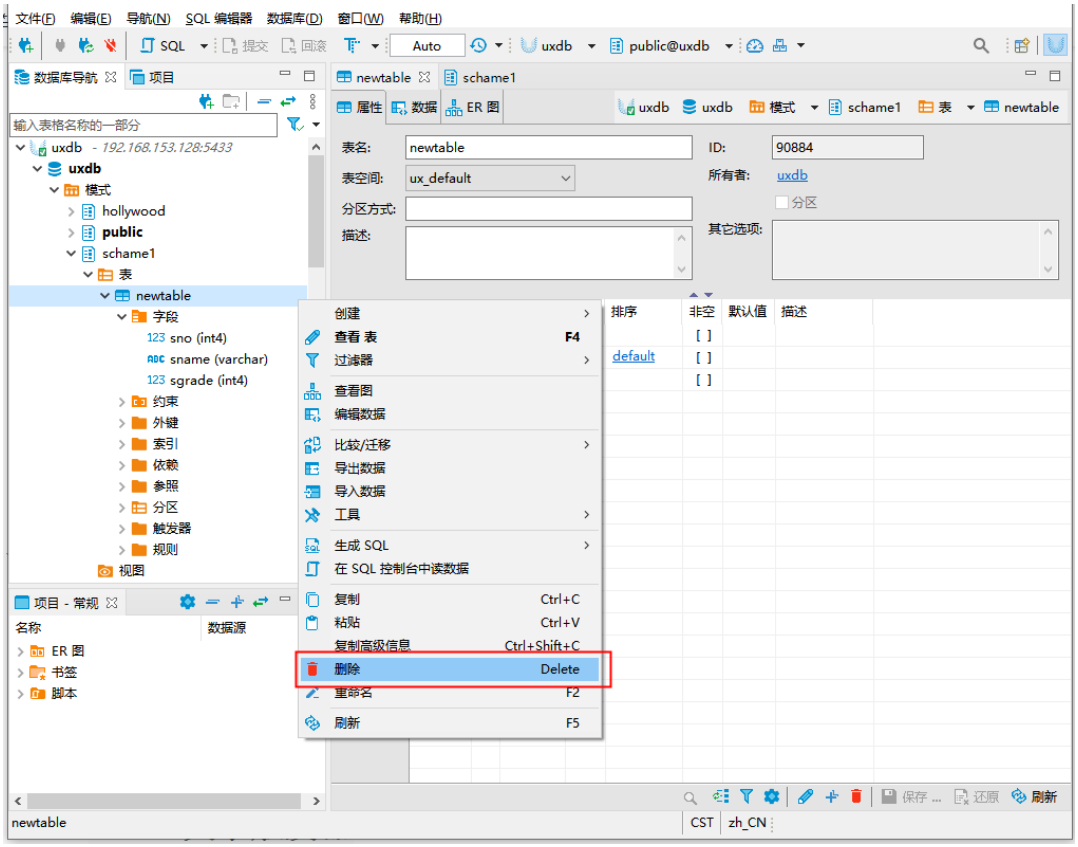


7.2.2.16. 删除表

通过UXDB数据库对象管理工具可视化图形界面的引导，删除表。

1. 鼠标定位在导航树上“表名称”所在节点，右键单击“表名称”。

2. 在弹出式菜单中选择“删除”菜单项，或按快捷键Delete。



7.2.3. 基于SQL语句的表管理

7.2.3.1. CREATE TABLE

- 大纲

  CREATE TABLE tablename ...

- 描述

  数据库中的表由CREATE TABLE语句定义。

  详细参数请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“CREATE TABLE”。

> **注意**
>
> 1. 在该语句中必须对基表命名，并且说明该表中每个列的列名、数据类型等。
> 2. 创建一个表，必须分别具有所有列类型或OF子句中类型的USAGE特权。

- 示例

  创建表my_first_table，如下所示。

  CREATE TABLE my_first_table(

```
first_column TEXT,
second_column INTEGER
);
```

7.2.3.2. ALTER TABLE

- 大纲

  ALTER TABLE ...

- 描述

  表创建后，可以使用ALTER TABLE语句修改表的定义。

  详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“ALTER TABLE”。

- 示例

  - 给表增加一列、删除表中的列、修改表名、列名等。

    - 向一个表增加一个类型为varchar的列。

      ```
      ALTER TABLE my_first_table ADD COLUMN address varchar(30);
      ```

    - 从表中删除一列。

      ```
      ALTER TABLE my_first_table DROP COLUMN address RESTRICT;
      ```

    - 在一个操作中更改两个现有列的类型。

      ```
      ALTER TABLE my_first_table
      ALTER COLUMN first_column TYPE varchar(80),
      ALTER COLUMN second_column TYPE varchar(100);
      ```

    - 重命名表。

      ```
      ALTER TABLE my_first_table RENAME TO my_table;
      ```

  - 使用INSERT、DELETE、UPDATE语句对表中数据进行增、删、改操作。

    - 把表films的列kind 中的单词Drama改成Dramatic。

      ```
      UPDATE films SET kind = 'Dramatic' WHERE kind = 'Drama';
      ```

    - 删除表films中Musical。

      ```
      DELETE FROM films WHERE kind <> 'Musical';
      ```

    - 向films中插入一行。

```
INSERT INTO films VALUES ('UA502', 'Bananas', 105, '1971-07-13', 'Comedy', '82
 minutes');
```

7.2.3.3. DROP TABLE

- 大纲

  DROP TABLE

- 描述

  当不再使用一个表时，可以将其删除。删除表时，将产生以下结果：

  1. 表的结构信息从数据字典中删除，表中的数据不可访问。
  2. 表上的所有索引和触发器被一起清除。
  3. 所有建立在该表上的同义词、视图和存储过程变为无效。
  4. 所有分配给表的簇标记为空闲，可被分配给其他的数据库对象。

  一般情况下，普通用户只能删除自己模式下的表。若要删除其他模式下的表，则必须具有DROP TABLE的数据库权限。

  详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“DROP TABLE”。

- 示例

  - 删除表my_first_table。

    ```
    DROP TABLE my_first_table;
    ```

  - 如果要删除的表被其他表引用，即其他表的外键引用了表的任何主键或唯一键，则需要在DROP TABLE语句中包含CASCADE选项，如下所示。

    ```
    DROP TABLE my_first_table CASCADE;
    ```

7.2.3.4. 清空表

某些情况下，当表的数据不再使用时，需要删除表的所有行，即清空该表。 UXDB支持以下方式来删除表中的所有的行。

1. 使用DELETE语句。
2. 使用DROP和CREATE语句。
3. 使用truncate语句。

7.2.3.4.1. 使用DELETE

- 大纲

  DELETE FROM my_first_table;

- 描述

使用DELETE语句删除表中的行。下面的语句将删除my_first_table表中的所有行：

详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“DELETE”。

但当表有很多行时，使用DELETE清空表会消耗过多的系统资源。

7.2.3.4.2. 使用DROP和CREATE

- 大纲

```
DROP TABLE my_first_table;
CREATE TABLE my_first_table (…);
```

- 描述

使用DROP删除一个表，然后创建一个同名的表，也可以达到清空表的效果。例如，下面的语句先删除my_first_table表，之后再重新创建。

当删除和重新创建表时，所有与之相关联的索引、完整性约束和触发器也被删除。同样，所有针对被删除表的授权也会被删除。

7.2.3.4.3. 使用truncate

- 大纲

```
truncate TABLE my_first_table;
```

- 描述

使用truncate语句能删除表中的所有行。下面的语句清空my_first_table 表。

truncate语句提供了一种快速、有效地删除表所有行的方法。并且truncate是一个DDL语句，不会产生任何回滚信息。执行truncate会立即提交，而且不能回滚。

详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“TRUNCATE”。

truncate语句并不影响与被删除的表相关联的任何结构、约束、触发器或者授权。另外，UXDB数据库truncate表后，原来分配给该表的空间会被释放，供其他数据库对象使用，大大提高空间的利用效率。

> **注意**
>
> 一般情况下，普通用户只能truncate自己模式下的表。若要truncate其他模式下的表，则必须具有DROP TABLE的数据库权限。
>
> 如果要清空的表被其他表引用，即其他表的外键引用了表的任何主键或唯一键，并且子表不为空或子表的外键约束未被禁用，则不能truncate该表。

7.2.3.5. 查询表信息

表创建后，可以使用SELECT语句对表中数据进行查询操作。

查询DUAL表，如下所示。

SELECT * FROM DUAL;
DUMMY

X
(1 row)

详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“SELECT”。

7.3. 视图

7.3.1. 关于视图

视图是CREATE VIEW语句定义的导出表，由一个或多个表（或其他视图）中的数据的一种定制的表示，是用一个查询定义的。所以视图可认为是一个存储的查询或是一个虚表(virtual table)。视图几乎可在任何使用表的地方使用。

由于视图是由表导出的，所以视图和表存在许多类似。视图可以被查询，而在修改、插入或删除时具有一定的限制。例如，在视图上执行的操作影响了视图的基本表中的数据时，该操作应受到基本表的完整性约束和触发器的限制。

视图与表不同。一个视图不分配任何存储空间，不真正地包含数据。由查询定义的视图对应于视图引用表中的数据。视图只在数据字典中存储其定义。

引入视图可以通过限制对表的预定义行集合的存取，为表提供附加的安全性；可以为用户简化命令；可用于保存复杂查询等。

视图是一个虚拟表，其内容由查询语句（SELECT）定义。同真实的表一样，视图包含一系列带有名称的列和行数据。但是，视图并不在数据库中以存储的数据值集形式存在。行和列数据来自由定义视图的查询所引用的表，并且在引用视图时动态生成。

物化视图是查询结果集的一份持久化存储，所以它与普通视图完全不同，而非常趋近于表。 可以是基础表中部分数据的一份简单拷贝，也可以是多表join之后产生的结果或其子集，或者原始数据的聚合指标等等。所以，物化视图不会随着基础表的变化而变化，所以它也称为快照（snapshot）。

7.3.2. 基于图形界面的视图管理

7.3.2.1. 创建视图

通过UXDB数据库对象管理工具可视化图形界面的引导，创建视图。

1. 鼠标定位在导航树上视图或“视图名称”所在节点，右键单击视图或“视图名称”。
2. 在弹出式菜单中选择“新建视图”菜单项，新建视图。

7.3.2.2. “基本属性”页

基本属性页，可以定义或查看视图的主要属性。

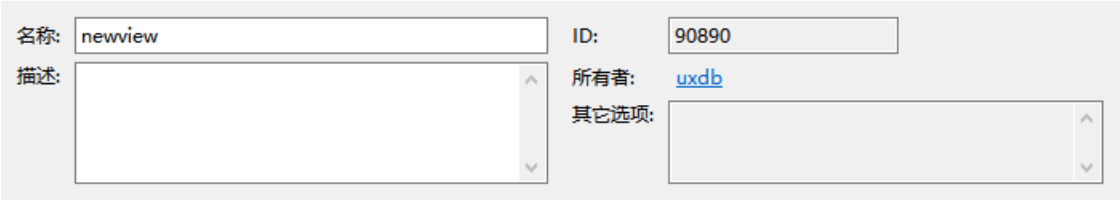
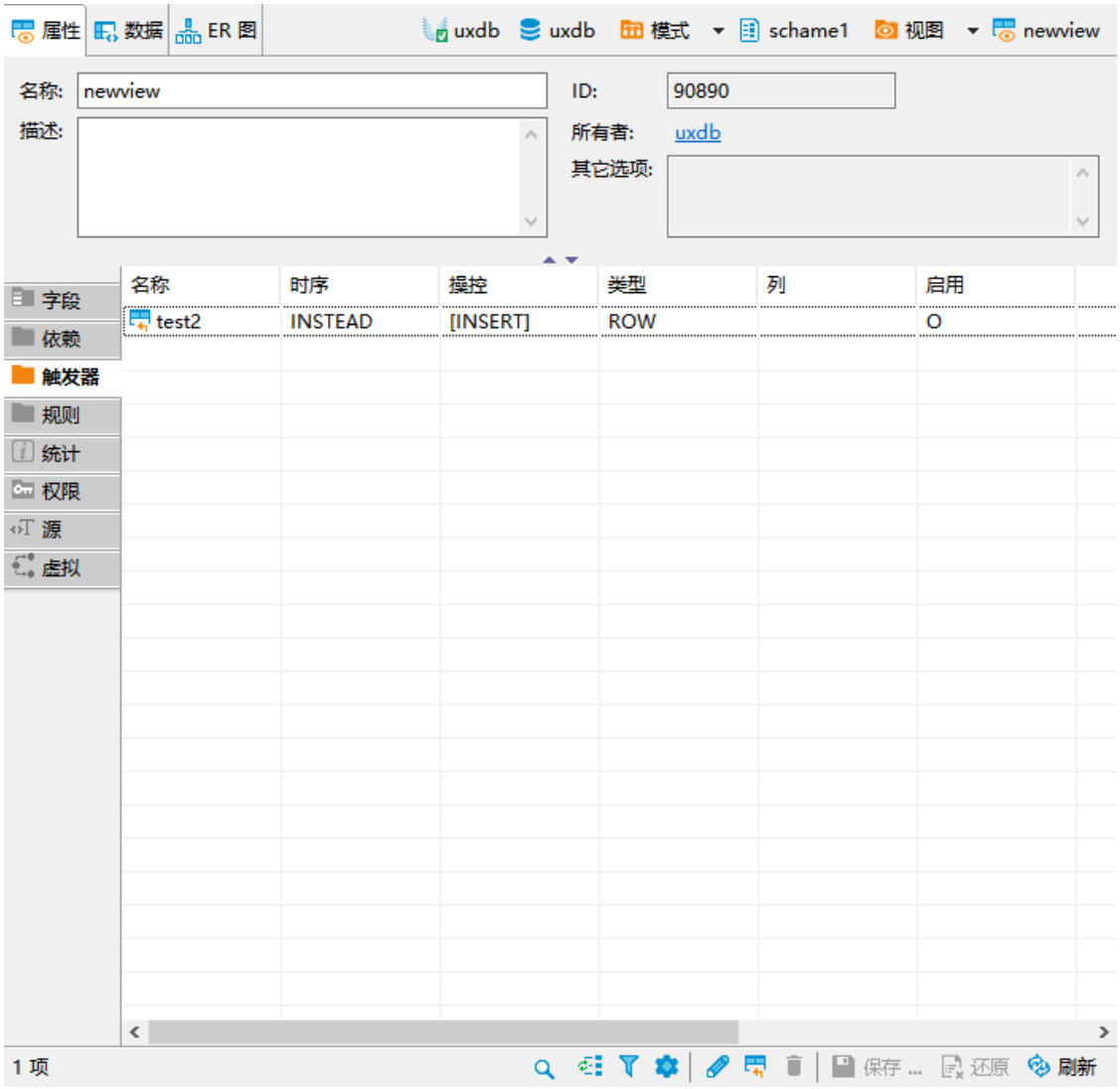


表 7.7. “基本属性”页说明

名称	说明
名称	待创建的或者当前查看的视图的名称。在“创建视图”操作过程中，用户可以指定一个有效的UXDB数据库标识符作为视图名。同一个模式中视图名不能重复。
所有者	该视图的所有者名称，所有者应该是数据库系统的用户之一。

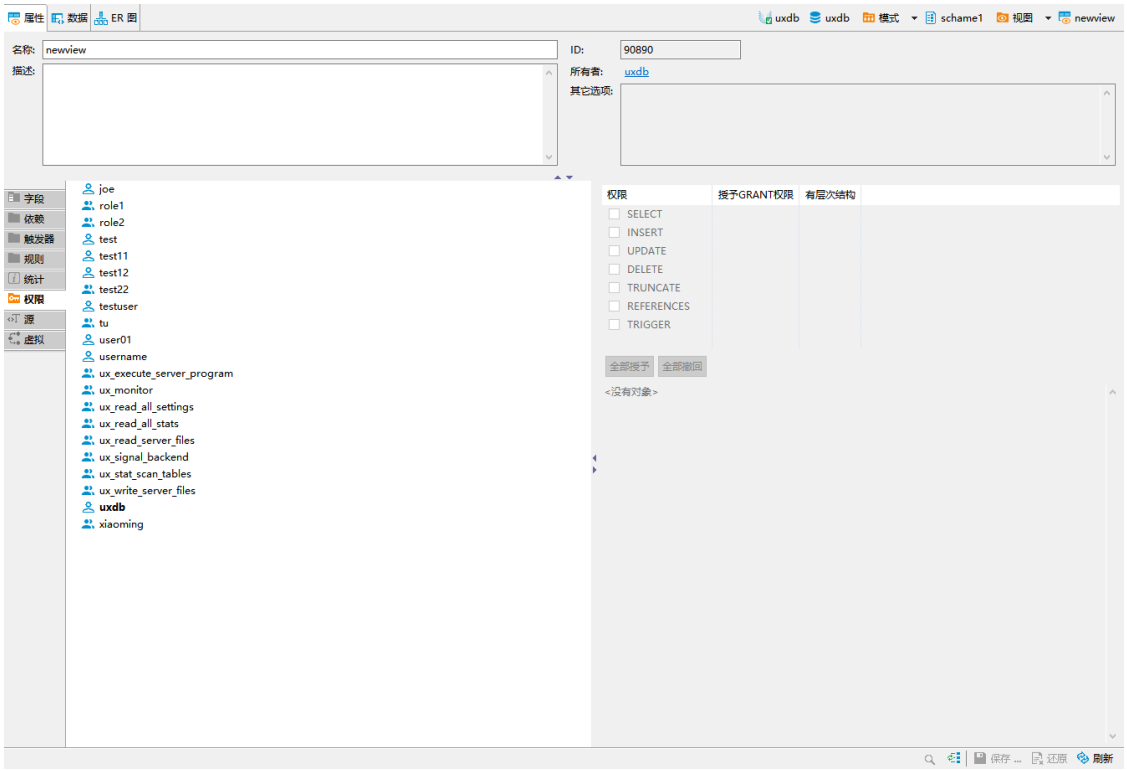
7.3.2.3. “触发器”页

触发器页，可以定义或查看触发器的主要属性，可以通过导航树上触发器功能项进行触发器的创建、删除。



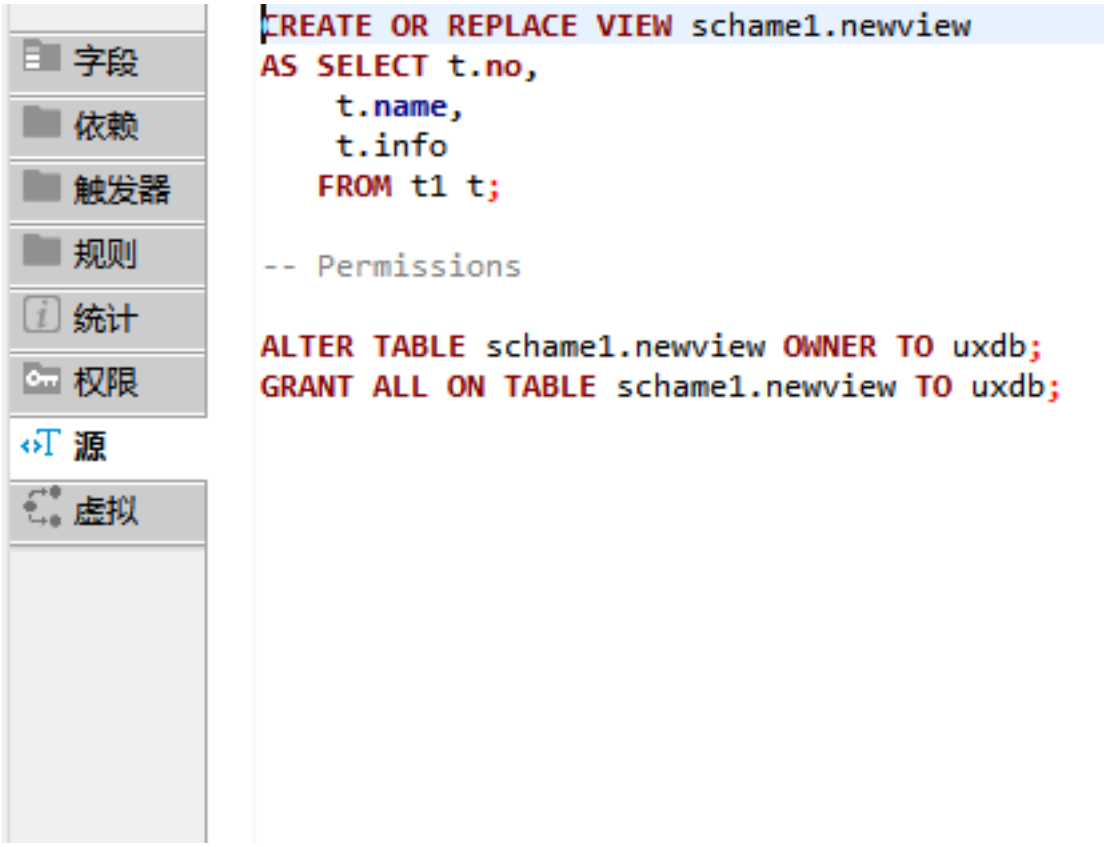
7.3.2.4. “权限”页

权限属性页，可以定义或查看视图的权限。切换“用户”/“角色”时，权限将随之更新。



7.3.2.5. “源”页

通过信息浏览区“源”对话框，用户可以查看视图的DDL语言。



7.3.2.6. 删除视图

通过UXDB数据库对象管理工具可视化图形界面的引导，删除视图。

1. 鼠标定位在导航树上“视图名称”所在节点，右键单击“视图名称”。
2. 在弹出式菜单中选择“删除”菜单项，或按快捷键Delete。

7.3.2.7. 创建物化视图

通过UXDB数据库对象管理工具可视化图形界面的引导，创建物化视图。

1. 鼠标定位在导航树上物化视图或“物化视图名称”所在节点，右键单击物化视图或“物化视图名称”。
2. 在弹出式菜单中选择“新建物化视图”菜单项，新建物化视图。

7.3.2.8. 物化视图的“基本属性”页

基本属性页，可以定义或查看视图的主要属性。

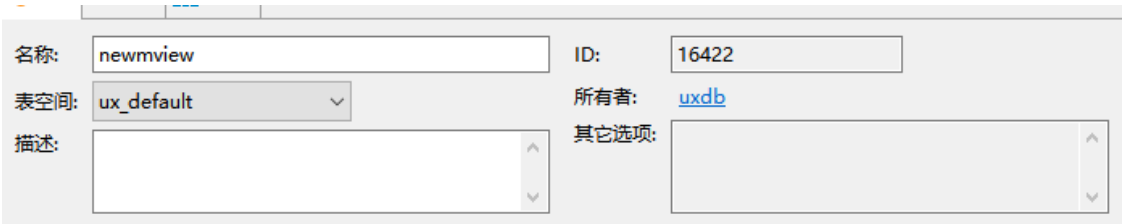
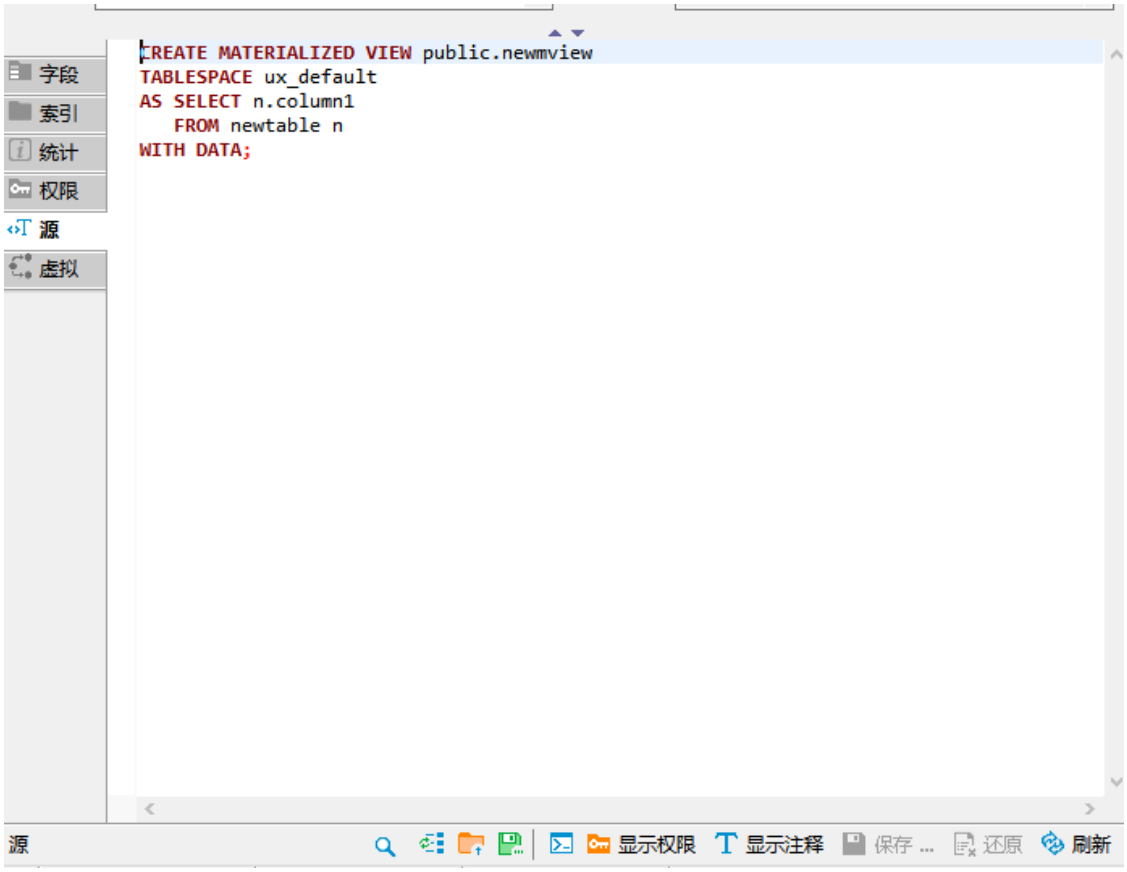


表 7.8. 物化视图的“基本属性”页说明

名称	说明
名称	待创建的或者当前查看的物化视图的名称。 在“创建物化视图”操作过程中，用户可以指定一个有效的UXDB数据库标识符作为物化视图名。同一个模式中物化视图名不能重复。
所有者	该物化视图的所有者名称，所有者应该是数据库系统的用户之一。
表空间	物化视图创建在的存储位置，用户可以通过下拉框选择。

7.3.2.9. 物化视图的“源”页

通过信息浏览区“源”对话框，用户可以查看物化视图的 DDL 语言。



7.3.2.10. 删除物化视图

通过UXDB数据库对象管理工具可视化图形界面的引导，删除视图。

1. 鼠标定位在导航树上“物化视图名称”所在节点，右键单击“物化视图名称”。

2. 在弹出式菜单中选择“删除”菜单项，或按快捷键Delete。

7.3.3. 基于SQL语句的视图管理

7.3.3.1. 查看视图

1. 通过检查系统表ux_views，用户可以确定现有视图的集合。

```
uxdb=# select viewname from ux_views;
             viewname
---------------------------------------
 ux_shadow
 ux_settings
 ux_file_settings
 ux_hba_file_rules
 ux_config
 ux_replication_origin_status
 .
 .
 .
```

2. uxsql程序的dv命令行选项可以用来列出现有视图。

```
uxdb=# \dv
      List of relations
 Schema |  Name  | Type | Owner
--------+--------+------+-------
 public | myview | view | uxdb
(1 row)
```

7.3.3.2. CREATE VIEW

- 大纲

  CREATE VIEW ...

- 描述

  CREATE VIEW定义一个查询的视图。

  详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“CREATE VIEW”。

> **注意**
>
> 如果给定了一个模式名（例如CREATE VIEW myschema.myview...），那么该视图会被创建在指定的模式中。否则，它会被创建在当前模式中。
>
> 视图的名称不能与同一模式中任何其他视图、表、序列、索引或 外部表同名。

- 示例

  在student表中创建一个查询视图，如下所示。

```
create view myview as select * from student ;
```

7.3.3.3. ALTER VIEW

- 大纲

  ALTER VIEW [IF EXISTS] name ALTER [COLUMN] column_name SET DEFAULT expression
  ALTER VIEW [IF EXISTS] name ALTER [COLUMN] column_name DROP DEFAULT
  ALTER VIEW [IF EXISTS] name OWNER TO { new_owner | CURRENT_USER | SESSION_USER }
  ALTER VIEW [IF EXISTS] name RENAME TO new_name
  ALTER VIEW [IF EXISTS] name SET SCHEMA new_schema
  ALTER VIEW [IF EXISTS] name SET (view_option_name [= view_option_value] [, ...])
  ALTER VIEW [IF EXISTS] name RESET (view_option_name [, ...])

- 描述

  ALTER VIEW更改一个视图的多种辅助属性。

  详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“ALTER VIEW”。

> **注意**
>
> 1. 使用ALTER VIEW，必须拥有该视图。
> 2. 更改一个视图的模式，必须具有新模式上的CREATE特权。
> 3. 更改拥有者，必须是新拥有角色的一个直接或者间接成员，并且该角色必须具有该视图的模式上的CREATE特权。（这些限制强制修改拥有者不能做一些通过删除和重建视图做不到的事情。但是，数据库管理员能够更改任何视图的所有权。）

7.3.3.4. DROP VIEW

- 大纲

  DROP VIEW [IF EXISTS] name [, ...] [CASCADE | RESTRICT]

- 描述

  DROP VIEW删除一个现有的视图。

  详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“DROP VIEW”。

> **注意**
>
> 视图的拥有者才有权限执行这个命令。

7.4. 索引

7.4.1. 关于索引

索引是与表相关的一种数据结构，是为了提高数据检索的性能而建立的。利用它可快速地确定指定的信息。但是，索引在总体上也增加数据库的负担，因此，创建索引要谨慎。

UXDB索引为表数据提供快速存取路径。索引适用于一范围的行查询或指定行的查询。

任何时候都可以在表上创建和删除索引。索引逻辑地和物理地独立于数据，它们的建立或删除对表没有影响。

索引可建立在一个表的一列或多列上。一旦建立，将由UXDB自动维护和使用，对用户是完全透明的。系统的优化器会根据统计信息确定是否使用索引来提高系统性能。

索引一旦创建，系统则必须维护索引与表之间的同步。因此，当一个表上存在许多索引时，修改、删除和插入操作的性能会下降。所以，一些不是很必要的索引应该删除。

索引有唯一索引和非唯一索引。唯一索引保证表中没有两行在定义索引的列上具有重复值。在唯一码上自动地定义唯一索引实施UNIQUE完整性约束。

7.4.2. 查看索引

1. 通过检查系统表ux_indexes，用户可以确定现有索引的集合。

```
uxdb=# select indexname from ux_indexes;
              indexname
-----------------------------------------------
 ux_aggregate_fnoid_index
 ux_am_name_index
 ux_am_oid_index
 ux_amop_fam_strat_index
 ux_amop_opr_fam_index
 ux_amop_oid_index
.
.
.
```

2. uxsql程序的di命令行选项也可以用来列出所有索引。

7.4.3. CREATE INDEX

- 大纲

```
CREATE [ UNIQUE ] INDEX [ CONCURRENTLY ] [ [ IF NOT EXISTS ] name ] ON [ ONLY ]
 table_name [ USING method ]
( { column_name | ( expression ) } [ COLLATE collation ] [ opclass ] [ ASC | DESC ] [ NULLS
 { FIRST | LAST } ] [, ...] )
[ INCLUDE ( column_name [, ...] ) ]
[ WITH ( storage_parameter = value [, ... ] ) ]
[ TABLESPACE tablespace_name ]
[ WHERE predicate ]
```

- 描述

使用CREATE INDEX语句可以创建索引。

详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“CREATE INDEX”。

> **注意**
>
> 1. 缺省情况下，CREATE INDEX语句创建B-树索引。

2. CREATE FULLTEXT INDEX语句创建GIN索引。
3. 使用USING method子句可以选择创建B-树索引、HASH索引或GIN索引。

- 示例

1. 在表films中的列title上创建一个B-树索引。

   CREATE UNIQUE INDEX title_idx ON films (title);

   函数索引是根据表的一个或多个列上的函数结果建立的索引。

2. 函数索引可以提高基于函数调用结果的数据的访问效率。

   #如果创建了下面的索引：
   CREATE INDEX emp_lower_ename_idx ON scott.emp (lower(ename));
   #则下面的查询就可以通过该索引提高查询速度：
   SELECT * FROM scott.emp WHERE lower(ename) = 'value';

   函数索引定义中引用的函数可以具有多个参数变量，但是它们必须是表中的列。函数索引总是单列索引（函数结果值）。

7.4.4. ALTER INDEX

- 大纲

ALTER INDEX [IF EXISTS] name RENAME TO new_name
ALTER INDEX [IF EXISTS] name SET TABLESPACE tablespace_name
ALTER INDEX name ATTACH PARTITION index_name
ALTER INDEX name DEPENDS ON EXTENSION extension_name
ALTER INDEX [IF EXISTS] name SET (storage_parameter = value [, ...])
ALTER INDEX [IF EXISTS] name RESET (storage_parameter [, ...])
ALTER INDEX [IF EXISTS] name ALTER [COLUMN] column_number
SET STATISTICS integer
ALTER INDEX ALL IN TABLESPACE name [OWNED BY role_name [, ...]]
SET TABLESPACE new_tablespace [NOWAIT]

- 描述

ALTER INDEX更改一个索引的定义。

详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“ALTER INDEX”。

注意

1. 使用ALTER TABLE可以实现ALTER INDEX操作相同的效果。实际上，ALTER INDEX只是ALTER TABLE应用在索引上的形式的别名。
2. 一个索引的拥有者不能与其基表的拥有者不同。更改基表的拥有者会自动地更改索引的拥有者。
3. 不允许更改系统目录索引的任何部分。

- 示例

重命名一个现有索引。

ALTER INDEX distributors RENAME TO suppliers;

7.4.5. REINDEX

- 大纲

  REINDEX [(VERBOSE)] { INDEX | TABLE | SCHEMA | DATABASE | SYSTEM }
  [CONCURRENTLY] name

- 描述

  当一个表经过大量的增删改操作后，表的数据在物理文件中可能存在大量碎片，从而影响访问速度。此外，当删除表的大量数据后，若不再对表执行插入操作，索引所处的段可能占用了大量并不使用的簇，从而浪费了存储空间。

  因此，可以使用重建索引来对索引的数据进行重组，使数据更加紧凑，并释放不需要的空间，从而提高访问效率和空间效率。

  UXDB提供REINDEX重建索引。

  详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“REINDEX”。

- 示例

  - 重建单个索引，如下所示。

    REINDEX INDEX my_index;

  - 重建表my_table上的所有索引，如下所示。

    REINDEX TABLE my_table;

7.4.6. DROP INDEX

- 大纲

  DROP INDEX [CONCURRENTLY] [IF EXISTS] name [, ...] [CASCADE | RESTRICT]

- 描述

  当用户不再需要该索引，或者该索引没有为针对其相关的表所发布的查询提供所期望的性能改善时。需要删除一个索引。

  UXDB提供DROP INDEX重建索引。

  详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“DROP INDEX”。

> **注意**
>
> 要想删除索引，则该索引必须包含在用户的模式中或用户必须具有DROP INDEX的数据库权限。

- 参数

表 7.9. DROP INDEX 参数说明

参数	描述
CASCADE	自动删除依赖于该索引的对象，然后删除所有依赖于那些对象的对象。
RESTRICT	如果有任何对象依赖于该索引，则拒绝删除它，是默认值。

- 示例

删除索引title_idx，如下所示。

```
DROP INDEX title_idx;
```

7.5. 存储过程/函数

7.5.1. 关于存储过程/函数

存储过程/函数是由过程化语言（PL/SQL）书写，经编译和优化后存储在数据库服务器中的过程/函数。UXDB的存储过程/函数结合了SQL的数据操作能力和过程化语言的数据处理能力，方便程序开发人员设计出更加灵活的应用系统。

存储过程/函数相当于数据库中的一个对象，用户在使用存储过程/函数时首先使用CREATE PROCEDURE/FUNCTION语句创建存储过程/函数。在其它的存储过程/函数或用户的应用程序中可以使用CALL语句执行存储过程。用户创建的函数可以和其他系统内置的函数一样使用。当存储过程/函数不再需要时，可以使用DROP PROCEDURE/FUNCTION删除存储过程/函数。

7.5.2. 基于图形界面的存储过程/函数管理

7.5.2.1. 新建存储过程

通过UXDB数据库对象管理工具可视化图形界面的引导，创建存储过程/函数。

1. 鼠标定位在导航树上存储过程或“存储过程名称”所在节点，右键单击存储过程或“存储过程名称”。

2. 在弹出式菜单中选择“新建存储过程”菜单项，新建存储过程。

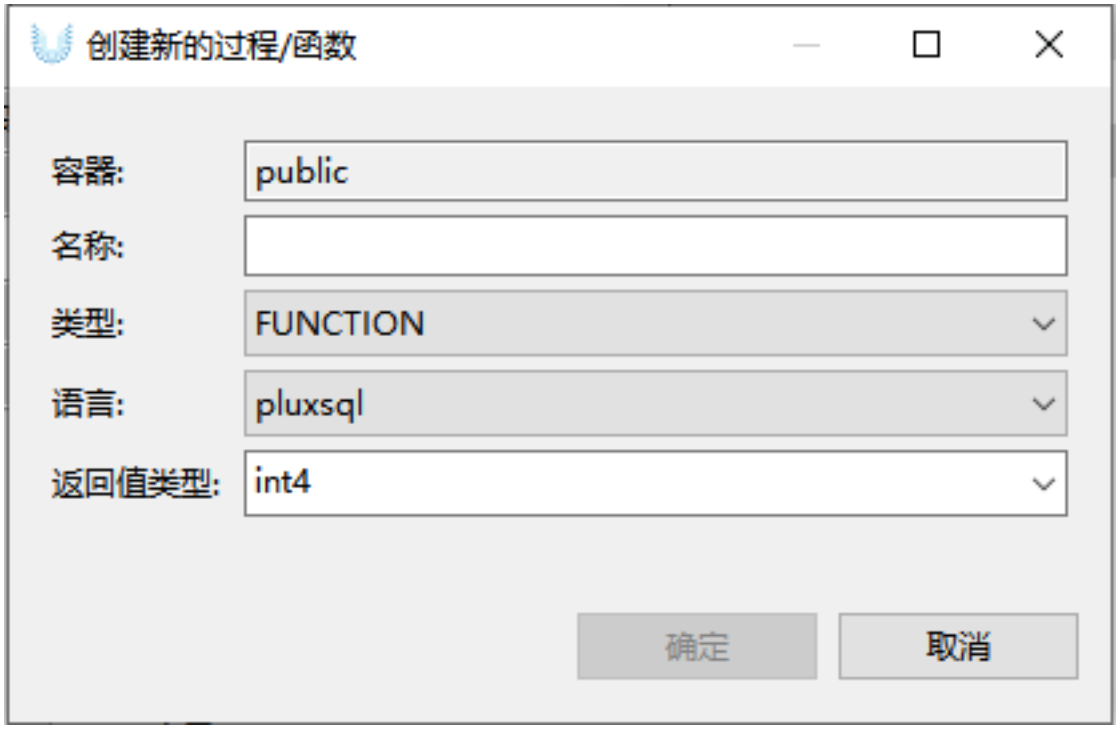


表 7.10. 过程/函数参数说明

参数	描述
容器	模式名称，默认自动填写所操作的模式。
名称	创建存储过程的标识，同模式中存储过程名称不能重复。
类型	支持PROCEDURE和FUNCTION类型选择。
语言	通过下拉框选择语言，如sql、pluxsql、c等语言。
返回值类型	存储过程/函数返回的数据类型。

7.5.2.2. “基本属性”页

基本属性页，可以定义或查看存储过程的基本属性。

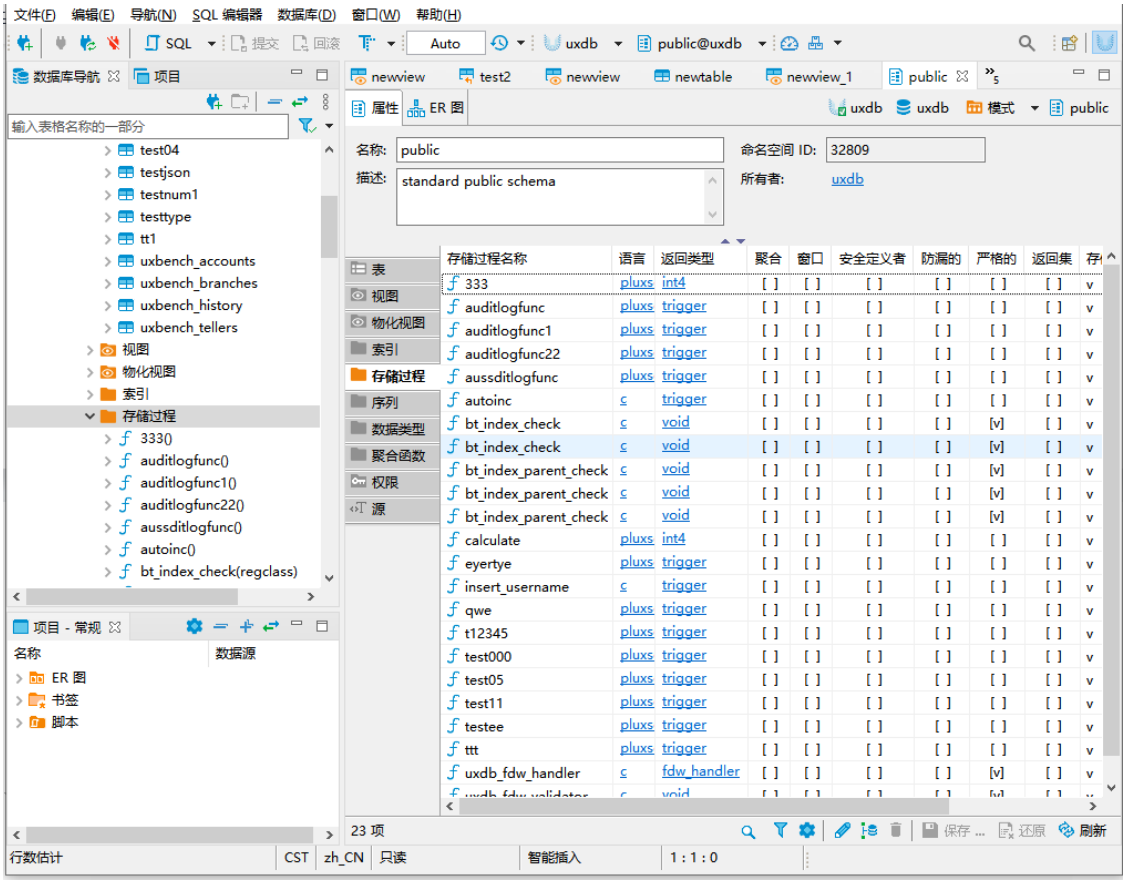


表 7.11. 存储过程“基本属性”页参数说明

参数	说明
名称	待创建的或者当前查看的“存储过程”的名称。在“新建存储过程”操作过程中，可以指定一个有效的UXDB数据库标识符作为存储过程名。同模式中存储过程名称不能重复。
所有者	该存储过程的所有者名称，所有者应该是数据库系统的用户之一。

7.5.2.3. “权限”页

权限属性页，可以定义或查看存储过程/函数的权限。切换“用户”/“角色”时，权限将随之更新。

7.5.2.4. “源”页

通过信息浏览区对话框，用户可以查看存储过程/函数的DDL语言。

7.5.2.5. 删除存储过程/函数

通过UXDB数据库对象管理工具可视化图形界面的引导，删除存储过程/函数。

1. 鼠标定位在导航树上“存储过程”所在节点，右键单击“存储过程名称”。
2. 在弹出式菜单中选择“删除”菜单项，或按快捷键Delete。

7.5.3. 基于SQL语句的存储过程/函数管理

7.5.3.1. 查看存储过程/函数

- 通过检查系统表ux_proc，用户可以确定有关函数、过程、聚集函数以及窗口函数的信息。
- uxsql程序的df命令行选项也可以用来列出（只包括聚合/常规/程序/触发器/窗口）函数。
- uxsql程序的dL命令行选项也可以用来列出所有过程语言。

7.5.3.2. CREATE PROCEDURE

- 大纲

  CREATE OR REPLACE PROCEDURE

- 描述

  创建一个新过程或者替换一个已有的定义。

  详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“CREATE PROCEDURE”。

> **注意**
>
> 1. 用户必须具有所使用的语言上的USAGE特权。
> 2. 如果这个命令中包括了一个方案名称，则该存储过程将被创建在该方案中。否则存储过程将被创建在当前的方案中。新存储过程的名称不能匹配同一方案中具有相同输入参数类型的任何现有存储过程或函数。但是，具有不同参数类型的存储过程和函数可以共享同一个名称（这被称为重载）。
> 3. 替换一个已有存储过程的当前定义，请使用CREATE OR REPLACE。
> 4. PROCEDURE。但不能采用这种方式更改存储过程的名称或者参数类型（如果尝试这样做，实际上会创建一个新的、不同的存储过程）。
> 5. 当CREATE OR REPLACE PROCEDURE被用来替换一个现有的存储过程时，该存储过程的拥有关系和权限保持不变。所有其他的存储过程属性会被赋予这个命令中指定的或者暗示的值。必须拥有（包括成为拥有角色的成员）该存储过程才能替换它。
> 6. 创建存储过程的用户将成为该存储过程的拥有者。

- 示例

  简单存储过程示例，如下所示。

```
uxdb=# create table tbl(id int);
CREATE TABLE
uxdb=# CREATE PROCEDURE insert_data(a integer, b integer)
LANGUAGE SQL
AS $$
```

```
INSERT INTO tbl VALUES (a);
INSERT INTO tbl VALUES (b);
$$;
CREATE PROCEDURE
uxdb=# CALL insert_data(1, 2);
CALL
uxdb=# select * from tbl ;
 id
----
  1
  2
(2 rows)


CREATE OR REPLACE PROCEDURE increment1(i INTEGER)
AS $$
BEGIN
RAISE NOTICE '%', i + 1;
END;
$$ LANGUAGE pluxsql;
```

7.5.3.3. ALTER PROCEDURE

- 大纲

  ALTER PROCEDURE

- 描述

  UXDB使用ALTER PROCEDURE更改一个过程的定义。

  详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“ALTER PROCEDURE”。

  > **注意**
  >
  > 1. 必须拥有该过程。
  > 2. 更改一个存储过程的方案，必须有新方案上的CREATE特权。
  > 3. 要更改拥有者，你还必须是新拥有角色的直接或间接成员，并且那个角色在该存储过程的方案上拥有CREATE特权（这些限制强制更新拥有者无法做到通过删除和重建该存储过程无法做到的事情。超级用户总是能够更改任何存储过程的拥有关系）。

- 示例

  - 重命名具有两个integer类型参数的过程insert_data为insert_record。

    ```
    ALTER PROCEDURE insert_data(integer, integer) RENAME TO insert_record;
    ```

  - 将具有两个integer类型参数的过程insert_data的拥有者改为joe。

    ```
    ALTER PROCEDURE insert_data(integer, integer) OWNER TO joe;
    ```

- 将具有两个integer类型参数的过程insert_data的方案改为accounting。

ALTER PROCEDURE insert_data(integer, integer) SET SCHEMA accounting;

7.5.3.4. DROP PROCEDURE

- 大纲

DROP PROCEDURE

- 描述

删除一个现有存储过程的定义。

详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“DROP PROCEDURE”。

> **注意**
>
> 该存储过程的拥有者才有权限执行此命令。 该存储过程的参数类型必须指定，因为可能存在多个不同的存储过程具有相同名称和不同参数列表。

- 示例

DROP PROCEDURE do_db_maintenance();

7.5.3.5. CREATE FUNCTION

- 大纲

CREATE OR REPLACE FUNCTION

- 描述

创建一个新函数或者替换一个现有的函数。

详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“CREATE FUNCTION”。

> **注意**
>
> 1. 要定义一个函数，用户必须具有该语言上的USAGE特权。
> 2. 创建该函数的用户将成为该函数的拥有者。

- 示例

1. 创建一个函数，如下所示。

```
CREATE FUNCTION add(integer, integer) RETURNS integer
AS 'select $1 + $2;'
LANGUAGE SQL
IMMUTABLE
```

RETURNS NULL ON NULL INPUT;

2. 调用一个函数，如下所示。

select add(1,1);

7.5.3.6. ALTER FUNCTION

- 大纲

ALTER FUNCTION

- 描述

更改一个函数的定义。

详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“ALTER FUNCTION”。

> **注意**
>
> 1. 函数的拥有者才有权限执行此命令。
> 2. 更改一个函数的模式，必须具有新模式上的CREATE特权。
> 3. 更改拥有者，必须是新拥有角色的一个直接或者间接成员，并且该角色必须具有在该函数的模式上的CREATE特权（这些限制强制修改拥有者不能做一些通过删除和重建该函数做不到的事情。但是，数据库管理员能够更改任何函数的所有权）。

- 示例

1. 把用于类型integer的函数sqrt 重命名为square_root，如下所示。

ALTER FUNCTION sqrt(integer) RENAME TO square_root;

2. 把用于类型integer的函数sqrt 的拥有者改为joe，如下所示。

ALTER FUNCTION sqrt(integer) OWNER TO joe;

3. 把用于类型integer的函数sqrt 的模式改为maths，如下所示。

ALTER FUNCTION sqrt(integer) SET SCHEMA maths;

7.5.3.7. DROP FUNCTION

- 大纲

DROP FUNCTION

- 描述

删除一个已有函数的定义。

详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“DROP FUNCTION”。

> **注意**
>
> 1. 要执行这个命令用户必须是该函数的拥有者。
> 2. 该函数的参数类型必须被指定，因为多个不同的函数可能会具有相同的函数名和不同的参数列表。

- 示例

  1. 删除平方根函数，如下所示。

     DROP FUNCTION sqrt(integer);

  2. 在一个命令中删除多个函数，如下所示。

     DROP FUNCTION sqrt(integer), sqrt(bigint);

7.6. 触发器

7.6.1. 关于触发器

触发器是一种特殊的存储过程。它在插入、删除或修改特定表和视图中的数据时触发执行，它比数据库本身标准的功能有更精细和更复杂的数据控制能力。

触发器是一种特殊类型的存储过程，不由用户直接调用。创建触发器时需要对其进行定义，以便在对特定表或列作特定类型的数据修改时执行。

与存储过程相同，触发器也存储在数据库中，使用SQL和PL/SQL语句编写。存储过程和触发器最主要的区别在于它们运行的方式不同。存储过程由用户、应用程序或触发器显式调用，而触发器则是当触发事件发生时由系统触发执行。

触发器按照所触发动作的间隔尺寸可以分为语句级触发器和行级触发器，CREATE TRIGGER语句的FOR EACH子句定义了触发动作的间隔尺寸，它既可以是FOR EACH STATEMENT，也可以是FOR EACH ROW。

7.6.2. 表级触发器

表级触发器是依赖于已存在的表、对特定表的相关操作会引动发触发器调用相应的触发器函数。

7.6.2.1. CREATE TRIGGER

- 描述

  CREATE TRIGGER创建一个新触发器。该触发器将被关联到指定的表、视图或者外部表并且在表上发生特定操作时将执行指定的函数function_name。

- 示例

  只要表accounts的一行即将要被更新时会执行函数check_account_update，如下所示。

CREATE TRIGGER check_update
BEFORE UPDATE ON accounts
FOR EACH ROW
EXECUTE FUNCTION check_account_update();

7.6.2.2. ALTER TRIGGER

- 描述

  ALTER TRIGGER更改一个现有触发器的属性。

  RENAME子句更改给定触发器的名称而不更改其定义。

  DEPENDS ON EXTENSION子句把该触发器标记为依赖于一个扩展，这样如果扩展被删除，该触发器也会被自动删除。

  要更改一个触发器的属性，必须拥有该触发器所作用的表。

- 示例

  重命名一个现有的触发器，如下所示。

  ALTER TRIGGER emp_stamp ON emp RENAME TO emp_track_chgs;

7.6.2.3. DROP TRIGGER

- 描述

  DROP TRIGGER删除一个现有的触发器定义。要执行这个命令，当前用户必须是触发器基表的拥有者。

- 示例

  销毁表films上的触发器if_dist_exists，如下所示。

  DROP TRIGGER if_dist_exists ON films;

7.6.2.4. 表级触发器的禁止和开启

表级触发器的禁止或者开启功能由ALTER TABLE提供，而非ALTER TRIGGER，因为ALTER TRIGGER无法针对性启用或者禁止一个表上的触发器。

详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“ALTER TABLE”中的DISABLE TRIGGER选项和ENABLE TRIGGER选项。

7.6.3. 事件触发器

UXDB中的事件触发器和表级触发器（依赖于表上并且只捕捉DML事件）不同，事件触发器对数据库来说是全局的，并且可以捕捉DDL事件。

和表级触发器一样，可以用任何事件触发器支持的过程语言编写事件触发器，目前UXDB不支持用纯SQL编写事件触发器。

只要一个与事件触发器相关的事件在事件触发器所在的数据库中发生，该事件触发器就会被触发。UXDB当前支持的事件有ddl_command_start、ddl_command_end、table_rewrite和sql_drop。

7.6.3.1. CREATE EVENT TRIGGER

- 描述

  CREATE EVENT TRIGGER创建一个新的事件触发器。只要指定的事件发生并且与该触发器相关的WHEN条件（如果有）被满足，该触发器的函数将被执行。创建事件触发器的用户会成为它的拥有者。

  只有数据库管理员能创建事件触发器。

  在单用户模式（见uxdb）中事件触发器被禁用。如果一个错误的事件触发器禁用了数据库让你甚至无法删除它，可以重启到单用户模式，这样就能删除它。

- 示例

  禁止执行任何DDL命令，如下所示。

```
CREATE OR REPLACE FUNCTION abort_any_command()
RETURNS event_trigger
LANGUAGE plsql
AS $$
BEGIN
RAISE EXCEPTION 'command % is disabled', tg_tag;
END;
$$;
CREATE EVENT TRIGGER abort_ddl ON ddl_command_start
EXECUTE FUNCTION abort_any_command();
```

7.6.3.2. ALTER EVENT TRIGGER

- 大纲

```
ALTER EVENT TRIGGER name DISABLE
ALTER EVENT TRIGGER name ENABLE [ REPLICA | ALWAYS ]
ALTER EVENT TRIGGER name OWNER TO { new_owner | CURRENT_USER |
 SESSION_USER }
ALTER EVENT TRIGGER name RENAME TO new_name
```

- 描述

  ALTER EVENT TRIGGER更改一个现有事件触发器的属性。

  详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“ALTER EVENT TRIGGER”。

> **注意**
>
> 只有数据库管理员有权限更改一个时间触发器。

7.6.3.3. DROP EVENT TRIGGER

- 描述

  DROP EVENT TRIGGER删除一个已有的事件触发器。要执行这个命令，当前用户必须是事件触发器的拥有者。

- 示例

  销毁触发器snitch，如下所示。

  DROP EVENT TRIGGER snitch;

7.6.3.4. 事件触发器使用示例

这里是一个用 C 编写的事件触发器函数的简单例子。

函数noddl在每一次被调用时抛出一个异常。 事件触发器定义把该函数和 ddl_command_start事件关联在了一起。其效果就是所有 DDL 命令都被阻止运行。

1. 编写触发器函数的源代码。

```
#include "postgres.h"
#include "commands/event_trigger.h"

PG_MODULE_MAGIC;
PG_FUNCTION_INFO_V1(noddl);
Datum
noddl(PG_FUNCTION_ARGS)
{
  EventTriggerData *trigdata;

  if (!CALLED_AS_EVENT_TRIGGER(fcinfo))  /* internal error */
    elog(ERROR, "not fired by event trigger manager");

  trigdata = (EventTriggerData *) fcinfo->context;

  ereport(ERROR,
    (errcode(ERRCODE_INSUFFICIENT_PRIVILEGE),
        errmsg("command \"%s\" denied", trigdata->tag)));

  PG_RETURN_NULL();
}
```

2. 在编译了源代码后，声明函数和触发器。

   CREATE FUNCTION noddl() RETURNS event_trigger
   AS 'noddl' LANGUAGE C;

3. 创建触发器。

   CREATE EVENT TRIGGER noddl ON ddl_command_start

```
EXECUTE FUNCTION noddl();
```

4. 执行触发器的操作。

```
=# \dy
                List of event triggers
 Name |       Event       | Owner | Enabled | Function | Tags
------+-------------------+-------+---------+----------+------
 noddl | ddl_command_start | dim   | enabled | noddl    |
(1 row)

=# CREATE TABLE foo(id serial);
ERROR:  command "CREATE TABLE" denied
```

在这种情况下，为了在需要时能运行某些 DDL 命令，你必须删除该事件触发器 或者禁用它。只在一个事务期间禁用该触发器会比较方便：

```
BEGIN;
ALTER EVENT TRIGGER noddl DISABLE;
CREATE TABLE foo (id serial);
ALTER EVENT TRIGGER noddl ENABLE;
COMMIT;
```

7.6.4. 基于图形界面的表级触发器管理

7.6.4.1. 新建触发器

通过UXDB数据库对象管理工具可视化图形界面的引导，创建触发器。

1. 鼠标定位在导航树上触发器或“触发器名称”所在节点，右键单击触发器或“触发器名称”。
2. 在弹出式菜单中选择“新建触发器”菜单项，新建触发器。
3. 在新建触发器窗口，填写触发器的名称及通过下拉框选择触发函数。
4. 单击窗口的确定按钮，跳转到“源”界面，单击下方的保存按钮，将弹出一个SQL预览窗口，单击执行即可实现触发器的创建。

7.6.4.2. “基本属性”页

基本属性页，可以定义或查看触发器的基本属性。

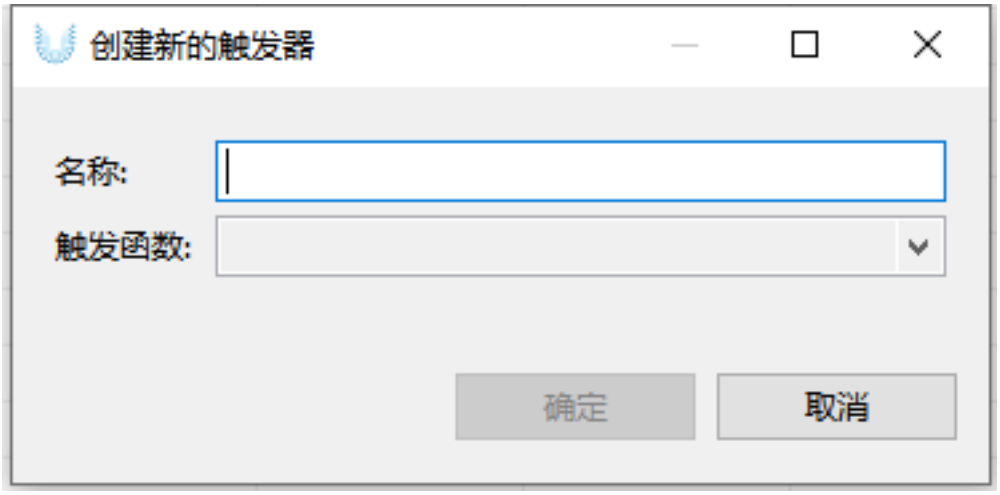


表　7.12.　表级触发器“基本属性”页参数说明

参数	说明
触发器名称	待创建的或者当前查看的触发器的名称。 在“新建触发器”操作过程中，用户可以指定一个有效的UXDB数据库标识符作为触发器名。 触发器名不能重复。
触发函数	触发函数为创建的存储过程，一般在使用触发器时，优先创建所需的存储过程，在创建触发器时，通过触发函数的下拉框选项选择匹配的存储过程函数即可。

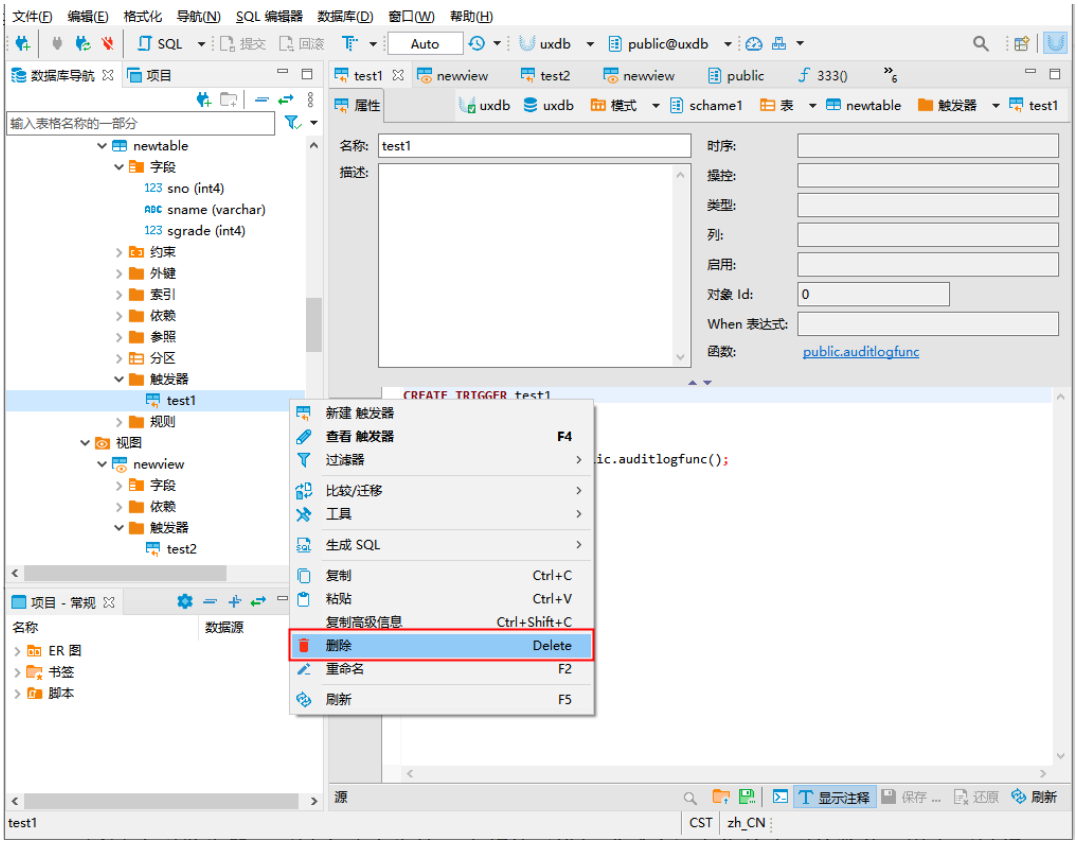
7.6.4.3.　“源”页

通过信息浏览区源对话框，用户可以查看触发器的DDL语言。

7.6.4.4.　删除触发器

通过UXDB数据库对象管理工具可视化图形界面的引导，删除触发器。

1. 鼠标定位在导航树上“触发器”所在节点，右键单击“触发器名称”。
2. 在弹出式菜单中选择“删除”菜单项，或按快捷键Delete。



7.7. 序列管理

7.7.1. 序列发生器

通过序列发生器，可以自动产生按一定规律（增加或减小）变化的序列的数值。用户可以根据需要定义每次序列变化的步长，序列能达到的极限值，当达到极限值时是否再从初始值重新开始。一旦定义了序列发生器，用户可取出序列值，取出序列值后，序列发生器自动变为下一序列值。序列发生器在很大程度上方便了用户的操作。不用对序列发生器进行维护，数据库会自动完成序列值的变化。序列发生器还可以与其它数据库对象结合使用。如：将某一字段的缺省值与某一序列发生器相关联，从而实现自动增长序列字段的功能。序列作为一种独立的模式对象存在，它的功能与自增长数据类型相同。可以在表数据的默认值中利用序列操作函数对序列进行使用。

7.7.2. 基于图形界面的序列管理

7.7.2.1. 新建序列

通过UXDB数据库对象管理工具可视化图形界面的引导，创建序列。

1. 鼠标定位在导航树上序列或“序列名称”所在节点，右键单击序列或“序列名称”。
2. 在弹出式菜单中选择“新建序列”菜单项，新建序列。

7.7.2.2. “基本属性”页

基本属性页，可以定义或查看序列的基本属性。

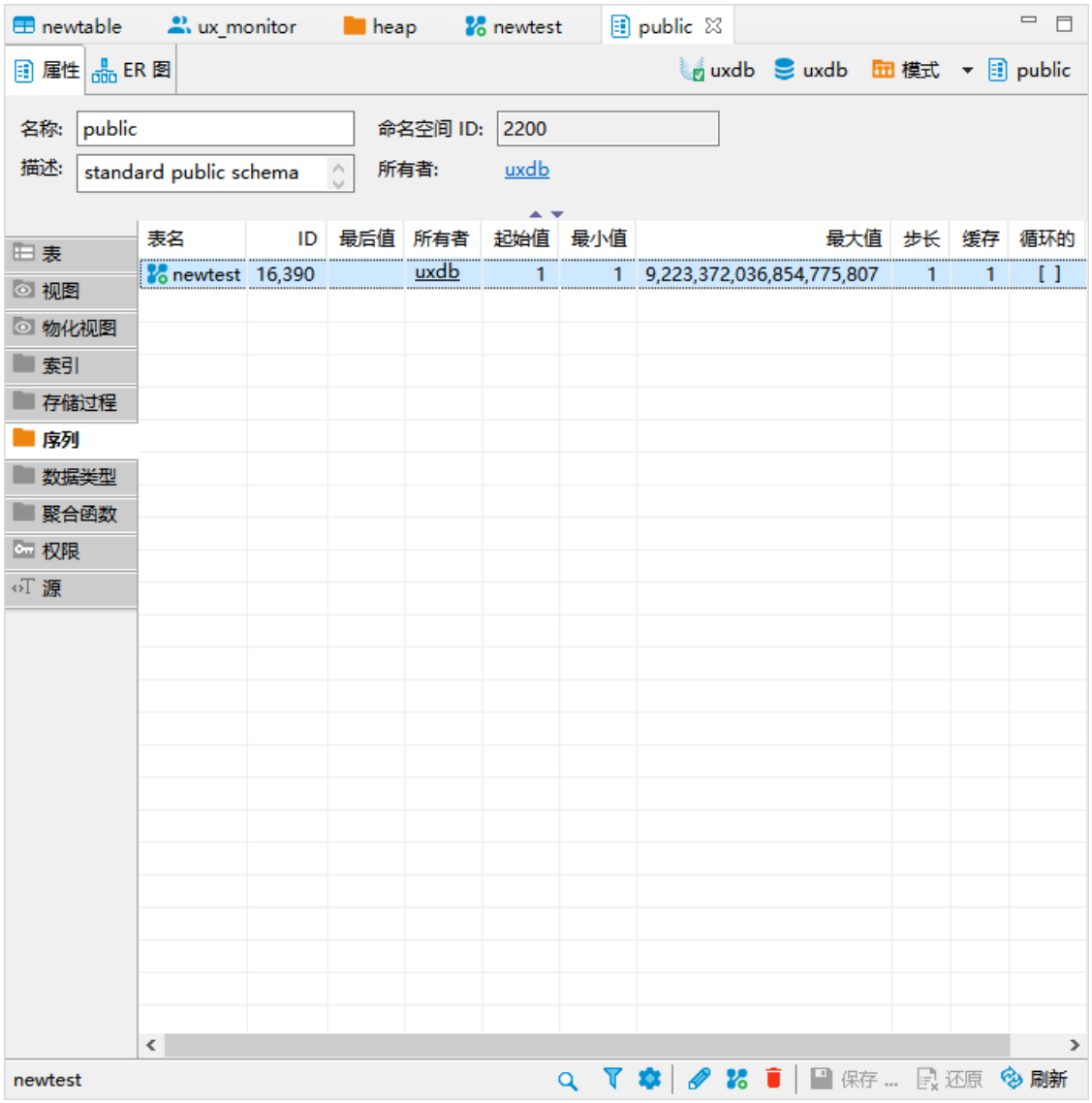


表　7.13.　序列管理“基本属性”页参数说明

参数	说明
名称	待创建的或者当前查看的序列的名称。在“新建序列”操作过程中，可以指定一个有效的UXDB数据库标识符作为序列名。同一模式下序列名不能重复。
所有者	该序列的所有者名称，所有者应该是数据库系统的用户之一。
起始值	序列起始值，默认为1。
步长	序列每次递增的数值。创建序列时，默认该值为1。用户可以自定义。
最小值	序列可允许的最小值。创建序列时，默认该字段最初为“1”。

参数	说明
最大值	序列可允许的最大值。创建序列时，默认为999999。
缓存	默认为1。由数据库预分配并存储的值的数目，序列号可以存放在内存中的已预分配值获取。使用完已预分配值后，系统自动再次获取预分配值。
循环	默认取消。如果选中，表示序列号达到最大值时可复位并继续下去。

7.7.2.3. 删除序列

通过UXDB数据库对象管理工具可视化图形界面的引导，删除序列。

1. 鼠标定位在导航树上“序列”所在节点，右键单击“序列名称”。
2. 在弹出式菜单中选择“删除”菜单项，或按快捷键Delete。

7.7.3. SQL语句

7.7.3.1. CREATE SEQUENCE

- 大纲

  CREATE SEQUENCE

- 描述

  创建一个新的序列数发生器。

  详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“CREATE SEQUENCE”。

- 示例

  创建一个称作serial的上升序列，从 101 开始，如下所示。

  CREATE SEQUENCE serial START 101;

7.7.3.2. ALTER SEQUENCE

- 大纲

  ALTER SEQUENCE

- 描述

  更改一个现有序列的参数。

  详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“ALTER SEQUENCE”。

> 注意
>
> 1. 使用ALTER SEQUENCE，必须拥有该序列。

2. 更改一个序列的模式，必须拥有新模式上的CREATE特权。
3. 更改拥有者，必须是新拥有角色的一个直接或者间接成员，并且该角色必须具有该域的模式上的CREATE特权（这些限制强制修改拥有者不能做一些通过删除和重建该序列做不到的事情。但是，数据库管理员能够更改任何序列的所有权。）。

7.7.3.3. DROP SEQUENCE

- 大纲

  DROP SEQUENCE

- 描述

  删除序列。

  详细信息请参见《优炫数据库 V2.1 参考手册》“SQL命令”章节中的“DROP SEQUENCE”。

  注意

  一个序列只能被其拥有者或数据库管理员删除。

- 示例

  删除序列serial，如下所示。

  DROP SEQUENCE serial;

第 8 章 事务管理

8.1. 数据库事务

同大多数的商业数据库一样，UXDB数据库也引入了事务的概念。本章介绍UXDB中的事务以及事务的管理。

8.1.1. 事务的概念

数据库中的事务一个独立的逻辑单元。它包括了一个或多个 SQL 语句。数据库工作的基本单位是事务，事务是不能被拆分的。一个事务执行的结果只可能有两个：事务内的语句都成功执行，事务执行成功，即事务提交；或事务执行失败，本事务不对数据造成任何影响，即事务回滚。为了更形象的描述事务，举一个简单的例子：银行数据库如何处理客户转账。一个客户有这样的要求：把一笔钱从A帐户转到B帐户。这个要求可以拆分为三个独立的部分：

- 帐户A划走这笔钱
- 帐户B增加这笔钱
- 记录这笔交易

整个交易过程数据库确保最终达到如下两种状态中的一个：

- 三条语句都成功执行，达到转账目的并且做交易记录。 整个事务提交，数据应用到数据库。
- 其中一条语句执行过程发生错误，比如A帐户资金不足，B帐户不存在，或者在整个交易过程中发生硬件故障，数据库停机。 整个事务回滚，执行成功的语句被撤销， 执行失败的语句不对数据造成影响，还未执行的语句不再执行。数据回滚到此事务未开始的状态。无论什么原因,整个业务逻辑不允许出现诸如帐户A上的钱被划走，而却没有到达帐户 B，或者这个交易没有被记录这样的中间结果，整个交易必须是原子性的，要么整个操作成功，要么整个操作失败。数据库的事务概念正是吻合了此要求。

一条SQL语句成功执行标志着：

- SQL语句被解析，语法正确。
- SQL语句是有效的，语义正确。
- 一些数据可能被更改，但是这些更改不是永久性的，在此语句所在事务提交前，数据的更改并没有在整个数据库生效。

对于事务的执行结果，只能有提交和回滚。

- 提交操作意味着：事务内的语句对数据的改变被数据库接受。这将造成数据对于整个数据库发生永久性的改变。
- 回滚操作意味着：这个事务内所有对数据造成的更改全部被撤销。

一条SQL语句执行失败会回滚整个事务。

8.1.2. 事务的特性

为了保证数据库中的数据一致性，确保UXDB能够在并发访问和系统发生故障时对数据进行维护，事务作为数据库工作的基本单位，具有下列四个特性，称为事务的ACID特性。

- 原子性（Atomicity）

  一个事务对数据库的所有操作，是一个不可分割的工作单元。这些操作要么全部执行，要么什么也不做。

- 一致性（Consistency）

  一个事务独立执行的结果应保持数据库的一致性，即数据不会因为事务的执行而遭受破坏。例如事务的概念里银行转账的例子，保证了数据的一致性。

- 隔离性（Isolation）

  在多个事务并发执行时，系统应保证与这些事务先后单独执行时的结果一样，此时称事务达到了隔离性的要求，也就是在多个并发事务执行时，保证执行结果是正确的，如同单用户环境一样。隔离性是由KingbaseES 的并发控制子系统实现的。

- 持久性（Durability）

  一个事务一旦完成全部操作后，它对数据库的所有更新应永久地反映在数据库中。即使以后系统发生故障，也应保留这个事务执行的痕迹。

事务是恢复和并发控制的基本单位。UXDB能够保证事务的ACID特性不被破坏，分为以下几种情况：

- 多个事务并发运行时，不同事务的操作交叉执行。UXDB保证多个事务的交叉运行不影响这些事务的原子性。
- 事务在运行过程中被强行终止。UXDB保证被强行终止的事务对数据库和其他事务没有任何影响。

8.1.3. 数据库对事务的管理

对于事务的管理，UXDB支持事务的隐式和显式提交，即自动提交和非自动提交。

- 自动提交事务

  默认一个SQL语句为一个事务，当此语句执行成功后，由系统隐式提交。数据更改被数据库接受。当SQL执行失败，自然视做回滚，不对数据造成影响。UXDB默认提交事务方式为自动提交。

- 非自动提交事务

  事务的开始为第一个SQL语句。事务的结束必须用户显式的使用COMMIT语句提交整个事务或者使用ROLLBACK语句回滚整个事务。如果客户端连接断开，那么未提交的事务做回滚操作。

  当用户想自己控制事务的执行，可以显式的用BEGIN语句开始一个事务，然后在事务结束时应该使用COMMIT或ROLLBACK语句显式提交。

  接口中也有对事务提交方式的控制，配置相应的配置文件可以使会话处于自动提交或非自动提交模式。

一个事务的提交和回滚对数据库实例造成的影响如下所示。

- 一个事务提交对数据库实例造成的影响
  - 所有数据库更改对整个数据库生效。
  - 保证此事务产生的日志写到磁盘。

- 释放所有此事务进行过程中申请的锁。
- 数据库标记此事务结束。

- 一个事务回滚对数据库实例造成的影响
  - 撤销所有对数据的更改到此事务开始之前的状态。
  - 释放所有此事务进行过程中申请的锁。
  - 数据库标记此事务结束。

8.2. 数据的一致性和并发控制

在实际应用中数据库资源对用户是共享的，不同的用户在不同或相同的时刻使用数据库，这就是数据库的并发操作。在多用户数据库环境中，并发操作提高了数据库资源的使用效率，但若对这种并发操作不加限制就会破坏数据的完整性和一致性问题。

由于数据库工作的基本单位是事务，针对事务并发操作中的各类问题，SQL标准提出了事务的隔离级别概念。

8.2.1. 事务的隔离级别

并发操作虽然可以改善系统的资源利用率和短事务的响应时间，但是在运行中必须对并发事务加以控制，否则会引发一些问题，SQL92标准对这些情况进行了分类，以及提出了解决办法。

由于一个事务可能包含多个SQL语句，SQL语句是顺序执行的。T1和T2两个事务在一个数据库上并发执行，可能有如下类型的问题：

1. 读“脏”数据（Dirty Read）

   事务T1更新了一个数据，并且在T1提交或者回滚前，另外一个事务T2读取了那个数据。如果T1这个时候回滚，那么T2就读取了一个未提交的虚假的值，即“脏”数据。

2. 不可重复读（Non-Repeatable 或 Fuzzy Read）

   事务T1读取了一个数据。这个时候另外一个事务T2修改或者删除了这个数据并且提交了。如果T1尝试再读取一次这个数据，就会发现再次读到的数据与之前不一致或不存在。

3. 幻象读（Phantom Read）

   事务T1读取了若干个满足某个查询条件的数据。这个时候事务T2创建了一个数据恰好也满足T1的这个查询条件。如果这个时候T1再次根据这个查询条件进行读取，它会发现会多出了部分数据。

很显然，高并发度带来了数据的不一致，这些情况一些是用户不愿意见到，一些则是无法接受的。用户需要在高并发带来的高性能和数据的一致性之间做取舍。SQL92标准提出了事务隔离级别，来解决上面的问题。

表 8.1. 事务的隔离级别

隔离级别	读脏数据	不可重复读	读幻象数据
读未提交(Read uncommitted)	可能	可能	可能

隔离级别	读脏数据	不可重复读	读幻象数据
读已提交(Read committed)	不可能	可能	可能
可重复的读 (Repeatable read)	不可能	不可能	可能
可串行化 (Serializable)	不可能	不可能	不可能

四个隔离级别从上到下对事务执行的并发程度进行了不同程度的限制，更加严格的限制在带来更好的数据一致性的同时，也会损失更多并发带来的高性能。

实际使用中，隔离级别并不是越高就越好，大多数情况下应用并不需要很高的数据一致性。相反的，在多用户环境下，更强调的是并发度。所以综合考虑选取一个折中的办法往往能达到最优的效果。

- 读已提交 (Read committed)

  READ COMMITTED 是UXDB默认的事务隔离级别。在运行在该隔离级别的事务中，查询语句只能看到该查询开始执行之前提交的数据，而不会看到任何未提交的数据或查询执行期间并发的其它事务提交的数据，但是该查询可以看到本事务中查询之前执行的数据更新。

  在READ COMMITTED隔离级别下，对于事务T1中的更新和删除语句，与查询语句相同，只能看到语句开始执行时提交的数据行，但是，这些数据行可能已经被同时并发的另一个事务B更新，在这种情况下，事务T1 将等待事务T2回滚或提交。如果事务 T2 回滚，则事务T1在原来的数据上继续更新，如果事务T2提交了，将分两种情况，1）：如果事务T2删除了该行，则事务 T1 忽略该行，2）：如果事务T2对该行进行了更新，则事务T1 判断该行的新值是否还满足条件，如果满足条件，则事务T1在新数据行上进行更新，如果不满足条件，则忽略该行。

  在READ COMMITTED隔离级别下，同一个事务的不同查询看到的可能是不同的数据，因此会出现不可重复读的问题。这种隔离级别对于大多数的应用已经能够满足要求，但有些应用需要提供更加严格的数据一致性。

  为了避免同样的查询得到不同的结果，在READ COMMITTED隔离级别下，应用程序在同一事务内应该尽量避免重复的查询。

- 可串行化(Serializable)

  SERIALIZABLE提供了更加严格的事务隔离级别，在该事务隔离级别下，事务并发执行，其结果与某个串行执行顺序的结果完全相同。

  在运行在 SERIALIZABLE 隔离级别的事务中，查询语句只能看到该事务开始执行之前已经提交的数据，而不会看到任何未提交的数据或事务执行期间，并发的其它事务提交的数据，但是该查询可以看到本事务中查询之前执行的数据更新。

  在SERIALIZABLE 隔离级别下，对于事务 T1 中更新和删除语句，与查询语句相同，只能看到事务开始执行时提交的数据行，但是，这些数据行可能已经被同时并发的另一个事务 T2 更新（事务 T2 开始的时间在事务 T1 前），在这种情况下，事务T1将等待事务T2回滚或提交。如果事务T2回滚，则事务T1在原来的数据上继续更新，如果事务T2提交了，则事务T1就会出错回滚，因为一个串行事务不能修改在该事务开始后被其它事务修改的数据。

  SERIALIZABLE隔离级别可以确保每个事务看到一个完全一致的数据，但是当出现并发更新时，后更新的事务必须回滚，只能在之前的更新提交后再重新执行。

  在SERIALIZABLE隔离级别下，应用程序如果碰到更新语句回滚，应该重执行更新语句所在的整个事务。

8.2.2. 并发控制的实现

UXDB在多用户环境的并发访问，是通过多版本并发控制和封锁机制共同实现的。

8.2.2.1. 多版本并发控制

多版本并发控制，即MVCC（Multi-version concurrency control）。对于表里的某个一行R，多版本并发控制通过每个写R的操作建立R的一个新版本，当执行一个读R的操作时，针对操作所在的事务，系统选择R的一个正确版本来进行操作。多版本并发控制的一个显著好处是读操作和写操作并不互斥。

例如：如果用户A希望读取表T上的所有数据，而此时用户B正在更新表T上的某行数据r。对于多版本并发控制系统，B用户的更新操作对于r生成新的版本r(i)进行修改,而不是直接修改修改数据。用户A读操作同样获取一个版本的数据r(j)。所以利用区分版本的方式，用户A不必等待B的完成。

这里的版本概念也称作快照（snapshot）。一个快照就是在某个时刻的数据库映像。比如，当事务开始时，系统为对应的时间点的数据生成一个快照。在这个快照中包含的数据十分关键，可能包含了那个时刻未提交的事务所产生的数据，还可能包含了那个时刻已经提交的事务产生的数据。换句话说一些数据在这个快照中是可见的，另一些是不可见的。毫无疑问，这影响到了数据的一致性。

可以用快照对数据的可见性，定义数据库的隔离级别。

- 读已提交 (Read committed)

  当一个事务隔离级别为读已提交时，在事务里每个SQL语句开始时，按规则生成一个快照，对于还没有提交的数据，认为不可见，否则可见。

- 可串行化 (Serializable)

  当一个事务隔离级别为可串行化时，事务开始时生成一个快照，如果在这个事务开始时还没有提交的数据，认为它们不可见，否则可见。

多版本控制带来的巨大好处是最大可能的支持并发的查询操作，即使修改数据时也不会阻塞读。

8.2.2.2. 锁

除了多版本并发控制，锁机制也是实现并发访问的必不可少的手段。

表 8.2. 锁

编号	锁模式（LOCKMODE）	对应操作	与之冲突的模式
1	AccessShareLock	SELECT	8
2	RowShareLock	SELECT FOR UPDATE	7,8
3	RowExclusiveLock	INSERT, UPDATE, DELETE	5,6,7,8
4	ShareUpdateExclusiveLock	VACUUM	4,5,6,7,8
5	ShareLock	CREATE INDEX	3,4,6,7,8
6	ShareRowExclusiveLock	like ExclusiveLock, but allows ROW SHARE	3,4,5,6,7,8
7	ExclusiveLock	blocks ROW SHARE	2,3,4,5,6,7,8
8	AccessExclusiveLock	DROP TABLE, ALTER TABLE	全部

UXDB使用锁控制用户对数据库对象，包括:表、页面、元组、事务、非表对象（函数，触发器等）的并发访问。

如果一个对象被加上了一种模式的锁，那么其他用户如果想加上和此模式相冲突的锁，就会发生冲突。这个用户必须等待，直到加上的锁被释放。

两到多个事务相互争用资源时，可能发生死锁。UXDB支持自动检测死锁，并且自动处理。例如：事务T1已经获得表A上的排它锁（如成功更新表A上的列a：update A set a = 1;），事务T2已经获得表B 上的排它锁（如成功更新表B行的列a：update B set a = 1;），现在T1申请表B上的排它锁（如尝试更新表B上的列 a：update B set a = 1;），由于冲突，发生锁等待，这时T2申请表A上的排它锁（如尝试更新表A上的列a：update A set a = 1;），也冲突，形成死锁。这时UXDB检测到T1和T2发生死锁，强制回滚T2，解决死锁。

第 9 章 存储管理

9.1. 表空间

9.1.1. 关于表空间

表空间允许在文件系统里定义那些代表数据库对象的文件存放的位置。 一旦创建了表空间，那么就可以在创建数据库对象的时候引用它。一个数据库可以有一个或多个表空间，创建数据库时自动创建系统表空间，并为缺省的默认表空间。一个表空间只隶属于一个数据库，只有在创建了数据库之后才能创建属于它的表空间。

隶属于一个数据库的表空间用于存储该数据库的数据库对象。在创建数据库对象时可以使用TABLESPACE子句指明该对象所使用的表空间；没有给出TABLESPACE子句，则这些对象使用缺省表空间。

UXDB数据库中的表空间允许数据库管理员在文件系统中定义用来存放表示数据库对象的文件的位置。一旦被创建，表空间就可以在创建数据库对象时通过名称引用。

通过使用表空间，管理员可以控制一个UXDB安装的磁盘布局。 这么做至少有两个用处。首先，如果初始化集簇所在的分区或者卷用光了空间，而又不能在逻辑上扩展或者做别的什么操作，那么表空间可以被创建在一个不同的分区上，直到系统可以被重新配置。

其次，表空间允许管理员根据数据库对象的使用模式来优化性能。例如，一个很频繁使用的索引可以被放在非常快并且非常可靠的磁盘上，如一种非常贵的固态设备。同时，一个很少使用的或者对性能要求不高的存储归档数据的表可以存储在一个便宜但比较慢的磁盘系统上。

> **警告**
>
> 即便是位于主要的UXDB数据目录之外，表空间也是数据库集簇的一部分并且不能被视作数据文件的一个自治集合。它们依赖于包含在主数据目录中的元数据，并且因此不能被附加到一个 不同的数据库集簇或者单独备份。类似地，如果丢失一个表空间（文件删除、磁盘失效等）， 数据库集簇可能会变成不可读或者无法启动。把一个表空间放在一个临时文件系统 （如一个内存虚拟盘）上会带来整个集簇的可靠性风险。

9.1.2. 基于图形界面的表空间管理

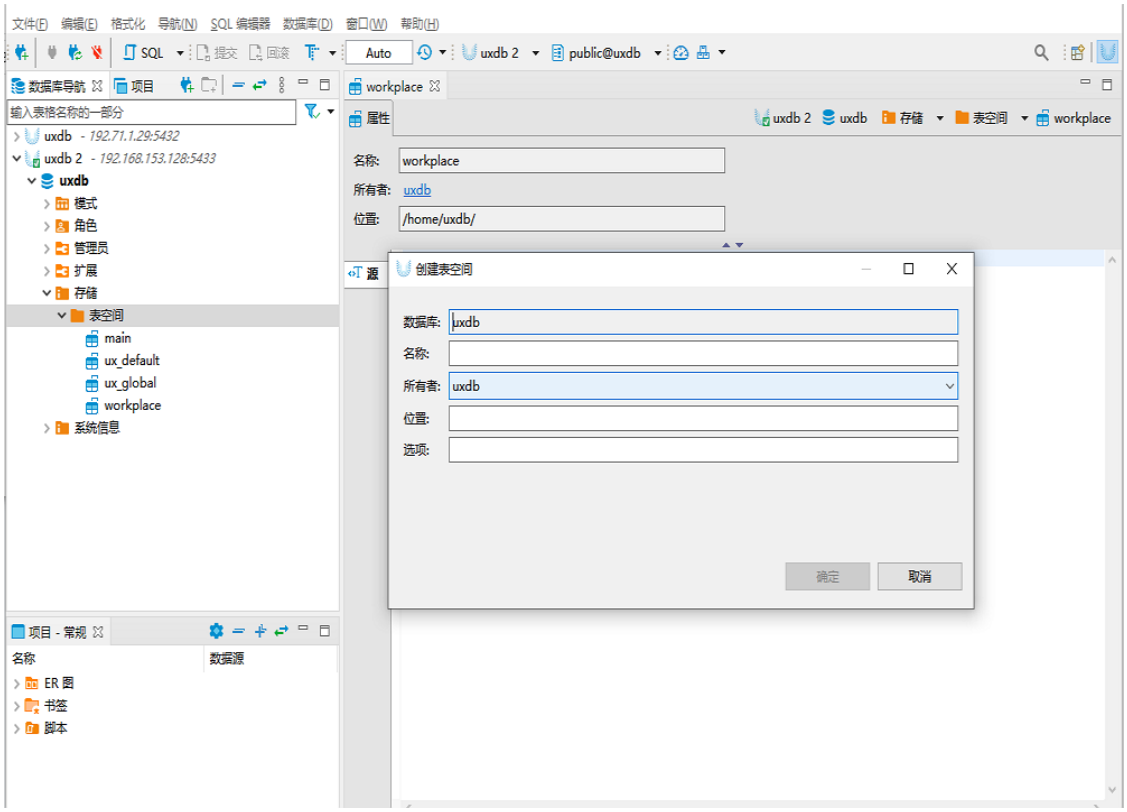
9.1.2.1. 新建表空间

通过UXDB数据库对象管理工具可视化图形界面的引导，创建表空间。

鼠标定位在导航树上表空间或“表空间名称”所在节点，右键单击表空间或“表空间名称”；在弹出式菜单中选择“新建表空间”菜单项，新建表空间。

9.1.2.2. “基本属性”页

基本属性属性页，可以定义或查看表空间的主要属性。



“基本属性”页包含以下信息。

- 数据库

  表空间归属于数据库的名称，创建时默认填写，在有多个数据库存在时，用户可以自定义在某个数据库下创建表空间。

- 名称

  待创建的或者当前查看的表空间的名称。在“新建表空间”操作过程中，用户可以指定一个有效的标识符作为表空间名。表空间名不能重复。

- 所有者

  该表空间的所有者名称，所有者应该是数据库系统的用户之一。

- 位置

  该表空间所存储的位置。显示该表空间已使用的空间大小。

9.1.2.3. “源”页

通过信息浏览区源对话框，用户可以查看创建表空间的SQL语句。

9.1.2.4. 删除表空间

通过UXDB数据库对象管理工具可视化图形界面的引导，删除表空间。

鼠标定位在导航树上“表空间”所在节点，右键单击“表空间名称”；在弹出式菜单中选择“删除”菜单项，即可实现表空间的删除。

9.1.3. SQL语句

9.1.3.1. 查看表空间信息

当初始化数据库实例时，UXDB数据库会自动创建三个表空间，ux_default和ux_global和main。

ux_global表空间被用于共享系统目录。ux_default表空间是template1和template0数据库的默认表空间（因此也是所有其他数据库的默认表空间，除非在使用CREATE DATABASE创建数据库时被其中的TABLESPACE子句覆盖）。

main表空间类似于ux_default表空间，可以通过手动指定默认空间。用户可通过检查系统表ux_tablespace，确定现有表空间的集合。

```
uxdb=# SELECT spcname FROM ux_tablespace;
  spcname
------------
 ux_default
 ux_global
 main
(3 行记录)
```

uxsql程序的db元命令也可以用来列出现有的表空间。

```
uxdb=# \db
        表空间列表
   名称   |拥有者|所在地
------------+--------+--------
 main       |uxdb   |
 ux_default |uxdb   |
 ux_global  |uxdb   |
(3 行记录)
```

要定义一个表空间，使用CREATE TABLESPACE命令，例如：

```
CREATE TABLESPACE fastspace LOCATION '/ssd1/uxsino/data';
```

这个位置必须是一个已有的空目录，并且属于UXDB操作系统用户。所有后续在该表空间中创建的对象都将被存放在这个目录下的文件中。该位置不能放在可移动或者瞬时存储上，因为如果表空间丢失会导致集簇无法工作。

> **注意**
>
> - 通常在每个逻辑文件系统上创建多于一个表空间没有什么意义，因为你无法控制在一个逻辑文件系统中特定文件的位置。不过，UXDB不强制任何这样的限制，并且事实上它不会注意你的系统上的文件系统边界。它只是在你告诉它要使用的目录中存储文件。
> - 表空间的创建本身必须作为一个数据库超级用户完成，但在创建完之后之后你可以允许普通数据库用户来使用它。要这样做，给数据库普通用户授予表空间上的CREATE权限。

表、索引和整个数据库都可以被分配到特定的表空间。想这么做，在给定表空间上有CREATE权限的用户必须把表空间的名字以一个参数的形式传递给相关的命令。例如，下面的命令在表空间space1中创建一个表：

CREATE TABLE foo(i int) TABLESPACE space1;

另外，还可以使用default_tablespace参数：

SET default_tablespace = space1;
CREATE TABLE foo(i int);

当default_tablespace被设置为非空字符串，那么它就为没有显式TABLESPACE子句的CREATE TABLE和CREATE INDEX命令提供一个隐式TABLESPACE子句。

9.1.3.2. 表空间设置

- 大纲

  ALTER TABLESPACE name RENAME TO new_name
  ALTER TABLESPACE name OWNER TO { new_owner | CURRENT_USER | SESSION_USER }
  ALTER TABLESPACE name SET (tablespace_option = value [, ...])
  ALTER TABLESPACE name RESET (tablespace_option [, ...])

- 描述

  ALTER TABLESPACE用于更改一个表空间的定义。

  > **注意**
  >
  > 1. 更改一个表空间的定义，必须拥有它。
  > 2. 修改拥有者，还必须是新拥有角色的一个直接或间接成员（注意数据库管理员自动拥有这些特权）。

- 参数

  temp_tablespaces参数

  temp_tablespaces参数，决定临时表和索引的位置，以及用于大数据集排序等目的的临时文件的位置。这可以是一个表空间名的列表，而不是只有一个。因此，与临时对象有关的负载可以散布在多个表空间上。每次要创建一个临时对象时，将从列表中随机取一个成员来存放它。

  与一个数据库相关联的表空间用来存储该数据库的系统目录。此外，如果没有给出TABLESPACE子句并且没有在default_tablespace或temp_tablespaces（如适用）中指定其他选择，它还是在该数据库中创建的表、索引和临时文件的默认表空间。如果一个数据库被创建时没有指定表空间，它会使用其模板数据库相同的表空间。

- 示例

  如重命名表空间，如下所示。

  uxdb=# **ALTER TABLESPACE workplace RENAME TO jobplace;**
  ALTER TABLESPACE
  uxdb=# **\db**
  表空间列表
  名称 |拥有者| 所在地
  ----------+--------+----------

```
 jobplace   | uxdb   | /home/uxdb
 main       | uxdb   |
 ux_default | uxdb   |
 ux_global  | uxdb   |
(4 行记录)
```

9.1.3.3. 表空间删除

表空间一旦被创建，就可以被任何数据库使用，前提是请求的用户具有足够的权限。这也意味着，一个表空间只有在所有使用它的数据库中所有对象都被删除掉之后才可以被删掉。

删除一个空的表空间，使用DROP TABLESPACE命令。

> **注意**
>
> 1. 一个表空间只能被其拥有者或数据库管理员删除。
> 2. 在一个表空间能被删除前，其中必须没有任何数据库对象。即使当前数据库中没有对象正在使用该表空间，也可能有其他数据库的对象存在于其中。
> 3. 如果该表空间被列在任何活动会话的temp_tablespaces设置中，DROP也可能会失败，因为可能有临时文件存在其中。