优炫数据库工具使用手册 2.1

UXSINO
优 炫 软 件

优炫数据库工具使用手册 2.1

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。

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

表格清单

前言

1. 文档目的

本文档介绍UXDB客户端、服务器端的工具和使用方法的参考信息。为相关技术人员和用户提供参考。

2. 文档对象

- 技术支持工程师
- 维护工程师
- 优炫数据库用户

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	2022-12-08	第一次正式发布。

第 1 章 UXDB客户端应用

本节包含UXDB客户端工具和使用方法的参考信息。这些命令并不全是通用工具，某些需要特殊权限。这些应用的共同特征是它们可以在任意主机上运行，与数据库服务器的位置无关。

当在命令行上指定用户和数据库名时，它们的大小写会被保留，空格或特殊字符需要放在引号内。表名和其他标识符的大小写不会被保留，若要保留需将表名和其他标识符放在引号内。

通过下命令添加工具路径到操作系统环境变量，这样在任何路径下都可以调用UXDB客户端工具：

export PATH=/home/uxdb/uxdbinstall/dbsql/bin:$PATH

1.1. clusterdb

clusterdb — 聚簇一个UXDB数据库

1.1.1. 用法

clusterdb [*connection-option*...] [--verbose | -v] u [--table | -t *table*] ... [*dbname*]

clusterdb [*connection-option*...] [--verbose | -v] --all | -a

1.1.2. 描述

clusterdb是一个工具，它用来对一个UXDB数据库中的表进行重新聚簇。它会寻找之前已经被聚簇过的表，并且再次在最后使用过的同一个索引上对它们重新聚簇。没有被聚簇过的表将不会被影响。

clusterdb是 SQL 命令CLUSTER的封装。在通过这个工具和其他方法访问服务器来聚簇数据库之间没有实质性的区别。

1.1.3. 选项

clusterdb接受下列命令行参数：

-a
--all

聚簇所有数据库。

[-d] *dbname*
[--dbname=]*dbname*

指定要被聚簇的数据库名称。如果这个参数没有被指定并且-a（或--all）没有被使用，数据库名将从环境变量UXDATABASE中读出。如果该环境变量也没有被设置，指定给该连接的用户名将被用作数据库名。

-e
--echo

回显clusterdb生成并发送给服务器的命令。

-q
--quiet

不显示进度消息。

-t *table*
--table=*table*

只聚簇*table*。可以通过写多个-t开关来聚簇多个表。

-v
--verbose

在处理期间打印详细信息。

-V
--version

打印clusterdb版本并退出。

-?
--help

显示关于clusterdb命令行参数的帮助并退出。

clusterdb也接受下列命令行参数用于连接参数：

-h *host*
--host=*host*

指定运行服务器的机器的主机名。如果该值以一个斜线开始，它被用作 Unix 域套接字的目录。

-p *port*
--port=*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*
--username=*username*

要作为哪个用户连接。

-w
--no-password

永远不要发出密码提示。如果服务器需要密码验证，而通过其他方式(如.pgpass文件)无法获得密码，则连接尝试将失败。此选项在没有用户可以输入密码的批处理作业和脚本中很有用。

-W
--password

强制clusterdb在连接到一个数据库之前提示输入密码。

这个选项不是必需的，因为如果服务器要求输入密码，clusterdb将自动提示输入密码。但是，clusterdb将浪费一次连接尝试来发现服务器想要一个密码。在某些情况下建议用-W来避免额外的连接尝试。

--maintenance-db=*dbname*

指定要连接的数据库名称。如果没有指定，将使用UXDB数据库。而如果它也不存在，将使用template1。

1.1.4. 环境变量

UXDATABASE
UXHOST
UXPORT
UXUSER

默认连接参数。

UX_COLOR

规定在诊断消息中是否使用颜色。可选的值为always，auto，never。

和大部分其他UXDB工具相似，这个工具也使用libuxsql支持的环境变量。

1.1.5. 诊断

在有困难时，可以在CLUSTER和uxsql中找潜在问题和错误消息的论述。数据库服务器必须运行在目标主机上。同样，任何libuxsql前端库使用的默认连接设置和环境变量都将适用于此。

1.1.6. 示例

要聚簇数据库test：

$ **clusterdb test**

foo是一个在xyzzy数据库中有聚簇索引的表。

要聚簇在数据库xyzzy中的一个表foo：

$ **clusterdb --table foo xyzzy**

1.2. createdb

createdb — 创建一个新的UXDB数据库

1.2.1. 用法

createdb [*connection-option...*] [*option...*] [*dbname* [*description*]]

1.2.2. 描述

createdb创建一个新的UXDB数据库。

通常，执行这个命令的数据库用户将成为新数据库的所有者。但是，如果执行用户具有合适的权限，可以通过-O选项指定一个不同的所有者。

createdb是SQL命令CREATE DATABASE的封装。通过这个工具和其他方法访问服务器来创建数据库之间没有实质性的区别。

1.2.3. 选项

createdb接受下列命令行参数：

dbname

指定要被创建的数据库名。该名称必须在这个集群中所有UXDB数据库中唯一。默认是创建一个与当前系统用户同名的数据库。

description

指定与新创建的数据库相关联的一段注释。

-D tablespace
--tablespace=tablespace

指定该数据库的默认表空间（这个名称被当作一个双引号引用的标识符处理）。

-e
--echo

回显createdb生成并发送到服务器的命令。

-E encoding
--encoding=encoding

指定要在这个数据库中使用的字符编码模式。

-l locale
--locale=locale

指定要在这个数据库中使用的语言区域。这等效于同时指定--lc-collate和--lc-ctype。

--lc-collate=locale

指定要在这个数据库中使用的LC_COLLATE设置

--lc-ctype=locale

指定要在这个数据库中使用的LC_CTYPE设置。

-O owner
--owner=owner

指定拥有这个新数据库的数据库用户（这个名称被当作一个双引号引用的标识符处理）。

-T template
--template=template

指定用于创建这个数据库的模板数据库（这个名称被当作一个双引号引用的标识符处理）。

-V
--version

打印createdb版本并退出。

-?
--help

显示关于createdb命令行参数的帮助并退出。

createdb也接受下列命令行参数用于连接参数：

-h host
--host=host

指定运行服务器的机器的主机名。如果该值以一个斜线开始，它被用作Unix域套接字的目录。

-p port
--port=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。

--running-mode = standard | compatible | mysql

运行模式，默认模式是standard。standard表示标准模式，compatible表示兼容模式，mysql表示mysql模式。

-U username
--username=username

要作为哪个用户连接。

-w
--no-password

永远不要发出密码提示。如果服务器需要密码验证，而通过其他方式(如.pgpass文件)无法获得密码，则连接尝试将失败。此选项在没有用户可以输入密码的批处理作业和脚本中很有用。

-W
--password

强制createdb在连接到一个数据库之前提示输入密码。

这个选项不是必需的，因为如果服务器要求输入密码，createdb将自动提示输入密码。但是，createdb将浪费一次连接尝试来发现服务器想要一个密码。在某些情况下建议用-W来避免额外的连接尝试。

--maintenance-db=dbname

指定要创建新数据库时连接的数据库名称。如果没有指定，将使用UXDB数据库。而如果它也不存在（或者如果它就是要创建新数据库的名称），将使用template1。

1.2.4. 环境变量

UXDATABASE

如果被设置，就是要创建的数据库名，除非在命令行中覆盖。

UXHOST
UXPORT
UXUSER

默认连接参数。如果没有在命令行或UXDATABASE指定要创建的数据库名，UXUSER也可以确定要创建的数据库名称。

UX_COLOR

规定在诊断消息中是否使用颜色。可选的值为always，auto，never。

和大部分其他UXDB工具相似，这个工具也使用libuxsql支持的环境变量。

1.2.5. 诊断

在有困难时，可以在CREATE DATABASE和uxsql中找潜在问题和错误消息的论述。数据库服务器必须运行在目标主机上。同样，任何libuxsql前端库使用的默认连接设置和环境变量都将适用于此。

1.2.6. 示例

要使用默认数据库服务器创建数据库demo：

$ **createdb demo**

要在主机eden、端口5000上使用template0模板数据库创建数据库demo，下面是命令行命令和底层SQL命令：

$ **createdb -p 5000 -h eden -T template0 -e demo**
CREATE DATABASE demo TEMPLATE template0;

1.3. createuser

createuser — 创建一个新的UXDB用户

1.3.1. 用法

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

1.3.2. 描述

createuser创建一个新UXDB用户（或者更准确些应该是一个角色）。只有超级用户和具有CREATEROLE特权的用户才能创建新用户，因此createuser必须由超级用户或具有CREATEROLE特权的用户调用。

如果希望创建一个新的超级用户，必须以超级用户进行连接，而不仅仅是具有CREATEROLE特权。作为一个超级用户意味着绕过数据库中所有访问权限检查的能力，因此超级用户权限不能轻易被授予。

createuser是 SQL 命令CREATE ROLE的封装。在通过这个工具和其他方法访问服务器来创建用户之间没有实质性的区别。

1.3.3. 选项

createuser接受下列命令行参数：

username

指定要被创建的UXDB用户的名称。这个名称必须与这个UXDB中所有现存角色不同。

-c number
--connection-limit=number

为该新用户设置一个最大连接数。默认值为不设任何限制。

-d
--createdb

新用户将被允许创建数据库。

-D
--no-createdb

新用户将不被允许创建数据库。这是默认值。

-e
--echo

回显createuser生成并发送给服务器的命令。

-g role
--role=role

指定一个角色，新角色将立即加入其中成为其成员。如果要把新角色加入到多个角色中作为成员，可以写多个-g。

-i
--inherit

新角色将自动继承把它作为成员的角色的特权。这是默认值。

-I
--no-inherit

新角色将不会自动继承把它作为成员的角色的特权。

--interactive

如果没有在命令行上指定用户名，提示缺少用户名。并且-d/-D，-r/-R，-s/-S选项任何属性没有在命令行被指定时，也会提示缺少属性（而不会使用缺省值）。

-l
--login

新用户将被允许登录（即，该用户名能被用作初始会话用户标识符）。这是默认值。

-L
--no-login

新用户将不被允许登入（一个没有登录特权的角色仍然可以以管理数据库权限的方式而存在）。

-P
--pwprompt

如果给定，createuser将发出一个提示要求新用户输入密码。你没有计划使用密码，这就不是必须的。

-r
--createrole

新用户将被允许创建新的角色（即，这个用户将具有CREATEROLE特权）。

--running-mode = standard | compatible | mysql

运行模式，默认模式是standard。standard表示标准模式，compatible表示兼容模式，mysql表示mysql模式。

-R
--no-createrole

新用户将不被允许创建新角色。这是默认值。

--security

检查输入密码是否满足强度要求，满足则通过，以下任意一条不满足则返回错误信息并退出。规定如下：

1. 长度必须在8-99个字符之间（包括8、99）。
2. 至少包含一个大写字母。
3. 字少包含一个小写字母。
4. 至少包含一个数字。
5. 至少包含一个特殊字符。
6. 首尾任意一处不能带有空格。

-s
--superuser

新用户将成为一个超级用户。

-S
--no-superuser

新用户将不会成为一个超级用户。这是默认值。

-V
--version

打印createuser版本并退出。

--replication

新用户将具有REPLICATION特权。

--no-replication

新用户将不具有REPLICATION特权。这是默认值。

-?
--help

显示有关createuser命令行参数的帮助并退出。

createuser也接受下列命令行参数作为连接参数：

-h host
--host=host

指定运行服务器的机器的主机名。如果该值以一个斜线开始，它被用作Unix域套接字的目录

-p port
--port=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
--username=username

要作为哪个用户连接（不是要创建的用户名）。

-w
--no-password

永远不要发出密码提示。如果服务器需要密码验证，而通过其他方式(如.pgpass文件)无法获得密码，则连接尝试将失败。此选项在没有用户可以输入密码的批处理作业和脚本中很有用。

-W
--password

强制createuser在连接到一个数据库之前提示输入密码（用来连接到服务器，而不是新用户的密码）。

这个选项不是必需的，因为如果服务器要求输入密码，createuser将自动提示输入密码。但是，createuser将浪费一次连接尝试来发现服务器想要一个密码。在某些情况下建议用-W来避免额外的连接尝试。

1.3.4. 环境变量

UXHOST
UXPORT
UXUSER

默认连接参数。

UX_COLOR

规定在诊断消息中是否使用颜色。可选的值为always，auto，never。

和大部分其他UXDB工具相似，这个工具也使用libuxsql支持的环境变量。

1.3.5. 诊断

在有困难时，可以在CREATE ROLE和uxsql中找潜在问题和错误消息的论述。数据库服务器必须运行在目标主机上。同样，任何libuxsql前端库使用的默认连接设置和环境变量都将适用于此。

1.3.6. 示例

要在默认数据库服务器上创建一个用户joe：

```
$ createuser joe
```

要在默认数据库服务器上创建一个用户joe并提示要求一些额外属性：

```
$ createuser --interactive joe
Shall the new role be a superuser? (y/n) n
Shall the new role be allowed to create databases? (y/n) n
Shall the new role be allowed to create more new roles? (y/n) n
```

要在主机eden、端口5000上的服务器创建一个用户joe并带有显式指定的属性：

```
$ createuser -h eden -p 5000 -S -D -R -e joe
CREATE ROLE joe NOSUPERUSER NOCREATEDB NOCREATEROLE INHERIT LOGIN;
```

创建用户joe为一个超级用户并且立刻分配一个密码：

```
$ createuser -P -s -e joe
Enter password for new role: xyzzy
Enter it again: xyzzy
CREATE ROLE joe PASSWORD 'md5b5f5ba1a423792b526f799ae4eb3d59e' SUPERUSER
 CREATEDB CREATEROLE INHERIT LOGIN;
```

在上面的示例中，在录入新密码时新密码并没有真正地被回显，但是为了清晰，我们特意把它列了出来。如你所见，该密码在被发送给客户端之前会被加密。

1.4. dropdb

dropdb — 移除一个UXDB数据库

1.4.1. 用法

dropdb [*connection-option...*] [*option...*] *dbname*

1.4.2. 描述

dropdb移除一个现有的UXDB数据库。执行这个命令的用户必须是数据库超级用户或该数据库的所有者。

dropdb是 SQL 命令DROP DATABASE的封装。在通过这个工具和其他方法访问服务器来删除数据库之间没有实质性的区别。

1.4.3. 选项

dropdb接受下列命令行参数：

dbname

指定要被移除的数据库的名称。

-e
--echo

回显dropdb生成并发送给服务器的命令。

-i
--interactive

在做任何破坏性的工作之前发出验证提示。

-V
--version

打印dropdb版本并退出。

--if-exists

如果数据库不存在不抛出一个错误而是发出一个提醒。

-?
--help

显示有关dropdb命令行参数的帮助并退出。

dropdb也接受下列命令行参数作为连接参数：

-h host
--host=host

指定运行服务器的机器的主机名。如果该值以一个斜线开始，它被用作Unix域套接字的目录。

-p port
--port=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。

--running-mode = standard | compatible | mysql

运行模式，默认模式是standard。standard表示标准模式，compatible表示兼容模式，mysql表示mysql模式。

-U username
--username=username

要作为哪个用户连接。

-w
--no-password

永远不要发出密码提示。如果服务器需要密码验证，而通过其他方式(如.pgpass文件)无法获得密码，则连接尝试将失败。此选项在没有用户可以输入密码的批处理作业和脚本中很有用。

-W
--password

强制dropdb在连接到一个数据库之前提示输入密码。

这个选项不是必需的，因为如果服务器要求输入密码，dropdb将自动提示输入密码。但是，dropdb将浪费一次连接尝试来发现服务器想要一个密码。在某些情况下建议用-W来避免额外的连接尝试。

--maintenance-db=dbname

指定要连接到来发现哪些其他数据库应该被删除的数据库名。如果没有指定，将使用UXDB数据库。而如果它也不存在，将使用template1。

1.4.4. 环境变量

UXHOST
UXPORT
UXUSER

默认连接参数。

UX_COLOR

规定在诊断消息中是否使用颜色。可选的值为always，auto，never。

和大部分其他UXDB工具相似，这个工具也使用libuxsql支持的环境变量。

1.4.5. 诊断

在有困难时，可以在DROP DATABASE和uxsql中找潜在问题和错误消息的论述。数据库服务器必须运行在目标主机上。同样，任何libuxsql前端库使用的默认连接设置和环境变量都将适用于此。

1.4.6. 示例

要在默认数据库服务器上移除数据库demo：

```
$ dropdb demo
```

要使用在主机eden、端口5000上的服务器中移除数据库demo，并带有验证和回显，看看下面的命令：

```
$ dropdb -p 5000 -h eden -i -e demo
Database "demo" will be permanently deleted.
Are you sure? (y/n) y
DROP DATABASE demo;
```

1.5. dropuser

dropuser — 移除一个UXDB用户

1.5.1. 用法

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

1.5.2. 描述

dropuser移除一个已有的UXDB用户。只有超级用户以及具有CREATEROLE特权的用户能够移除UXDB用户（要移除一个超级用户，连接用户必须是一个超级用户）。

dropuser是SQL命令DROP ROLE的封装。在通过这个工具和其他方法访问服务器来删除用户之间没有实质性的区别。

1.5.3. 选项

dropuser接受下列命令行参数：

username

指定要移除的UXDB用户的名字。如果没有在命令行指定并且使用了-i/--interactive选项，将被提醒要求一个用户名。

-e
--echo

回显dropuser生成并发送给服务器的命令。

-i
--interactive

在实际移除该用户之前提示要求确认，并且在没有在命令行指定用户名时提示指定移除的用户名。

-V
--version

打印dropuser版本并退出。

--if-exists

如果用户不存在时不抛出一个错误而是发出一个提示。

-?
--help

显示有关dropuser命令行参数的帮助并退出。

dropuser也接受下列命令行参数作为连接参数：

-h *host*
--host=*host*

指定运行服务器的机器的主机名。如果该值以一个斜线开始，它被用作 Unix 域套接字的目录。

-p *port*
--port=*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。

--running-mode = standard | compatible | mysql

运行模式，默认模式是standard。standard表示标准模式，compatible表示兼容模式，mysql表示mysql模式。

-U *username*
--username=*username*

要作为哪个用户连接（不是要移除的用户名）。

-w
--no-password

永远不要发出密码提示。如果服务器需要密码验证，而通过其他方式(如.pgpass文件)无法获得密码，则连接尝试将失败。此选项在没有用户可以输入密码的批处理作业和脚本中很有用。

-W
--password

强制dropuser在连接到一个数据库之前提示输入密码。

这个选项不是必需的，因为如果服务器要求输入密码，dropuser将自动提示输入密码。但是，dropuser将浪费一次连接尝试来发现服务器想要一个密码。在某些情况下建议用-W来避免额外的连接尝试。

1.5.4. 环境变量

UXHOST
UXPORT
UXUSER

默认连接参数。

UX_COLOR

规定在诊断消息中是否使用颜色。可选的值为always，auto，never。

和大部分其他UXDB工具相似，这个工具也使用libuxsql支持的环境变量。

1.5.5. 诊断

在有困难时，可以在DROP DATABASE和uxsql中找潜在问题和错误消息的论述。数据库服务器必须运行在目标主机上。同样，任何libuxsql前端库使用的默认连接设置和环境变量都将适用于此。

1.5.6. 示例

要从默认数据库服务器移除用户joe：

$ **dropuser joe**

要使用在主机eden、端口 5000 上的服务器移除用户joe，并带有验证和回显，可使用下面的命令：

$ **dropuser -p 5000 -h eden -i -e joe**
Role "joe" will be permanently removed.
Are you sure? (y/n) **y**
DROP ROLE joe;

1.6. ecux

ecux — 嵌入式SQL C预处理器

1.6.1. 用法

ecux［*option*...］*file*...

1.6.2. 描述

ecux是用于C程序的嵌入式SQL预处理器。它通过将SQL调用替换为特殊函数调用把带有嵌入式SQL语句的C程序转换为普通C代码。输出文件可以被任何C编译器工具链处理。

ecux将把命令行中给出的每一个输入文件转换为相应的 C 输出文件。 输入文件更适宜于使用扩展名.uxc。 该扩展名将被替换为.c来决定输出文件名。 输出文件名也可以使用-o选项覆盖。

1.6.3. 选项

ecux接受下列命令行参数：

-c

自动从SQL代码生成确定的C代码。当前，仅对EXEC SQL TYPE起效。

-C *mode*

设置一个兼容性模式。*mode*可以是INFORMIX或INFORMIX_SE。

-D *symbol*

定义一个C预处理器符号。

-h

分析头文件，这个选项包含选项“-c”。

-i

分析系统include文件。

-I *directory*

指定一个额外的包括路径，用来寻找通过EXEC SQL INCLUDE包括的文件。默认值是.（当前目录）、/usr/local/include、在编译时定义的UXDB“include”包括目录（默认：/home/uxdb/uxdbinstall/dbsql/include）以及/usr/include。

-o *filename*

指定ecux应该将它的所有输出写到给定的*filename*。

-r *option*

选择运行时行为。*option*可以是下列之一：

no_indicator

不使用指示器而使用特殊值来表示空值。

prepare

在使用所有语句之前准备它们。libecux将保持一个预备语句的缓冲并当语句再被执行时重用该语句。如果缓冲满了，libecux将释放最少使用的语句。

questionmarks

为兼容性原因允许使用问号作为占位符。

-t

打开事务的自动提交。在这种模式下，每一个SQL命令会被自动提交，除非它位于一个显式事务块中。在默认模式中，命令只有当EXEC SQL COMMIT被发出时才被提交。

-v

打印额外信息，包括版本和“include”路径。

--version

打印ecux版本并退出。

-?
--help

显示关于ecux命令行参数的帮助并退出。

1.6.4. 注解

在编译预处理好的C代码文件时，编译器需要能够找到UXDB包括目录中的ECUX头文件。因此，在调用编译器时，你可能必须使用-I选项（例如，-I/home/uxdb/uxdbinstall/dbsql/include）。

使用带有嵌入式SQL的C代码的程序必须被链接到libecux库，例如使用链接器选项-L/home/uxdb/uxdbinstall/dbsql/lib -lecux。

可以使用ux_config找到适合安装的目录。

1.6.5. 示例

如果你有一个名为prog1.uxc的嵌入式SQL C源文件，你可以使用下列命令序列创建一个可执行程序：

```
ecux prog1.uxc
cc -I /home/uxdb/uxdbinstall/dbsql/include -c prog1.c
cc -o prog1 prog1.o -L /home/uxdb/uxdbinstall/dbsql/lib -lecux
```

1.7. oid2name

oid2name — 解析一个UXDB数据目录中的OID和文件节点

1.7.1. 用法

oid2name [*option*...]

1.7.2. 描述

oid2name是一个帮助管理员检查被UXDB使用的文件结构的工具程序。要使用它，你需要熟悉数据库文件结构。

> **注意**
>
> 名称“oid2name”是有历史原因的，它确实有些误导性，因为在你使用它的大部分时间里，你实际关心的是表的文件结点编号（在数据目录中是可见的文件名）。请确定你理解表OID和表文件结点之间的区别！

oid2name连接到一个目标数据库并且抽取OID、文件节点或者表名信息。你也可以让它显示数据库OID或表空间OID。

1.7.3. 选项

oid2name接受下列命令行参数：

-f *filenode*
--filenode=*filenode*

显示具有文件结点*filenode*的表的信息。

-i
--indexes

在列举中包括索引和序列。

-o *oid*
--oid= *oid*

显示具有OID的表的信息 oid。

-q
--quiet

忽略头部（用于脚本）。

-s
--tablespaces

显示表空间OID。

-S
--system-objects

包括系统对象（位于 information_schema、ux_toast和ux_catalog模式中）。

-t *tablename_pattern*
--table=*tablename_pattern*

显示匹配*tablename_pattern*的表的信息。

-V
--version

打印oid2name版本并退出。

-x
--extended

为要显示的每个对象显示更多信息：表空间名、模式名以及OID。

-?
--help

显示有关oid2name命令行参数的帮助并退出。

oid2name也接受下列用于连接参数的命令行参数：

-d *database*
--dbname=*database*

要连接的数据库。

-h *host*
--host=*host*

数据库服务器的主机。

-p *port*
--port=*port*

数据库服务器的端口。

-U *username*
--username=*username*

用于连接的用户名。

要显示特定表，通过使用-o、-f和-t选择要显示哪个表。 -o采用一个 OID， -f采用一个文件节点， 而-t采用一个表名（实际上，它是一个LIKE模式，因此你可以用诸如foo%之类的东西）。这些选项你想用多少就用多少，最后的列举将包括所有匹配任意一个这些选项的对象。但是注意这些选项只能显示由-d给定的数据库中的对象。

如果你没有给出任何-o、-f或者-t，但是给出了-d，它将列出由-d指定的数据库中的所有表。在这种模式下，-S和-i选项控制什么会被列出。

如果你也没有给出-d，它将显示一个数据库 OID 的列表。你也可以给出-s来得到一个表空间列表。

1.7.4. 环境变量

UXHOST
UXPORT
UXUSER

默认连接参数。

和大部分其他UXDB工具相似，这个工具也使用libuxsql支持的环境变量。

1.7.5. 注解

oid2name要求一个运行着的数据库服务器并且其系统目录没有损坏。因此它对于数据库损坏的情况用处有限。

1.7.6. 示例

```
$ # 在这个数据库服务器中到底有什么？
$ oid2name
All databases:
   Oid  Database Name  Tablespace
----------------------------------
  17228       alvherre  ux_default
  17255     regression  ux_default
  17227      template0  ux_default
      1      template1  ux_default
```

```
$ oid2name -s
All tablespaces:
     Oid  Tablespace Name
------------------------
    1663       ux_default
    1664        ux_global
  155151          fastdisk
  155152           bigdisk

$ # 进入数据库alvherre。
$ cd $UXDATA/base/17228

$ # 得到默认表空间中前十个数据库对象，按尺寸排序。
$ ls -lS * | head -10
-rw-------  1 alvherre alvherre 136536064 sep 14 09:51 155173
-rw-------  1 alvherre alvherre  17965056 sep 14 09:51 1155291
-rw-------  1 alvherre alvherre   1204224 sep 14 09:51 16717
-rw-------  1 alvherre alvherre    581632 sep  6 17:51 1255
-rw-------  1 alvherre alvherre    237568 sep 14 09:50 16674
-rw-------  1 alvherre alvherre    212992 sep 14 09:51 1249
-rw-------  1 alvherre alvherre    204800 sep 14 09:51 16684
-rw-------  1 alvherre alvherre    196608 sep 14 09:50 16700
-rw-------  1 alvherre alvherre    163840 sep 14 09:50 16699
-rw-------  1 alvherre alvherre    122880 sep  6 17:51 16751

$ # 查看文件 155173 。
$ oid2name -d alvherre -f 155173
From database "alvherre":
  Filenode  Table Name
----------------------
    155173    accounts

$ # 查看多个对象。
$ oid2name -d alvherre -f 155173 -f 1155291
From database "alvherre":
  Filenode     Table Name
-------------------------
    155173       accounts
   1155291  accounts_pkey

$ # 你可以混合选项，并且用 -x 得到更多细节。
$ oid2name -d alvherre -t accounts -f 1155291 -x
From database "alvherre":
  Filenode     Table Name      Oid  Schema  Tablespace
-----------------------------------------------------
    155173       accounts   155173  public  ux_default
   1155291  accounts_pkey  1155291  public  ux_default

$ # 为每个数据库对象显示磁盘空间。
$ du [0-9]* |
> while read SIZE FILENODE
> do
>   echo "$SIZE       `oid2name -q -d alvherre -i -f $FILENODE`"
```

```
> done
16          1155287  branches_pkey
16          1155289  tellers_pkey
17561          1155291  accounts_pkey
...

$ # 为每个数据库对象按尺寸排序显示磁盘空间。
$ du [0-9]* | sort -rn | while read SIZE FN
> do
>   echo "$SIZE   `oid2name -q -d alvherre -f $FN`"
> done
133466           155173    accounts
17561          1155291  accounts_pkey
1177           16717  ux_proc_proname_args_nsp_index
...

$ # 使用ux_tblspc目录，查看表空间。
$ cd $UXDATA/ux_tblspc
$ oid2name -s
All tablespaces:
    Oid  Tablespace Name
-----------------------
   1663       ux_default
   1664        ux_global
 155151          fastdisk
 155152           bigdisk

$ # 哪些数据库在表空间 "fastdisk" 中有对象？
$ ls -d UX_10_201707211/*
155151/17228/  155151/UX_VERSION

$ # 查看数据库 17228 ？
$ oid2name
All databases:
   Oid  Database Name  Tablespace
----------------------------------
 17228       alvherre  ux_default
 17255     regression  ux_default
 17227      template0  ux_default
     1      template1  ux_default

$ # 查看数据库17228在该表空间中有哪些对象？
$ # 17228是数据库oid，按照实际oid查看
$ cd UX_10_201707211/17228
$ ls -l
total 0
-rw-------  1 uxdb uxdb 0 sep 13 23:20 155156

$ # 查看对象155156。
$ oid2name -d alvherre -f 155156
From database "alvherre":
  Filenode  Table Name
----------------------
   155156         foo
```

1.8. reindexdb

reindexdb — 重索引一个UXDB数据库

1.8.1. 用法

reindexdb [*connection-option...*] [*option...*] [--schema | -S *schema*] ... [--table | -t *table*] ... [--index | -i *index*] ... [*dbname*]

reindexdb [*connection-option...*] [*option...*] --all | -a

reindexdb [*connection-option...*] [*option...*] --system | -s [*dbname*]

1.8.2. 描述

reindexdb是用于重建一个UXDB数据库中索引的工具。

reindexdb是SQL命令REINDEX的封装。在通过这个工具和其他方法访问服务器来重索引数据库之间没有实质性的区别。

1.8.3. 选项

reindexdb接受下列命令行参数：

-a
--all

重索引所有数据库。

[-d] *dbname*
[--dbname=]*dbname*

指定要被重索引的数据库名。如果这没有被指定并且没有使用-a（或--all），数据库名可以从环境变量UXDATABASE中被读出。如果环境变量也没被设置，则该连接指定的用户名将被用作数据库名。

-e
--echo

回显reindexdb生成并发送到服务器的命令。

-i *index*
--index=*index*

只是重建*index*。可以通过写多个-i开关来重建多个索引。

-q
--quiet

不显示进度消息。

-s
--system

索引数据库的系统目录。

-S *schema*
--schema=*schema*

只对*schema*重建索引。 通过写多个-S开关可以指定多个要重建索引的模式。

-t *table*
--table=*table*

只索引*table*。可以通过写多个-t开关来重建多个表索引。

-v
--verbose

在处理时打印详细信息。

-V
--version

打印reindexdb版本并退出。

-?
--help

显示有关reindexdb命令行参数的帮助并退出。

reindexdb也接受下列命令行参数用于连接参数：

-h *host*
--host=*host*

指定运行服务器的机器的主机名。如果该值以一个斜线开始，它被用作 Unix 域套接字的目录。

-p *port*
--port=*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*
--username=*username*

要作为哪个用户连接。

-w
--no-password

永远不要发出密码提示。如果服务器需要密码验证，而通过其他方式(如.pgpass文件)无法获得密码，则连接尝试将失败。此选项在没有用户可以输入密码的批处理作业和脚本中很有用。

-W
--password

强制reindexdb在连接到一个数据库之前提示输入密码。

这个选项不是必需的，因为如果服务器要求输入密码，reindexdb将自动提示输入密码。但是，reindexdb将浪费一次连接尝试来发现服务器想要一个密码。在某些情况下建议用-W来避免额外的连接尝试。

--maintenance-db=*dbname*

指定要连接到来发现哪些其他数据库应该被重索引的数据库名。如果没有指定，将使用uxdb数据库。而如果它也不存在，将使用template1。

1.8.4. 环境变量

UXDATABASE
UXHOST
UXPORT
UXUSER

默认连接参数。

和大部分其他UXDB工具相似，这个工具也使用libuxsql支持的环境变量。

1.8.5. 诊断

在有困难时，可以在REINDEX和uxsql中找潜在问题和错误消息的论述。数据库服务器必须运行在目标主机上。同样，任何libuxsql前端库使用的默认连接设置和环境变量都将适用于此。

1.8.6. 注解

reindexdb可能需要多次连接到UXDB服务器，每一次都会询问密。在这种情况下使用一个~/.uxpass文件会更方便。

1.8.7. 示例

要重索引数据库test：

$ **reindexdb test**

要重索引名为abcd的数据库中的表foo和索引bar：

$ **reindexdb --table foo --index bar abcd**

1.9. ux_basebackup

ux_basebackup — 获得UXDB集群的基础备份

1.9.1. 用法

ux_basebackup [*option*...]

1.9.2. 描述

ux_basebackup被用于获得一个正在运行的UXDB数据库集群的基础备份。获得这些备份不会影响连接到该数据库的其他客户端，并且可以被用于时间点恢复以及用作一个日志传送或流复制备用服务器的开始点。

ux_basebackup建立数据库集群文件的一份二进制副本，同时保证系统进入和退出备份模式。备份总是从整个数据库集群获得，不可能备份单个数据库或数据库对象。关于个体数据库备份，必须使用如ux_dump的工具。

备份通过一个常规UXDB连接制作，并且使用复制协议。该连接必须由一个超级用户或者一个具有REPLICATION权限的用户建立，并且ux_hba.conf必须显式地允许该复制连接。该服务器还必须被配置，使max_wal_senders设置得足够高以留出至少一个会话用于备份以及一个用于WAL流（如果使用流）。

在同一时间可以有多个ux_basebackup运行，但是从性能的角度来说最好只做一个备份并且复制结果。

ux_basebackup不仅能从主控机也能从备用机创建一个基础备份。要从备用机获得一个备份，设置备用机让它能接受复制连接（也就是，设置max_wal_senders和hot_standby，并且配置host-based authentication），还需要在主机上启用full_page_writes。

注意在来自备用机的在线备份中有一些限制：

- 不会在被备份的数据库集群中创建备份历史文件。
- 如果正在使用-X none，不保证备份所需的所有 WAL 文件在备份结束时被归档。
- 如果在在线备份期间备用机被提升为主控机，备份会失败。
- 备份所需的所有WAL记录必须包含足够的全页写，这要求你在主控机上启用full_page_writes并且不使用一个类似ux_compresslog的工具以archive_command从WAL文件中移除full_page_writes。

1.9.3. 选项

下列命令行选项控制输出的位置和格式。

-D *directory*
--uxdata=*directory*

将输出写到哪个目录。如果必要，ux_basebackup将创建该目录及任何父目录。该目录可能已经存在，但是如果该目录已经存在并且非空就是一个错误。

当备份处于tar模式中并且目录被指定为-（破折号）时，tar文件将被写到stdout。

这个选项是必需的。

-F *format*
--format=*format*

为输出选择格式。*format*可以是下列之一：

p
plain

把输出写成普通文件，使用和当前数据目录和表空间相同的布局。当集群没有额外表空间时，整个数据库将被放在目标目录中。如果集群包含额外的表空间，主数据目录将被放置在目标目录中，但是所有其他表空间将被放在它们位于服务器上的相同的绝对路径中。

这是默认格式。

t
tar

将输出写成目标目录中的tar文件。主数据目录将被写入到一个名为base.tar的文件中，并且其他表空间将被以其OID命名。

如果值-（破折号）被指定为目标目录，tar内容将被写到标准输出，适合于管道输出到其他程序，例如gzip。只有当集簇没有额外表空间并且没有使用WAL流时这才是可能的。

-r *rate*
--max-rate=*rate*

从该服务器传输数据的最大传输率。值的单位是千字节每秒。加上一个后缀M表示兆字节每秒。也接受后缀k，但是没有效果。合法的值在32千字节每秒到1024兆字节每秒之间。

其目标是限制在运行服务器上的ux_basebackup产生的影响。

这个选项会影响数据目录的传输。如果收集方法是fetch时，只有WAL文件受到影响。

-R
--write-recovery-conf

在输出目录中（或者当使用tar格式时再基础归档文件中）写一个最小的recovery.conf来简化设置一个备用服务器。recovery.conf文件将记录连接设置（如果有）以及ux_basebackup所使用的复制槽，这样流复制后面就会使用相同的设置。

-T *olddir=newdir*
--tablespace-mapping=*olddir=newdir*

在备份期间将目录*olddir*中的表空间重定位到*newdir*中。为使之有效，*olddir*必须正好匹配表空间所在的路径（但如果备份中没有包含*olddir*中的表空间也不是错误）。*olddir*和*newdir*必须是绝对路径。如果路径中包含了一个=符号，可用反斜线对它转义。对于多个表空间可以多次使用这个选项。

如果以这种方法重定位一个表空间，主数据目录中的符号链接会被更新成指向新位置。因此新数据目录已经可以被一个所有表空间位于更新后位置的新服务器实例使用。

--waldir=*waldir*

指定用于预写式日志目录的位置。*waldir*必须是绝对路径。只有当备份是普通文件模式时才能指定事务日志目录。

-X *method*
--wal-method=*method*

指定收集预写日志的方法，在备份中包括所需的预写式日志文件（WAL文件）。这包括所有在备份期间产生的预写日志。除非指定了方法none，可以在收集的目录中直接启动一个uxtmaster，而不需要参考日志归档，因此得到一个完全独立的备份。

我们支持下列收集预写式日志的方法：

n
none

不要在备份中包括预写式日志。

f
fetch

在备份末尾收集预写式日志文件。因此，有必要把wal-keep-segments参数设置得足够高，这样在备份末尾之前日志不会被移除。如果在要传输日志时它已经被轮转，备份将失败并且是不可用的。

预写式日志文件将被写入到base.tar文件。

s
stream

在创建备份时对事务日志进行流处理。这将开启一个到服务器的第二连接并且在运行备份时并行地开始流处理事务日志。因此，它将使用最多两个由max-wal-senders参数配置的连接。只要客户端能保持接收预写式日志，使用这种模式不需要在主控机上保存额外的预写式日志。

预写式日志文件被写入到一个名为ux_wal.tar的单独文件。

这个值是默认值。

-z
--gzip

启用对tar文件输出的gzip压缩，使用默认的压缩级别。只有使用tar格式时压缩才可用，并且会在所有tar文件名后面自动加上后缀.gz后缀。

-Z *level*
--compress=*level*

启用对tar文件输出的gzip压缩，并且制定压缩机别（0 到 9，0 是不压缩，9 是最佳压缩）。只有使用tar格式时压缩才可用，并且会在所有tar文件名后面自动加上后缀.gz后缀。

下列命令行选项控制备份的生成和程序的运行。

-c *fast|spread*
--checkpoint=*fast|spread*

将检查点模式设置为fast（立刻）或spread（默认值）。

-C
--create-slot

这个选项会导致在开始备份前创建一个由--slot选项指定名称的复制槽。如果槽已经存在则会发生错误。

-l *label*
--label=*label*

为备份设置标签。如果没有指定，将使用一个默认值“ux_basebackup base backup”。

-n
--no-clean

默认情况下，当ux_basebackup中止并出现错误时，它会在发现它无法完成任务之前删除它可能创建的所有目录（例如数据目录和预写日志目录）。该选项禁止更改，因为对于调试非常有用。

请注意，表空间目录不会以任何方式清理。

-N
--no-sync

默认情况下，ux_basebackup将等待所有文件被安全地写到磁盘上。这个选项导致ux_basebackup无需等待而返回，节省时间但意味着后续的操作系统崩溃可能会导致基础备份损坏。通常，此选项对测试非常有用，但在创建生产安装时不应使用此选项。

-P
--progress

启用进度报告。启用这个选项将在备份期间发送进度报告。由于数据库可能在备份期间改变，这仅仅是一个大致进度并且可能不会完全达到100%。特别是，当WAL日志被包括在备份中时，总数据量无法预先估计，在这种情况中估计的目标尺寸会在它传递不带WAL的总估值后增加。

当这个选项被启用时，备份将从枚举整个数据库的大小开始，然后返回并发送实际内容。这可能使备份需要多花一些时间，特别是它在发送第一个数据之前花费的时间最长。

-S *slotname*
--slot=*slotname*

指定WAL流使用的复制槽。这个选项只能和-X　stream一起使用。如果该基础备份准备用作一个使用复制槽的流复制备用服务器，它应该使用recovery.conf中相同的复制槽名称。那样就可以确保服务器不会移除基础备份结束和流复制开始之前必要的WAL数据。

如果未指定此选项并且服务器支持临时复制槽，则自动使用临时复制槽进行WAL流式传输。

-v
--verbose

启用冗长模式。将在启动和关闭期间输出一些额外步骤，并且如果进度报告也被启用，还会显示当前正在被处理的确切文件名。

--no-slot

此选项禁止在备份过程中创建临时复制槽，即使受到服务器支持。

如果在使用日志流时未使用选项-S给出插槽名称，则会默认创建临时复制槽。

此选项的主要目的是在服务器没有空闲复制槽时允许进行基本备份。通常情况下，使用复制槽是首选，因为它可以防止备份期间服务器删除所需的WAL。

--no-verify-checksums

如果在取基础备份的服务器上启用了校验码验证，则禁用校验码验证。

默认情况下，校验码会被验证并且校验码失败将会导致一种非零的退出状态。不过，基础备份在这种情况下将不会被移除，就好像使用了--no-clean选项一样。校验和验证失败也将报告在ux_stat_database视图中。

下列命令行选项控制数据库连接参数：

-d *connstr*
--dbname=*connstr*

指定用于连接到服务器的参数。

为了和其他客户端应用一致，该选项被称为--dbname。但是因为ux_basebackup并不连接到集簇中的任何特定数据库，连接字符串中的数据库名将被忽略。

-h *host*
--host=*host*

指定运行服务器的机器的主机名。如果该值以一个斜线开始，它被用作 Unix 域套接字的目录。默认值取自UXHOST环境变量（如果设置），否则会尝试一个 Unix 域套接字连接。

-p *port*
--port=*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。默认用UXPORT环境变量中的值（如果设置），或者一个编译在程序中的默认值。

-s *interval*
--status-interval=*interval*

指定发送回服务器的状态包之间的秒数，更容易地监控服务器的进度。0表示完全禁用这种周期性的状态更新，当服务器请求更新时，仍然会发送更新，避免超时导致的断开连接。默认值是10秒。

-U *username*
--username=*username*

要作为哪个用户连接。

-w
--no-password

永远不要发出密码提示。如果服务器需要密码验证，而通过其他方式(如.pgpass文件)无法获得密码，则连接尝试将失败。此选项在没有用户可以输入密码的批处理作业和脚本中很有用。

-W
--password

强制ux_basebackup在连接到一个数据库之前提示输入密码。

这个选项不是必需的，因为如果服务器要求输入密码，ux_basebackup将自动提示输入密码。但是，ux_basebackup将浪费一次连接尝试来发现服务器想要一个密码。在某些情况下建议用-W来避免额外的连接尝试。

其他选项也可用：

-V
--version

打印ux_basebackup版本并退出。

-?
--help

显示有关ux_basebackup命令行参数的帮助并退出。

1.9.4. 环境变量

UX_COLOR

规定在诊断消息中是否使用颜色。可选的值为always，auto，never。

和大部分其他UXDB工具相似，这个工具也使用libuxsql支持的环境变量。

1.9.5. 注解

在备份开始时，需要在获取备份的服务器上写入一个检查点。如果未使用选项--checkpoint=fast，则可能需要一些时间，在这段时间内，ux_basebackup将显示为空闲状态。

备份包括数据目录和表空间中的所有文件：配置文件以及第三方放置在该目录中的附加文件（除了由UXDB管理的某些临时文件）。但是只复制常规文件和目录，除了保留用于表空间的符号链接。指向UXDB已知的某些目录的符号链接被复制为空目录。其他符号链接和特殊的设备文件会被跳过。

表空间默认以普通格式备份到与它们在服务器上相同的路径中，除非使用了--tablespace-mapping选项。如果没有这个选项并且表空间正在使用，一台服务器进行普通格式基础备份时将无法工作，因为备份必须要写入到与原始表空间相同的目录。

在使用tar格式模式时，用户应在启动UXDB服务器前解压每一个tar文件。如果有额外的表空间，用于它们的tar文件需要被解压到正确的位置。在这种情况下，服务器将根据base.tar中的tablespace_map文件的内容为那些表空间创建符号链接。

如果在源集簇上启用了组权限，在plain以及tar模式中ux_basebackup将保留组权限。

1.9.6. 示例

要创建服务器mydbserver的一个基础备份并将它存储在本地目录/home/uxdb/uxdbinstall/dbsql/data中：

$ **ux_basebackup -h mydbserver -D /home/uxdb/uxdbinstall/dbsql/data**

要创建本地服务器的一个备份，为其中每一个表空间产生一个压缩过的tar文件，并且将它存储在目录backup中，在运行期间显示一个进度报告：

$ **ux_basebackup -D backup -Ft -z -P**

要创建一个单一表空间本地数据库的备份并且使用bzip2压缩它：

$ **ux_basebackup -D - -Ft -X fetch | bzip2 > backup.tar.bz2**

（如果在该数据库中有多个表空间，这个命令将失败）。

要创建一个本地数据库的备份，其中/opt/ts中的表空间被重定位到./backup/ts：

$ **ux_basebackup -D backup/data -T /opt/ts=$(pwd)/backup/ts**

1.10. uxbench

uxbench — 在UXDB上运行一个基准测试

1.10.1. 用法

uxbench -i [*option...*] [*dbname*]

uxbench [*option...*] [*dbname*]

1.10.2. 描述

uxbench是一种在UXDB上运行基准测试的简单程序。它可能在并发的数据库会话中一遍一遍地运行相同序列的SQL命令，并且计算平均事务率（每秒的事务数）。默认情况下，uxbench会测试一种基于TPC-B但是要更宽松的场景，其中在每个事务中涉及五个SELECT、UPDATE以及INSERT命令。但是，通过编写自己的事务脚本文件很容易用来测试其他情况。

uxbench的典型输出像这样：

```
transaction type: <builtin: TPC-B (sort of)>
scaling factor: 10
query mode: simple
number of clients: 10
number of threads: 1
number of transactions per client: 1000
number of transactions actually processed: 10000/10000
tps = 85.184871 (including connections establishing)
tps = 85.296346 (excluding connections establishing)
```

前六行报告一些最重要的参数设置。接下来的行报告完成的事务数以及预期的事务数（后者就是客户端数量与每个客户端事务数的乘积），除非运行在完成之前失败，这些值应该是相等的（在-T模式中，只有实际的事务数会被打印出来）。最后两行报告每秒的事务数，分别代表包括和不包括开始数据库会话所花时间的情况。

默认的类型TPC-B事务测试要求预先设置好特定的表。可以使用-i（初始化）选项调用uxbench来创建并且填充这些表（当你在测试一个自定义脚本时，不需要这一步，但要按自定义测试的需要做一些设置工作）。初始化方法如下：

uxbench -i [*other-options*] *dbname*

*dbname*是要在其中进行测试的预先创建好的数据库的名称（你可能还需要-h、-p或-U选项来指定如何连接到数据库服务器）。

> **注意**
>
> uxbench -i会创建四个表uxbench_accounts、uxbench_branches、uxbench_history以及uxbench_tellers，如果同名表已经存在会被先删除。如果你已经有同名表，一定注意要使用另一个数据库！

在默认的情况下“比例因子”为 1，这些表初始包含的行数为：

```
table               # of rows
---------------------------------
uxbench_branches       1
uxbench_tellers        10
uxbench_accounts       100000
uxbench_history        0
```

你可以使用-s（比例因子）选项增加行的数量。-F（填充因子）选项也可以在这里使用。

完成了必要的设置后，就可以用不包括-i的命令运行基准，也就是：

uxbench [*options*] *dbname*

在近乎所有的情况中，你将需要一些选项来做一次有效的测试。最重要的选项是-c（客户端数量）、 -t（事务数量）、-T（时间限制）以及-f（指定一个自定义脚本文件）。

1.10.3. 选项

下面分成三个部分：数据库初始化期间使用的选项、运行基准测试时使用的选项、两种情况下都有用的选项。

1.10.3.1. 初始化选项

uxbench接受下列命令行初始化参数：

-i
--initialize

要求调用初始化模式。

-I *init_steps*
--init-steps=init_steps

只执行选出的一组普通初始化步骤。init_steps指定要被执行的初始化步骤，每一个步骤使用一个字符代表。每一个步骤都以指定的顺序被调用。默认是dtgvp。可用的步骤如下所示。

d（删除）

删除任何已有的uxbench表。

t（创建表）

创建标准uxbench场景使用的表，即uxbench_accounts、uxbench_branches、uxbench_history以及uxbench_tellers。

g（生成数据）

生成数据并且装入到标准的表中，替换掉已经存在的任何数据。

v（清理）

在标准的表上调用VACUUM。

p（创建主键）

在标准的表上创建主键索引。

f（创建外键）

在标准的表之间创建外键约束（注意这一步默认不会被执行）。

-F *fillfactor*
--fillfactor=*fillfactor*

用给定的填充因子创建表uxbench_accounts、uxbench_tellers以及uxbench_branches。默认是100。

-n
--no-vacuum

初始化后不执行清理。

-q
--quiet

把记录切换到安静模式，只是每 5 秒产生一个进度消息。默认的记录会每 100000 行打印一个消息，导致每秒输出很多行消息（特别是硬件设备好的情况下）。

--running-mode = standard | compatible | mysql

运行模式，默认模式是standard。standard表示标准模式，compatible表示兼容模式，mysql表示mysql模式。

-s *scale_factor*
--scale=*scale_factor*

将生成的行数乘以比例因子。例如，-s 100将在uxbench_accounts表中创建 10,000,000 行。默认为 1。当比例为 20,000 或更高时，用来保存账号标识符的列（*aid*列）将切换到使用更大的整数（bigint），这样才能足以保存账号标识符。

--foreign-keys

在标准的表之间创建外键约束。

--index-tablespace=*index_tablespace*

在指定的表空间而不是默认表空间中创建索引。

--tablespace=*tablespace*

在指定的表空间而不是默认表空间中创建表。

--unlogged-tables

把所有的表创建为非日志记录表而不是永久表。

--column-tables

把所有的表创建为列存表。

1.10.3.2. 基准选项

uxbench接受下列命令行基准参数：

-b *scriptname[@weight]*
--builtin=*scriptname[@weight]*

把指定的内建脚本加入到要执行的脚本列表中。@之后是一个可选的整数权重，它允许调节抽取该脚本的可能性。如果没有指定，它会被设置为1。可用的内建脚本有：tpcb-like、simple-update和select-only。这里也接受内建名称无歧义的前缀缩写。如果用上特殊的名字list，将会显示内建脚本的列表并且立刻退出。

-c *clients*
--client=*clients*

模拟的客户端数量，也就是并发数据库会话数量。默认为1。

-C
--connect

为每一个事务建立一个新连接，而不是只为每个客户端会话建立一个连接。这对于度量连接开销有用。

-d
--debug

打印调试输出。

-D *varname=value*
--define=*varname=value*

定义由自定义脚本使用的变量。允许多个-D选项。

-f *filename[@weight]*
--file=*filename[@weight]*

把一个从*filename*读到的事务脚本加入到被执行的脚本列表中。@后面是一个可选的整数权重，它允许调节抽取该测试的可能性。

-H *password*
--hidden=*password*

启动uxbench时在当前目录生成日志文件，文件名为uxbenchxxxxx.log(xxxxx为距离1970年1月1日的秒数)。*password*为服务器密码。

-j *threads*
--jobs=*threads*

uxbench中的工作者线程数量。在多CPU机器上使用多于一个线程会有帮助。客户端会尽可能均匀地分布到可用的线程上。默认为1。

-l
--log

把每一个事务的信息写到日志文件中。

-L *limit*
--latency-limit=*limit*

对持续超过*limit*毫秒的事务进行独立的计数和报告，这些事务被认为是迟到了的事务。

在使用限流措施时（--rate=...），滞后于计划超过*limit*毫秒并且因此没有希望满足延迟限制的事务根本不会被发送给服务器。这些事务被认为是被跳过的事务，它们会被单独计数并且报告。

-M *querymode*
--protocol=*querymode*

要用来提交查询到服务器的协议：

- simple：使用简单查询协议。
- extended使用扩展查询协议。
- prepared：使用带预备语句的扩展查询语句。

在prepared模式中，uxbench重用从第二次查询迭代开始的语法分析结果，因此uxbench运行速度比其他模式快。

默认是简单查询协议。

-n
--no-vacuum

在运行测试前不进行清理。如果你在运行一个不包括标准的表uxbench_accounts、uxbench_branches、uxbench_history和uxbench_tellers的自定义测试场景时，这个选项是必需的。

-N
--skip-some-updates

运行内建的简单更新脚本。这是-b simple-update的简写。

-P *sec*
--progress=*sec*

每*sec*秒显示进度报告。该报告包括运行了多长时间、从上次报告以来的tps、事务延迟的平均值和标准偏差。如果低于限流值（-R），延迟会相对于事务预定的开始时间（而不是事务实际的开始时间）计算，因此其中也包括了平均调度延迟时间。

-r
--report-latencies

在基准结束后，报告每个命令的每条语句的平均等待时间（从客户端的角度来说是执行时间）。

-R *rate*
--rate=*rate*

按照指定的速率执行事务而不是尽可能快地执行（默认行为）。该速率以tps（每秒事务数）形式给定。如果目标速率高于最大可能速率，则该速率限制不会影响结果。

该速率的目标是按照泊松分布的调度时间线开始事务。预计的开始时间表会基于客户端第一次开始的时间（而不是上一个事务结束的时间）前移。这种方法意味着当事务超过它们的原定结束时间时，更迟的事务有机会再次追赶上来。

当限流措施被激活时，运行结束时报告的事务延迟是从预订的开始时间计算而来的，因此它包括每一个事务不得不等待前一个事务结束所花的时间。该等待时间被称作调度延迟时间，并且它的平均值和最大值会被单独报告。关于实际事务开始时间的事务延迟（即在数据库中执行事务所花的时间）可以用报告的延迟减去调度延迟时间计算得到。

如果把--latency-limit和--rate一起使用，当一个事务在前一个事务结束时已经超过了延迟限制时，它可能会滞后非常多，因为延迟是从计划的开始时间计算得来。这类事务不会被发送给服务器，而是一起被跳过并且被单独计数。

调度延迟时间较长时表示系统无法用选定的客户端和线程数按照指定的速率处理事务。当平均的事务执行时间超过每个事务之间的调度间隔时，每一个后续事务将会落后更多，并且随着测试运行时间越长，调度延迟时间将持续增加。发生这种情况时，将不得不降低指定的事务速率。

-s *scale_factor*
--scale=*scale_factor*

在uxbench的输出中报告指定的比例因子。对于内建测试，这并非必需；正确的比例因子将通过对uxbench_branches表中的行计数来检测。不过，当只测试自定义基准（-f选项）时，比例因子将被报告为1。

-S
--select-only

执行内建的select-only脚本。是-b select-only简写形式。

-t *transactions*
--transactions=*transactions*

每个客户端运行的事务数量。默认为10。

-T *seconds*
--time=*seconds*

运行测试*seconds*秒，而不是为每个客户端运行固定数量的事务。-t和-T是互斥的。

-v
--vacuum-all

在运行测试前清理所有四个标准的表。在没有用-n以及-v时，uxbench将清理uxbench_tellers和uxbench_branches表，并且截断uxbench_history。

--aggregate-interval=*seconds*

聚集区间的长度（以秒计）。可以只与-l选项一起使用。通过这个选项，日志会包含每个区间的总结。

--log-prefix=*prefix*

为由--log创建的日志文件的文件名前缀。默认是uxbench_log。

--progress-timestamp

当显示进度（选项-P）时，使用一个时间戳（Unix 时间）取代从运行开始的秒数。单位是秒，在小数点后是毫秒精度。这可以有助于比较多种工具生成的日志。

--random-seed=SEED

设置随机数生成器种子。为系统的随机数生成器提供种子，然后随机数生成器会产生一个初始生成器状态序列，每一个线程一个状态。SEED的值可以是：time（默认值，种子基于当前时间）、rand（使用一种强随机源，如果没有可用的源则失败）或者一个无符号十进制整数值。一个uxbench脚本中会显式（random...函数）地或者隐式地（如--rate使用随机数生成器调度事务）调用随机数生成器。在被明确设置时，用作种子的值会显示在终端上。 还可以通过环境变量UXBENCH_RANDOM_SEED提供用于SEED的值。为了确保所提供的种子影响所有可能的使用，把这个选项放在第一位或者使用环境变量。

明确地设置种子允许准确地再生一个uxbench运行，对随机数而言。因为随机状态是针对每个线程管理，这意味着如果每一个线程有一个客户端并且没有外部或者数据依赖，则对于一个相同的调用就会有完全相同的uxbench运行。从一种统计的角度来看，再生运行不是什么好主意，因为它能隐藏性能可变性或者不正当地改进性能，即通过命中前一次运行的相同页面来改进性能。不过，它也可以对调试起到很大帮助作用，例如重新运行一种导致错误的棘手用例。请善用。

--sampling-rate=*rate*

采样率，在写入数据到日志时被用来减少日志产生的数量。如果给出这个选项，只有指定比例的事务被记录。1.0表示所有事务都将被记录，0.05表示只有5%的事务会被记录。

在处理日志文件时，记得要考虑这个采样率。例如，当计算tps值时，你需要相应地乘以这个数字（例如，采样率是0.01，你将只能得到实际TPS的1/100）。

1.10.3.3. 普通选项

uxbench接受下列命令行普通参数：

-h *hostname*
--host=*hostname*

数据库服务器的主机名。

-p *port*
--port=*port*

数据库服务器的端口号。

-U *login*
--username=*login*

要作为哪个用户连接。

-V
--version

打印uxbench版本并退出。

-?
--help

显示有关uxbench命令行参数的信息，并且退出。

1.10.4. 退出状态

成功运行将以状态0退出。退出状态为1表示静态问题，如无效的命令行选项。运行过程中的错误，例如数据库错误或脚本中的问题将导致退出状态为2。 在后一种情况下，uxbench将打印部分结果。

1.10.5. 注解

1.10.5.1. 在uxbench中实际执行的“事务”是什么？

uxbench执行从指定列表中随机选中的测试脚本。它们包括带有-b的内建脚本和带有-f的用户提供的自定义脚本。每一个脚本可以在其后用@指定一个相对权重，这样可以更改该脚本的抽取概率。默认权重是1。权重为0的脚本会被忽略。

默认的内建事务脚本（也会被-b tpcb-like调用）会在每个事务上发出七个从aid、tid、bid和balance中随机选择的命令。该场景来自于TPC-B基准，但并不是真正的TPC-B，只是名称类似。

1. BEGIN;
2. UPDATE uxbench_accounts SET abalance = abalance + :delta WHERE aid = :aid;
3. SELECT abalance FROM uxbench_accounts WHERE aid = :aid;
4. UPDATE uxbench_tellers SET tbalance = tbalance + :delta WHERE tid = :tid;
5. UPDATE uxbench_branches SET bbalance = bbalance + :delta WHERE bid = :bid;
6. INSERT INTO uxbench_history (tid, bid, aid, delta, mtime) VALUES (:tid, :bid, :aid, :delta, CURRENT_TIMESTAMP);
7. END;

如果选择simple-update内建脚本（还有-N），第4和5步不会被包括在事务中。这将避免更新那些表中的内容，但是会让该测试用例更符合TPC-B基准。

如果选择select-only内建脚本（还有-S），只会发出SELECT。

1.10.5.2. 自定义脚本

uxbench支持通过从一个文件中读取事务脚本替换默认的事务脚本（-f选项）来运行自定义的基准场景。在这种情况中，一个事务就是一个脚本文件的一次执行。

脚本文件包含一个或者多个被分号终结的SQL命令。空行以及以--开始的行会被忽略。脚本文件也可以包含元命令。

> **注意**
>
> 需要用分号来分隔连续的SQL命令（如果SQL命令后面跟着一个元命令则不需要一个分号）。

脚本文件有一种简单的变量替换功能。变量名必须由用命令行的-D选项设置，或者按下文所说的使用元命令设置。除了用-D命令行选项预先设置的任何变量之外，还有一些被自动预先设置

的变量，如下表所示。一个用-D为这些变量值指定的值会优先于自动的预设值。一旦被设置，可以在SQL命令中写:*variablename*来插入一个变量的值。当运行多于一个客户端会话时，每一个会话拥有它自己的变量集合。

表 1.1. 自动变量

变量	描述
client_id	标识客户端会话的唯一数字（从零开始）
default_seed	默认在哈希函数中使用的种子
random_seed	随机数生成器种子（除非用-D重载）
scale	当前的缩放因子

脚本文件元命令开始于一个反斜线（\）并且通常延伸到行的末尾，尽管可以通过写入反斜杠-返回继续附加行。一个元命令和它的参数用空白分隔。支持的元命令如下：

\gset *[prefix]*

此命令可以用于结束 SQL 查询，代替终止分号 (;)。

当使用此命令时，前面的 SQL 查询预期返回一行，存储变量名的列在列名后面，如果已经提供的话，则以*prefix*作为前缀。

下面的示例将第一个查询中的最终帐户余额放入变量*abalance*， 并且用第三个查询中的整数填充变量*p_two*和*p_three*。 第二个查询的结果将被丢弃。

```
UPDATE uxbench_accounts
  SET abalance = abalance + :delta
  WHERE aid = :aid
  RETURNING abalance \gset
-- compound of two queries
SELECT 1 \;
SELECT 2 AS two, 3 AS three \gset p_
```

\if *expression*
\elif *expression*
\else
\endif

这一组命令实现了可嵌套的条件块，类似于uxsql的*\if expression*。条件表达式与\set的相同，非零值会被解释为真。

\set *varname expression*

把变量*varname*为一个从*expression*计算得到的值。该表达式可以会包含整数常量、（例如5432）、双精度常量（例如3.14159）对变量:*variablename*的引用、一元（+、-）或者二元运算符（+、-、*、/、%）（保留它们通常的优先级、结合性和圆括号）。

函数和大部分操作符在NULL输入上会返回NULL。

对于条件目的，非零数字值是TRUE，数字零值以及NULL是FALSE。

太大或太小的整数和双常量，以及整数算术运算符（+,-, * 和 /）会引发溢出错误。

在没有为CASE提供最终的ELSE子句时，默认值是NULL。

示例：

```
\set ntellers 10 * :scale
\set aid (1021 * random(1, 100000 * :scale)) % \
        (100000 * :scale) + 1
\set divx CASE WHEN :x <> 0 THEN :y/:x ELSE NULL END
```

\sleep *number* [us | ms | s]

导致脚本执行休眠指定的时间，时间的单位可以是微秒（us）、毫秒（ms）或者秒（s）。如果单位被忽略，则默认值是秒。*number*要么是一个整数常量，要么是一个引用了具有整数值的变量的:*variablename*。

示例：

```
\sleep 10 ms
```

\setshell *varname command* [*argument* ...]

用给定的*argument*设置变量*varname*为shell命令*command*的结果。该命令必须通过它的标准输出返回一个整数值。

*command*和每个*argument*要么是一个文本常量，要么是一个引用了一个变量的:*variablename*。如果你想要使用以冒号开始的*argument*，需要在*argument*的开头写一个额外的冒号。

示例：

```
\setshell variable_to_be_assigned command literal_argument :variable ::literal_starting_with_colon
```

\shell *command* [*argument* ...]

与\setshell相同，但是结果被抛弃。

示例：

```
\shell command literal_argument :variable ::literal_starting_with_colon
```

1.10.5.3. 内建操作符

表 1.2 “按优先级升序排列的uxbench操作符”中列举的算数、按位、比较以及逻辑操作符都被编译到了uxbench中并且可以被用于\set中出现的表达式中。

表 1.2. 按优先级升序排列的uxbench操作符

操作符	简介	示例	结果
OR	逻辑或	5 or 0	TRUE
AND	逻辑与	3 and 0	FALSE
NOT	逻辑非	not false	TRUE

操作符	简介	示例	结果
IS [NOT] (NULL|TRUE|FALSE)	值测试	1 is null	FALSE
ISNULL|NOTNULL	空测试	1 notnull	TRUE
=	等于	5 = 4	FALSE
<>	不等于	5 <> 4	TRUE
!=	不等于	5 != 5	FALSE
<	小于	5 < 4	FALSE
<=	小于等于	5 <= 4	FALSE
>	大于	5 > 4	TRUE
>=	大于等于	5 >= 4	TRUE
|	整数按位OR	1 | 2	3
#	整数按位XOR	1 # 3	2
&	整数按位AND	1 & 3	1
~	整数按位NOT	~ 1	-2
<<	整数按位左移	1 << 2	4
>>	整数按位右移	8 >> 2	2
+	加	5 + 4	9
-	减	3 - 2.0	1.0
*	乘	5 * 4	20
/	除（整数会截断结果）	5 / 3	1
%	取模	3 % 2	1
-	取负	- 2.0	-2.0

1.10.5.4. 内建函数

下表列出的函数被编译在uxbench中，并且可能被用在出现于\set的表达式中。

表 1.3. uxbench 函数

函数	返回类型	描述	示例	返回值
abs(*a*)	和*a*相同	绝对值	abs(-17)	17
debug(*a*)	和*a*相同	把*a*打印到stderr，并且返回*a*	debug(5432.1)	5432.1
double(*i*)	double	造型成double	double(5432)	5432.0
exp(*x*)	double	指数	exp(1.0)	2.718281828459045
greatest(*a* [, ...])	如果任何*a*是double则为double，否则为integer	参数中的最大值	greatest(5, 4, 3, 2)	5
hash(*a* [, *seed*])	integer	hash_murmur2()的别名	hash(10, 5432)	-5817877081768721676

函数	返回类型	描述	示例	返回值
hash_fnv1a(*a* [, *seed*])	integer	FNV-1a hash	hash_fnv1a(10, 5432)	-7793829335365542153
hash_murmur2(*a* [, *seed*])	integer	MurmurHash2 hash	hash_murmur2(10, 5432)	-5817877081768721676
int(*x*)	integer	造型成int	int(5.4 + 3.8)	9
least(*a* [, ...])	如果任何*a*是double则为double，否则为integer	参数中的最小值	least(5, 4, 3, 2.1)	2.1
ln(*x*)	double	自然对数	ln(2.718281828459045)	1.0
mod(*i*, *j*)	integer	取模	mod(54, 32)	22
pi()	double	常量PI的值	pi()	3.14159265358979323846
pow(*x*, *y*), power(*x*, *y*)	double	求幂	pow(2.0, 10), power(2.0, 10)	1024.0
random(*lb*, *ub*)	integer	[lb, ub]中均匀分布的随机整数	random(1, 10)	1和10之间的一个整数
random_exponential(*lb*, *ub*, *parameter*)	integer	[lb, ub]中指数分布的随机整数	random_exponential(1, 10, 3.0)	1和10之间的一个整数
random_gaussian(*lb*, *ub*, *parameter*)	integer	[lb, ub]中高斯分布的随机整数	random_gaussian(1, 10, 2.5)	1和10之间的一个整数
random_zipfian(*lb*, *ub*, *parameter*)	integer	[lb, ub]中Zipfian分布的随机整数	random_zipfian(1, 10, 1.5)	1和10之间的一个整数
sqrt(*x*)	double	平方根	sqrt(2.0)	1.414213562

random函数使用均匀分布生成值，即所有的值都以相等的概率从指定的范围中抽出。random_exponential、random_gaussian以及random_zipfian函数要求一个额外的 double 参数，它决定分布的精确形状。

- 对于指数分布，*parameter*通过在*parameter*处截断一个快速下降的指数分布来控制分布，然后投影到边界之间的整数上。确切地来说，

  ```
  f(x) = exp(-parameter * (x - min) / (max - min + 1)) / (1 - exp(-parameter))
  ```

  然后*min*和*max*之间（包括两者）的值*i*会被以概率f(i) - f(i + 1)抽出。

  直观上，*parameter*越大，接近*min*的值会被越频繁地访问，并且接近*max*的值会被越少访问。*parameter*越接近0，访问分布会越均匀。该分布的近似值是范围中当时被抽取 *parameter*% 次接近*min*的最频繁的1%值。*parameter*值必须严格为正。

- 对于高斯分布，区间被映射到一个在左边-parameter和右边+parameter截断的标准正态分布（经典钟型高斯曲线）。区间中间的值更可能被抽到。准确地说，如果PHI(x)是标准正态分布的累计分布函数，均值mu定义为(max + min) / 2.0，有

```
f(x) = PHI(2.0 * parameter * (x - mu) / (max - min + 1)) /
       (2.0 * PHI(parameter) - 1)
```

则*min*和*max*（包括两者）之间的值*i*被抽出的概率是：f(i+0.5)-f(i-0.5)。直观上，*parameter*越大，靠近区间终端的值会被越频繁地抽出，并且靠近上下界两端的值会被更少抽出。大约 67% 的值会被从中间1.0/parameter的地方抽出，即均值周围0.5/parameter的地方。并且 95% 的值会被从中间2.0/parameter的地方抽出，即均值周围1.0/parameter的地方。例如，如果*parameter*是4.0，67% 的值会被从该区间的中间四分之一（1.0 / 4.0）抽出（即从3.0/8.0到5.0/8.0）。并且 95% 的值会从该区间的中间一半（2.0/4.0）抽出（第二和第三四分位）。为了 Box-Muller 变换的性能，*parameter*最小为 2.0。

- random_zipfian生成一个有界的Zipfian分布。*parameter*定义该分布有多么倾斜。*parameter*越大，绘制越接近间隔开头的值越频繁。分布是这样的，假设范围从1开始，绘制*k*与绘制*k+1*的概率之比为((*k*+1)/*k*)**_parameter_。例如，random_zipfian(1, ..., 2.5)生成值1大约(2/1)**2.5 = 5.66次，比 2 更频繁，它本身被产生(3/2)**2.5 = 2.76次，比3更频繁，依此类推。

  uxbench的实现是基于"Non-Uniform Random Variate Generation", Luc Devroye, p. 550-551,Springer 1986。 由于该算法的限制，*parameter*值限制范围为[1.001, 1000]。

哈希函数hash、hash_murmur2以及hash_fnv1a接受一个输入值和一个可选的种子参数。在没有提供种子的情况下，会使用:default_seed的值，该变量会被随机地初始化，除非用命令行的-D 选项重载。哈希函数可以被用于分散random_zipfian或random_exponential这样的随机函数的分布。例如，下列uxbench脚本模拟了社交媒体和博客平台上很常见的真实负载，其中少数账号产生了过量的负载。

```
\set r random_zipfian(0, 100000000, 1.07)
\set k abs(hash(:r)) % 1000000
```

在一些情况中需要几个不同的分布，它们彼此之间不相关并且隐式的随机数参数在此时就能派上用场，如下所示。

```
\set k1 abs(hash(:r, :default_seed + 123)) % 1000000
\set k2 abs(hash(:r, :default_seed + 321)) % 1000000
```

作为一个示例，内建的类TPC-B事务的全部定义是：

```
\set aid random(1, 100000 * :scale)
\set bid random(1, 1 * :scale)
\set tid random(1, 10 * :scale)
\set delta random(-5000, 5000)
BEGIN;
UPDATE uxbench_accounts SET abalance = abalance + :delta WHERE aid = :aid;
SELECT abalance FROM uxbench_accounts WHERE aid = :aid;
UPDATE uxbench_tellers SET tbalance = tbalance + :delta WHERE tid = :tid;
UPDATE uxbench_branches SET bbalance = bbalance + :delta WHERE bid = :bid;
INSERT INTO uxbench_history (tid, bid, aid, delta, mtime) VALUES (:tid, :bid, :aid, :delta,
 CURRENT_TIMESTAMP);
END;
```

这个脚本允许该事务的每一次迭代能够引用不同的、被随机选择的行（这个示例也展示了为什么让每一个客户端会话有自己的变量很重要—否则它们不会独立地接触不同的行）。

1.10.5.5. 为每个事务创建日志

使用-l选项但是不使用--aggregate-interval选项时，uxbench会把每一个事务的信息写入到一个日志文件。该日志文件将被命名为*prefix.nnn*，其中*prefix*默认为uxbench_log，而*nnn*是uxbench进程的PID。前缀可以用--log-prefix选项更改。如果-j选项是2或者更高（有多个工作者线程），那么每一个工作者线程将会有它自己的日志文件。第一个工作者的日志文件的命名将和标准的单工作者情况相同。其他工作者的额外日志文件将被命名为*prefix.nnn.mmm*，其中*mmm*是每个工作者的一个序列号，这种序列号从1开始。

日志的格式是：

client_id transaction_no time script_no time_epoch time_us [*schedule_lag*]

其中*client_id*表示哪个客户端会话运行该事务，*transaction_no*是那个会话已经运行了多少个事务的计数，*time*是以微秒计的总共用掉的事务时间，*script_no*标识了要使用哪个脚本文件（当用-f或者-b指定多个脚本时有用），而*time_epoch*/*time_us*是一个 Unix 纪元格式的时间戳以及一个显示事务完成时间的以微秒计的偏移量（适合于创建一个带有分数秒的 ISO 8601 时间戳）。 域*schedule_lag*是事务的预定开始时间和实际开始时间之间的差别，以微秒计。只有使用--rate选项时它才存在。当--rate和--latency-limit同时被使用时， 一个被跳过的事务的*time*会被报告为skipped。

这里是在单个客户端运行中生成的一个日志文件的片段：

```
0 199 2241 0 1175850568 995598
0 200 2465 0 1175850568 998079
0 201 2513 0 1175850569 608
0 202 2038 0 1175850569 2663
```

另一个示例使用的是--rate=100以及--latency-limit=5（注意额外的 *schedule_lag*列）：

```
0 81 4621 0 1412881037 912698 3005
0 82 6173 0 1412881037 914578 4304
0 83 skipped 0 1412881037 914578 5217
0 83 skipped 0 1412881037 914578 5099
0 83 4722 0 1412881037 916203 3108
0 84 4142 0 1412881037 918023 2333
0 85 2465 0 1412881037 919759 740
```

在这个示例中，事务82迟到了，因为它的延迟（6.173 ms）超过了 5ms限制。接下来的两个事务被跳过，因为它们在开始之前就已经迟到了。

在能够处理大量事务的硬件上运行一次长时间的测试时，日志文件可能变得非常大。--sampling-rate选项能被用来只记录事务的一个随机采样。

1.10.5.6. 聚合日志记录

通过--aggregate-interval选项，日志文件会使用一种不同的格式：

interval_start num_of_transactions latency_sum latency_2_sum min_latency max_latency
[*lag_sum lag_2_sum min_lag max_lag*]

其中*interval_start*是区间的开始（作为一个Unix纪元的时间戳）、*num_transactions*是该区间中的事务数、*sum_latency*是该区间中事务时延的总量、*sum_latency_2*是该区间中事务时延的平方和、*min_latency*是该区间中的最小时延、*max_latency*是该区间中的最大时延。接下来的字段*sum_lag*、*sum_lag_2*、*min_lag*以及*max_lag*只有在使用--rate选项时才存在。它们提供每个事务要等待前一个事务完成所花的时间的统计信息，即每个事务的计划启动时间与实际启动时间之间的差值。最后一个字段*skipped*只有在使用--latency-limit选项时才存在。它对因为启动过完被跳过的事务进行计数。每一个事务被计入在其提交时的区间中。

这里是一些输出示例：

```
1345828501 5601 1542744 483552416 61 2573
1345828503 7884 1979812 565806736 60 1479
1345828505 7208 1979422 567277552 59 1391
1345828507 7685 1980268 569784714 60 1398
1345828509 7073 1979779 573489941 236 1411
```

> **注意**
>
> 请注意，普通（未聚合）的日志文件显示每个事务使用了哪个脚本，聚合日志不包含索引。因此如果你需要针对每个脚本的数据，你需要自行聚合数据。

1.10.5.7. 语句延迟

通过-r选项，uxbench收集每一个客户端执行的每一个语句花费的事务时间。然后在基准完成后，它会报告这些值的平均值，作为每个语句的延迟。

对于默认脚本，输出看起来会像这样：

```
starting vacuum...end.
transaction type: <builtin: TPC-B (sort of)>
scaling factor: 1
query mode: simple
number of clients: 10
number of threads: 1
number of transactions per client: 1000
number of transactions actually processed: 10000/10000
latency average = 15.844 ms
latency stddev = 2.715 ms
tps = 618.764555 (including connections establishing)
tps = 622.977698 (excluding connections establishing)
statement latencies in milliseconds:
        0.002  \set aid random(1, 100000 * :scale)
        0.005  \set bid random(1, 1 * :scale)
        0.002  \set tid random(1, 10 * :scale)
        0.001  \set delta random(-5000, 5000)
        0.326  BEGIN;
        0.603  UPDATE uxbench_accounts SET abalance = abalance + :delta WHERE aid = :aid;
        0.454  SELECT abalance FROM uxbench_accounts WHERE aid = :aid;
        5.528  UPDATE uxbench_tellers SET tbalance = tbalance + :delta WHERE tid = :tid;
        7.335  UPDATE uxbench_branches SET bbalance = bbalance + :delta WHERE bid = :bid;
```

```
        0.371  INSERT INTO uxbench_history (tid, bid, aid, delta, mtime) VALUES (:tid, :bid, :aid, :delta,
CURRENT_TIMESTAMP);
        1.212  END;
```

如果指定了多个脚本文件，会为每一个脚本文件单独报告平均值。

注意为每个语句的延迟计算收集额外的时间信息会增加一些负荷。这将拖慢平均执行速度并且降低计算出的TPS。降低的总量会很显著地依赖于平台和硬件。对比使用和不适用延迟报告时的平均TPS值是评估时间开销是否明显的最佳方法。

1.10.5.8. 解决办法

使用uxbench产生各种完全没有意义的数据很容易，通过下面的方法可以帮你得到有意义的结论：

首先，永远不要相信任何只运行了几秒的测试。使用-t或-T选项让运行持续至少几分钟，这样可以用平均值去掉噪声。在一些情况中，你可能需要数小时来得到能重现的数字。多运行几次测试是一个好主意，这样可以看看数据能否重现。

对于默认的类TPC-B测试场景，初始化的比例因子（-s）应该至少和你想要测试的最大客户端数量一样大（-c），否则你将主要在度量更新争夺。在uxbench_branches表中只有-s行，并且每个事务都想更新其中之一，因此-c值超过-s将毫无疑问地导致大量事务被阻塞，等待其他事务处理。

默认的测试场景对初始化表的耗时非常敏感：表中死亡行和死亡空间的累积会改变结果。要理解结果，必须跟踪更新的总数以及何时发生清理。如果开启了自动清理，它可能会在度量的性能上产生不可预估的改变。

uxbench的一个限制是在尝试测试大量客户端会话时，它自身可能成为瓶颈。这可以通过在数据库服务器之外的一台机器上运行uxbench来缓解，不过必须是具有低网络延迟的机器。甚至可以在多个客户端机器上针对同一个数据库服务器并发地运行多个uxbench实例。

1.10.5.9. 安全性

如果不可信用户能够访问没有采用安全方案使用模式的数据库，不要在那个数据库中运行uxbench。uxbench使用非限定名称并且不会操纵搜索路径。

1.11. ux_config

ux_config — 获取已安装的UXDB的信息

1.11.1. 用法

ux_config [*option*...]

1.11.2. 描述

ux_config工具打印当前安装版本的UXDB的配置参数。它的设计目的之一是便于想与UXDB交互的软件包能够找到所需的头文件和库。

1.11.3. 选项

要使用ux_config，提供一个或多个下列选项：

--bindir

打印用户可执行文件的位置。例如使用这个选项来寻找uxsql程序。这通常也是ux_config程序所在的位置。

--docdir

打印文档文件的位置。

--htmldir

打印 HTML 文档文件的位置。

--includedir

打印客户端接口的C头文件的位置。

--pkgincludedir

打印其它C头文件的位置。

--includedir-server

打印用于服务器编程的C头文件的位置。

--libdir

打印对象代码库的位置。

--pkglibdir

打印动态可载入模块的位置，或者服务器可能搜索它们的位置（其它架构独立数据文件可能也被安装在这个目录）。

--localedir

打印区域支持文件的位置（如果在UXDB被编译时没有配置区域支持，这将是一个空字符串）。

--mandir

打印手册页的位置。

--sharedir

打印架构独立支持文件的位置。

--sysconfdir

打印系统范围配置文件的位置。

--uxxs

打印扩展makefile的位置。

--configure

打印当UXDB被配置编译时给予configure脚本的选项。这可以被用来重新得到相同的配置，或者找出是哪个选项编译了一个二进制包（不过注意二进制包通常包含厂商相关的自定补丁）。

--cc

打印用来编译UXDB的CC变量值。这显示被使用的C编译器。

--cppflags

打印用来编译UXDB的CPPFLAGS变量值。这显示在预处理时需要的C编译器开关（典型的是-I开关）。

--cflags

打印用来编译UXDB的CFLAGS变量值。这显示被使用的C编译器开关。

--cflags_sl

打印用来编译UXDB的CFLAGS_SL变量值。这显示被用来编译共享库的额外C编译器开关。

--ldflags

打印用来编译UXDB的LDFLAGS变量值。这显示链接器开关。

--ldflags_ex

打印用来编译UXDB的LDFLAGS_EX变量值。这只显示被用来编译可执行程序的链接器开关。

--ldflags_sl

打印用来编译UXDB的LDFLAGS_SL变量值。这只显示被用来编译共享库的链接器开关。

--libs

打印用来编译UXDB的LIBS变量值。这通常包含用于链接到UXDB中的外部库的-l开关。

--version

打印UXDB的版本。

-?
--help

显示有关ux_config命令行参数的帮助并退出。

如果给定多于一个选项，将按照相同的顺序打印信息，每行一项。如果没有给定选项，将打印所有可用信息，并带有标签。

1.11.4. 示例

查看当前UXDB的安装信息：

```
ux_config --configure
```

1.12. ux_dump

ux_dump — 把UXDB数据库抽取为一个脚本文件或其他归档文件

1.12.1. 用法

ux_dump [*connection-option...*] [*option...*] [*dbname*]

1.12.2. 描述

ux_dump是一种用于备份UXDB数据库的工具。即使数据库正在被并发使用，它也能创建一致的备份。ux_dump不阻塞其他用户访问数据库（读取或写入）。

ux_dump只转储单个数据库。要备份一个集群中对于所有数据库公共的全局对象（例如角色和表空间），应使用ux_dumpall。

转储可以被输出到脚本或归档文件格式。脚本转储是包含SQL命令的纯文本文件，它们可以用来重构数据库到它被转储时的状态。要从这样一个脚本恢复，将它传递给uxsql。脚本文件甚至可以被用来在其他机器和其他架构上重构数据库。在经过一些修改后，甚至可以在其他SQL数据库产品上重构数据库。

另一种可选的归档文件格式必须与ux_restore配合使用来重建数据库。它们允许ux_restore能选择恢复什么，或者甚至在恢复之前对条目重排序。归档文件格式被设计为在架构之间可移植。

当使用归档文件格式之一并与ux_restore组合时，ux_dump提供了一种灵活的归档和传输机制。ux_dump可以被用来备份整个数据库，然后ux_restore可以被用来检查归档并/或选择数据库的哪些部分要被恢复。最灵活的输出文件格式是“custom”格式（-Fc）和“directory”格式（-Fd）。它们允许选择和重排序所有已归档项、支持并行恢复并且默认是压缩的。“directory”格式是唯一一种支持并行转储的格式。

当运行ux_dump时，我们应该检查输出中有没有任何警告（打印在标准错误上），特别是考虑到下面列出的限制。

1.12.3. 选项

下列命令选项控制输出的内容和格式。

dbname

指定要被转储的数据库名。如果没有指定，将使用环境变量UXDATABASE。如果环境变量也没有设置，则使用指定给该连接的用户名。

-a
--data-only

只转储数据，而不转储模式（数据定义）。表数据、大对象和序列值都会被转储。

这个选项类似于指定--section=data，但是又不完全相同。

-b
--blobs

在转储中包括大对象。这是当--schema、--table或--schema-only被指定时的默认行为。因此，-b开关仅对于将大对象添加到已请求特定模式或表的转储中时有用。请注意，blob被视为数据，因此仅在使用--data-only时才会包含，但在使用--schema-only时不会包含。

-B
--no-blobs

排除转储中的大对象。

-c
--clean

在输出创建数据库对象的命令之前输出清除（删除）它们的命令（除非也指定了--if-exists，如果任何对象不存在于目的数据库中，恢复可能会产生一些不致命的错误消息）。

这个选项只对纯文本格式有意义。对于归档格式，你可以在调用ux_restore时指定该选项。

-C
--create

使得在输出的开始是一个创建数据库本身并且重新连接到被创建的数据库的命令（通过这种形式的一个脚本，在运行脚本之前你连接的是目标安装中的哪个数据库都没有关系）。如果也指定了--clean，脚本会在重新连接到目标数据库之前先删除它然后再重建。

通过--create，还会输出包括数据库的注释（如果有）以及与这个数据库相关的任何配置变量设置，也就是任何提到了这个数据库的ALTER DATABASE ... SET ...命令和ALTER ROLE ... IN DATABASE ... SET ...命令。该数据库本身的访问特权也会被转储，除非指定有--no-acl。

这个选项只对纯文本格式有意义。对于归档格式，你可以在你调用ux_restore时指定这个选项。

-E *encoding*
--encoding=*encoding*

以指定的字符集编码创建转储。在默认情况下，该转储会以该数据库的编码创建（另一种得到相同结果的方式是将UXCLIENTENCODING环境变量设置成想要的转储编码）。

-f *file*
--file=*file*

将输出发送到指定文件。对于基于输出格式的文件这个参数可以被忽略，在那种情况下将使用标准输出。不过对于目录输出格式必须给定这个参数，在目录输出格式中指定的是一个目录而不是一个文件。在这种情况中，该目录会由ux_dump创建并且之前必须不存在。

-F *format*
--format=*format*

选择输出的格式。*format*可以是下列之一：

p
plain

输出一个纯文本形式的SQL脚本文件（默认值）。

c
custom

输出一个适合于作为ux_restore输入的自定义格式归档。和目录输出格式一起，这是最灵活的输出格式，它允许在恢复时手动选择和排序已归档的项。这种格式在默认情况还会被压缩。

d
directory

输出一个适合作为ux_restore输入的目录格式归档。这将创建一个目录，其中每个被转储的表和大对象都有一个文件，外加一个所谓的目录文件，该文件以一种ux_restore能

读取的机器可读格式描述被转储的对象。一个目录格式归档能用标准 Unix 工具操纵，例如一个未压缩归档中的文件可以使用gzip工具压缩。这种格式默认情况下是被压缩的并且也支持并行转储。

t
tar

输出一个适合于输入到ux_restore中的tar格式归档。tar格式可以兼容目录格式，输出一个tar格式的归档会产生一个合法的目录格式归档。不过，tar格式不支持压缩。还有在使用tar格式时，表数据项的相对顺序不能在恢复过程中被更改。

-j *njobs*
--jobs=*njobs*

通过同时归档*njobs*个表来运行并行转储。这个选项缩减了转储的时间，但是它也增加了数据库服务器上的负载。你只能和目录输出格式一起使用这个选项，因为这是唯一一种让多个进程能在同一时间写其数据的输出格式。

ux_dump将打开*njobs* + 1 个到该数据库的连接，因此确保max-connections设置足够高以容纳所有的连接。

在运行一次并行转储时请求数据库对象上的排他锁可能导致转储失败。其原因是，ux_dump主控进程会在工作者进程将要稍后转储的对象上请求共享锁，以便确保在转储运行时不会有人删除它们并让它们出错。如果另一个客户端接着请求一个表上的排他锁，那个锁将不会被授予但是会被排入队列等待主控进程的共享锁被释放。因此，任何其他对该表的访问将不会被授予或者将排在排他锁请求之后。这包括尝试转储该表的工作者进程。如果没有任何防范措施，这可能会是一种经典的死锁情况。要检测这种冲突，ux_dump工作者进程使用NOWAIT选项请求另一个共享锁。 如果该工作者进程没有被授予这个共享锁，其他某人必定已经在同时请求了一个排他锁并且没有办法继续转储，因此ux_dump除了中止转储之外别无选择。

对于一个一致的备份，数据库服务器需要支持同步的快照，在UXDB 中引入了针对主服务器和备用服务器的特性 。有了这种特性，即便数据库客户端使用不同的连接，也可以保证他们看到相同的数据集。ux_dump -j使用多个数据库连接，它用主控进程连接到数据一次，并且为每一个工作者任务再一次连接数据库。如果没有同步快照特征，在每一个连接中不同的工作者任务将不能被保证看到相同的数据，这可能导致一个不一致的备份。

-n *pattern*
--schema=*pattern*

只转储匹配*pattern*的模式，这会选择模式本身以及它所包含的所有对象。当没有指定这个选项时，目标数据库中所有非系统模式都将被转储。多个模式可以通过书写多个-n开关来选择。另外，*pattern*参数可以被解释为一种根据uxsql的\d命令所用的相同规则编写的模式，这样多个模式也可以通过在该模式中书写通配字符来选择。在使用通配符时，如果需要阻止shell展开通配符需要谨慎引用该模式。

> **注意**
>
> 当-n被指定时，ux_dump不会尝试转储所选模式可能依赖的任何其他数据库对象。因此，无法保证一次指定模式转储的结果能够仅凭其本身被成功地恢复到一个干净的数据库中。
>
> 当-n被指定时，非模式对象（如二进制大对象）不会被转储。你可以使用--blobs开关将二进制大对象加回到该转储中。

-N *pattern*
--exclude-schema=*pattern*

不转储匹配*pattern*模式的任何模式。该模式被根据-n所用的相同规则被解释。-N可以被给定多次来排除匹配多个模式。

当-n和-N都被给定时，该行为是只转储匹配至少一个-n开关但是不匹配-N开关的模式。如果只有-N而没有-n，那么匹配-N的模式会被从一个正常转储中排除。

-o
--oids

转储对象标识符（OID）作为每个表数据的一部分。如果你的应用以某种方式引用OID列（例如在一个外键约束中），应使用这个选项。否则，这个选项不应该被使用。

-O
--no-owner

不输出设置对象拥有关系来匹配原始数据库的命令。默认情况下，ux_dump会发出**ALTER OWNER**或**SET SESSION AUTHORIZATION**语句来设置被创建的数据库对象的拥有关系。除非该脚本被一个超级用户（或是拥有脚本中所有对象的同一个用户）启动，这些语句都将会失败。要使一个脚本能够被任意用户恢复，但把所有对象的拥有关系都给这个用户，可指定-O。

这个选项只对纯文本格式有意义。对于归档格式，你可以在调用**ux_restore**时指定该选项。

--running-mode = standard | compatible | mysql

运行模式，默认模式是standard。standard表示标准模式，compatible表示兼容模式，mysql表示mysql模式。

-s
--schema-only

只转储对象定义（模式），而非数据。

这个选项是--data-only的逆选项。它和指定--section=pre-data --section=post-data相似，但是又不完全相同。

（不要把这个选项和--schema选项混淆，后者在“schema”的使用上有不同的含义）。

要为数据库中表的一个子集排除表数据，见--exclude-table-data。

-S *username*
--superuser=*username*

指定要在禁用触发器时使用的超级用户的用户名。只有使用--disable-triggers时，这个选项才相关（通常，最好省去这个选项，而作为超级用户来启动结果脚本来取而代之）。

-t *pattern*
--table=*pattern*

只转储名字匹配*pattern*的表，“pattern”还可以包括视图、物化视图、序列和外部表。通过写多个-t开关可以选择多个表。另外，*pattern*参数可以被解释为一种根据uxql的\d命令所用的相同规则编写的模式，这样多个表也可以通过在该模式中书写通配字符来选择。在使用通配符时，如果需要阻止 shell 展开通配符需要谨慎引用该模式。

当-t被使用时，-n和-N开关不会有效果，因为被-t选择的表将被转储而无视那些开关，并且非表对象将不会被转储。

> **注意**
>
> 当-t被指定时，ux_dump不会尝试转储所选表可能依赖的任何其他数据库对象。因此，无法保证一次指定表转储的结果能够仅凭其本身被成功地恢复到一个干净的数据库中。

-T *pattern*
--exclude-table=*pattern*

不转储名称匹配*pattern*的表。该参数被根据-t所用的相同规则被解释。-T可以被给定多次来排除匹配多个表。

当-t和-T都被给定时，该行为是只转储匹配至少一个-t开关但是不匹配-T开关的表。如果只有-T而没有-t，那么匹配-T的表会被从一个正常转储中排除。

--transform-table

将分布表/参考表转换成普通表。

-v
--verbose

启用详细输出模式。这将导致ux_dump向标准错误输出详细的对象注释以及转储文件的开始/停止时间，还有进度消息。

-V
--version

ux_dump版本并退出。

-x
--no-privileges
--no-acl

禁止转储特权（grant/revoke命令）。

-Z *0..9*
--compress=*0..9*

指定要使用的压缩级别。零意味着不压缩。对于自定义归档格式，这会指定个体表数据段的压缩，并且默认是进行中等级别的压缩。对于纯文本输出，设置一个非零压缩级别会导致整个输出文件被压缩，就好像它被gzip处理过一样，但是默认是不压缩。tar归档格式当前完全不支持压缩。

--binary-upgrade

这个选项用于就地升级功能。不推荐也不支持把它用于其他目的。

--column-inserts
--attribute-inserts

将数据转储为带有显式列名的INSERT命令（INSERT INTO *table* (*column*, ...) VALUES ...）。这将使得恢复过程非常慢，这主要用于使转储能够被载入到非UXDB数据库中。不过，由于这个选项为每一行都产生一个单独的命令，重载一行时的一个错误只会导致那一行被丢失而不是整个表内容丢失。

--disable-dollar-quoting

这个选项禁止在函数体中使用美元符号引用，并且强制它们使用SQL标准字符串语法被引用。

--disable-triggers

只有在创建一个只转储数据的转储时，这个选项才相关。它指示ux_dump包括在数据被重新载入时能够临时禁用目标表上的触发器的命令。如果你在表上有引用完整性检查或其他触发器，并且你在数据重新载入期间不想调用它们，请使用这个选项。

当前，必须作为超级用户来执行--disable-triggers命令。因此，你还应当使用-S指定一个超级用户名，或者最好是用一个超级用户的身份来启动生成脚本。

这个选项只对纯文本格式有意义。对于归档格式，你可以在调用ux_restore时指定这个选项。

--enable-row-security

只有在转储具有行安全性的表的内容时，这个选项才相关。默认情况下，ux_dump将把row-security设置为off来确保从该表中转储出所有的数据。如果用户不具有足够能绕过行安全性的特权，那么会抛出一个错误这个参数指示ux_dump将row-security设置为on，允许用户只转储该表中它们能够访问到的部分内容。

> **注意**
>
> 如果您当前使用此选项，则可能还需要以INSERT格式转储，因为还原期间的COPY FROM不支持行安全性。

--exclude-table-data=*pattern*

不转储匹配*pattern*表中的数据。该选项根据-t的相同规则被解释。--exclude-table-data可以被给定多次来排除匹配多个表。当你需要一个特定表的定义但不想要其中的数据时，可以使用这个选项。

--extra-float-digits=*ndigits*

在转储浮点数据时使用规定的*extra_float_digits*值，而不是最大可用精度。以备份目的生成的常规转储不使用此选项。

--if-exists

在清理数据库对象时使用条件命令(即添加IF EXISTS子句)。除非还指定了--clean，否则此选项无效。

--inserts

将数据转储为INSERT命令（而不是COPY）。这将使得恢复非常慢，这主要用于使转储能够被载入到非UXDB数据库中。不过，由于这个选项为每一行都产生一个单独的命令，重载一行时的一个错误只会导致那一行被丢失而不是整个表内容丢失。注意如果你已经重新安排了列序，该恢复可能会一起失败。--column-inserts选项对于列序改变是安全的，但是会更慢。

--load-via-partition-root

在为一个分区表转储数据时，让COPY语句或者INSERT语句把包含它的分区层次的根而不是分区自身作为目标。这导致在数据被装载时，会为每一个行重新确定合适的分区。如果在一台

服务器上重新装载数据时会出现行并不是总是落入到和原始服务器上相同的分区中的情况，这个选项就很有用。例如，如果分区列是文本类型并且两个系统中用于排序分区列的排序规则有着不同的定义，就会发生这种情况。

在从用这个选项制作的归档恢复时，最好不要使用并行，因为ux_restore将不能准确地知道一个给定的归档数据项将把数据装载到哪个分区中。这会导致效率不高，因为在并行任务见会有锁冲突，或者甚至可能由于在所有的相关数据被装载前建立了外键约束而导致重新装载失败。

--lock-wait-timeout=*timeout*

在转储的开始从不等待共享表锁的获得。而是在指定的*timeout*内不能锁定一个表时失败。超时时长可以用SET statement_timeout接受的任何格式指定（允许的格式根据你从其转出的服务器版本变化，并且所有版本都接受一个整数表示的毫秒数。）。

--no-comments

不转储注释。

--no-publications

不转储发布。

--no-security-labels

不转储安全标签。

--no-subscriptions

不转储订阅。

--no-sync

默认情况下，ux_dump将等待所有文件被安全地写入磁盘。这个选项会导致ux_dump无需等待而返回，但意味着后续的操作系统崩溃可能会导致转储损坏。通常，此选项对于测试非常有用，但在从生产安装中转储数据时不应使用此选项。

--no-synchronized-snapshots

在并行工作中不使用同步快照。

--no-tablespaces

不要输出创建表空间的命令，也不要为对象选择表空间。使用此选项，所有对象将在恢复期间的缺省表空间中创建。

这个选项只对纯文本格式有意义。对于归档格式，你可以在调用ux_restore时指定该选项。

--no-unlogged-table-data

不转储非日志记录表的内容。这个选项对于表定义（模式）是否被转储没有影响，它只会限制转储表数据。当从一个备用服务器转储时，在非日志记录表中的数据总是会被排除。

--on-conflict-do-nothing

增加ON CONFLICT DO NOTHING到INSERT命令。 除非规定了--inserts,--column-inserts或--rows-per-insert，否则此选项是无效的，

--quote-all-identifiers

强制引用所有标识符。当从UXDB主版本与ux_dump不同的服务器上转储一个数据库时或者当输出准备载入到一个具有不同主版本的服务器时，推荐使用这个选项。默认情况下，ux_dump只对在其主版本中是被保留词的标识符加上引号。在转储其他版本服务器时，这种默认行为有时会导致兼容性问题，因为那些版本可能具有些许不同的被保留词集合。使用--quote-all-identifiers能阻止这种问题，但代价是转储脚本更难阅读。

--rows-per-insert=nrows

数据转储为INSERT命令（而不是COPY）。 控制每个INSERT命令的最大行数。 指定的值必须大于零。重新加载期间的任何错误都将导致有问题的INSERT相关的行将丢失，而不是整个表内容。

--section=*sectionname*

只转储制定的部分。section名称可以是pre-data、data或post-data。可以多次指定此选项以选择多个部分。默认是转储所有节。

数据部分包含真正的表数据、大对象内容和序列值。pre-data包括所有其他数据定义项。post-data项包括索引、触发器、规则和除了已验证检查约束之外的约束的定义。

--serializable-deferrable

为转储使用一个可序列化事务，以保证所使用的快照与后来的数据库状态是一致的。但是这样做是在事务流中等待一个点，在该点上不能存在异常，这样就不会有转储失败或者导致其他事务serialization_failure回滚的风险。

此选项对于仅用于灾难恢复的转储没有好处。当原始数据库继续更新时，它对于用于加载数据库副本以进行报告或其他只读负载共享的转储很有用。没有它，转储可能反映一个与最终提交的事务的任何串行执行不一致的状态。例如，如果使用了批处理技术，一个批处理在转储中可以显示为关闭，而其中的所有项都不出现。

如果在启动ux_dump时没有活动的读写事务，则此选项将没有影响。如果有读写事务在活动，该转储的启动可能会被延迟一段不确定的时间。一旦开始运行，有没有这个开关的性能是相同的。

--snapshot=*snapshotname*

在做一个数据库的转储时指定一个同步的快照。

在需要把转储和一个逻辑复制槽或者一个并发会话同步时可以使用这个选项。

在并行转储的情况下，将使用这个选项指定的快照名而不是取一个新快照。

--strict-names

要求每一个模式（-n/--schema）和表（-t/--table）限定符匹配要转储的数据库中至少一个模式/表。注意，如果没有找到有这样的模式/表限定符匹配，即便没有--strict-names，ux_dump也将生成一个错误。

这个选项对-N/--exclude-schema、-T/--exclude-table或者--exclude-table-data没有效果。无法匹配任何对象的排除模式不会被当作错误。

--use-set-session-authorization

输出SQL标准的SET SESSION AUTHORIZATION命令取代ALTER OWNER命令来确定对象的所有关系。这让该转储更加兼容标准，但是根据该转储中对象的历史，该转储可能无法正常恢复。

而且，使用SET SESSION AUTHORIZATION的转储将一定会要求超级用户特权来恢复，而ALTER OWNER要求的特权较低。

-?
--help

显示有关ux_dump命令行参数的帮助并退出。

下列命令行选项控制数据库连接参数。

-d *dbname*
--dbname=*dbname*

指定要连接到的数据库名。这等效于指定*dbname*为命令行上的第一个非选项参数。

如果这个参数包含一个=符号或者以一个合法的URI前缀（UXDB://或uxdb://）开始，它将被视作一个*conninfo*字符串。

-h *host*
--host=*host*

指定服务器正在运行的机器的主机名。如果该值开始于一个斜线，它被用作一个 Unix 域套接字的目录。默认是从UXHOST环境变量中取得（如果被设置），否则将尝试一次 Unix 域套接字连接。

-p *port*
--port=*port*

指定服务器正在监听连接的 TCP 端口或本地 Unix 域套接字文件扩展名。默认是放在UXPORT环境变量中（如果被设置），否则使用编译在程序中的默认值。

-U *username*
--username=*username*

要作为哪个用户连接。

-w
--no-password

永远不要发出密码提示。如果服务器需要密码验证，而通过其他方式(如.pgpass文件)无法获得密码，则连接尝试将失败。此选项在没有用户可以输入密码的批处理作业和脚本中很有用。

-W
--password

在连接到数据库之前，强制ux_dump提示输入密码。

这个选项从来不是必须的，因为如果服务器要求输入密码，ux_dump将自动提示输入密码。但是，ux_dump将浪费一次连接尝试来发现服务器需要密码。在某些情况下，建议输入-W来避免额外的连接尝试。

用户输入的密码会在数据传输前以默认加密算法MD5进行加密，加密后与用户名（明文形式）一起传输到服务器端进行身份鉴别。如果用户想修改默认密码加密算法，只需要修改ux_hba.conf中的METHOD列认证方式即可。

--role=*rolename*

指定一个用来创建该转储的角色名。这个选项导致ux_dump在连接到数据库后发出一个SET ROLE *rolename*命令。当已认证用户（由-U指定）缺少ux_dump所需的特权但是能够切换到一个具有所需权利的角色时，这个选项很有用。一些安装有针对直接作为超级用户登录的策略，使用这个选项可以让转储在不违反该策略的前提下完成。

1.12.4. 环境变量

SKIP_WRAPPER

对备份导出的数据是否进行加密。默认值为0，当此环境变量为0时，会使用openssl命令及默认加密算法aes-256-cbc对备份数据进行加密；如果将此环境变量值设置为1，则不进行加密备份。

UXDATABASE
UXHOST
UXOPTIONS
UXPORT
UXUSER

默认连接参数。

UX_COLOR

规定在诊断消息中是否使用颜色。可能的值为always、 auto、never。

和大部分其他UXDB工具相似，这个工具也使用libuxsql支持的环境变量。

1.12.5. 诊断

ux_dump在内部执行SELECT语句。如果你运行ux_dump时出现问题，确定你能够从正在使用的数据库中选择信息，例如uxsql。此外，libuxsql前端-后端库所使用的任何默认连接设置和环境变量都将适用。

ux_dump的数据库活动会被统计收集器正常地收集。如果不想这样，你可以通过UXOPTIONS或ALTER USER命令设置参数track_counts为假。

1.12.6. 注解

如果你的数据库集群对于template1数据库有任何本地添加，要注意将ux_dump的输出恢复到一个真正的空数据库。否则你很可能由于以增加对象的重复定义而得到错误。要创建一个不带任何本地添加的空数据库，从template0而不是template1复制它，例如：

```
CREATE DATABASE foo WITH TEMPLATE template0;
```

当一个只含数据的转储被选中并且使用了选项--disable-triggers时，ux_dump在开始插入数据之前会发出命令禁用用户表上的触发器，并且接着在数据被插入之后发出命令重新启用它们。如果恢复中途被停止，系统目录可能会停留在一种错误状态。

ux_dump产生的转储文件不包含优化器用来做出查询计划决定的统计信息。因此，建议在从一个转储文件恢复后运行ANALYZE来确保最优性能。转储文件也不包含任何ALTER DATABASE ... SET命令，这些设置会与数据库用户及其他安装设置一起被ux_dumpall转储。

因为ux_dump被用来传输数据到更新版本的UXDB，ux_dump的输出被认为可以载入到比ux_dump版本更新的UXDB服务器中。ux_dump也能够从比其版本更旧的UXDB服务器中转储。不过，ux_dump无法从比起主版本号更新的UXDB服务器中转储，它甚至将拒绝冒着创建一个非法转储的风险尝试。还有，不保证ux_dump的输出能被载入到一个更旧主版本的服务器 — 即使该转储是从该版本的服务器中被取得也不行。将一个转储文件载入到一个更旧的服务器可能需要手工编辑该转储文件来移除旧服务器无法理解的语法。在跨版本的情况下，推荐使用--quote-all-identifiers选项，因为它可以避免因为不同UXDB版本间的保留词列表变化而发生问题。

在转储逻辑复制订阅时，ux_dump将生成使用connect = false选项的CREATE SUBSCRIPTION命令，这样恢复订阅时不会建立远程连接来创建复制槽或者进行初始的表拷贝。通过这种方式，可以无需到远程服务器的网络访问就能恢复该转储。然后就需要用户以一种合适的方式重新激活订阅。如果涉及到的主机已经改变，连接信息可能也必须被改变。在开启一次新的全表拷贝之前，截断目标表也可能是合适的。

1.12.7. 示例

要把一个数据库mydb转储到一个SQL脚本文件：

$ **ux_dump mydb > db.sql**

要把这样一个脚本重新载入到一个（新创建的）名为newdb的数据库中：

$ **uxql -d newdb -f db.sql**

要转储一个数据库到一个自定义格式归档文件：

$ **ux_dump -Fc mydb > db.dump**

要转储一个数据库到一个目录格式的归档：

$ **ux_dump -Fd mydb -f dumpdir**

要用5个并行的工作者任务转储一个数据库到一个目录格式的归档：

$ **ux_dump -Fd mydb -j 5 -f dumpdir**

要把一个归档文件重新载入到一个（新创建的）名为newdb的数据库：

$ **ux_restore -d newdb db.dump**

要转储一个名为mytab的表：

$ **ux_dump -t mytab mydb > db.sql**

要转储detroit模式中名称以emp开始的所有表，排除名为employee_log的表：

$ **ux_dump -t 'detroit.emp*' -T detroit.employee_log mydb > db.sql**

要转储名称以east或者west开始并且以gsm结束的所有模式，排除名称包含词test的任何模式：

$ ux_dump -n 'east*gsm' -n 'west*gsm' -N '*test*' mydb > db.sql

同样，用正则表达式记号法来合并开关：

$ ux_dump -n '(east|west)*gsm' -N '*test*' mydb > db.sql

要转储除了名称以ts_开头的表之外的所有数据库对象：

$ ux_dump -T 'ts_*' mydb > db.sql

要在-t和相关开关中指定一个大写形式或混合大小写形式的名称，你需要双引用该名称，否则它会被折叠到小写形式。但是双引号对于shell是特殊的，所以反过来它们必须被引用。因此，要转储一个有混合大小写名称的表，你需要类似这样的东西：

$ ux_dump -t "\"MixedCaseName\"" mydb > mytab.sql

1.13. ux_dumpall

ux_dumpall — 将一个UXDB数据库集群抽取到一个脚本文件中

1.13.1. 用法

ux_dumpall [*connection-option...*] [*option...*]

1.13.2. 描述

ux_dumpall工具可以一个集群中所有的UXDB数据库写出到（“转储”）一个脚本文件。该脚本文件包含可以用作uxsql的输入SQL命令来恢复数据库。它会对集群中的每个数据库调用ux_dump来完成该工作。ux_dumpall还转储对所有数据库公用的全局对象（ux_dump不保存这些对象）。 目前这包括数据库用户和组、表空间以及适合所有数据库的访问权限等属性。

因为ux_dumpall从所有数据库中读取表，所以你很可能需要以一个数据库超级用户的身份连接以便生成完整的转储。同样，你也需要超级用户特权执行保存下来的脚本，这样才能增加角色和组以及创建数据库。

SQL脚本将被写出到标准输出。使用[-f|file]选项或者shell操作符可以把它重定向到一个文件。

ux_dumpall需要多次连接到UXDB服务器（每个数据库一次）。如果你需要输入密码，可能每次都会要求输入密码。这种情况下使用一个~/.uxpass会比较方便。

1.13.3. 选项

下列命令行选项用于控制输出的内容和格式：

-a
--data-only

只转储数据，不转储模式（数据定义）。

-c
--clean

包括在重建数据库之前清除（移除）它们的SQL命令。角色和表空间的**DROP**命令也会被加入进来。

-E *encoding*
--encoding=*encoding*

用指定的字符集编码创建转储。默认情况下，转储使用数据库的编码创建（另一种得到相同结果的方法是设置UXCLIENTENCODING环境变量为想要的转储编码）。

-f *filename*
--file=*filename*

将输出发送到指定的文件中。如果省略，将使用标准输出。

-g
--globals-only

只转储全局对象（角色和表空间），而不转储数据库。

-o
--oids

将对象标识符（OID）转储为数据的一部分。如果你的应用以某种方式引用OID列（例如在外键约束中），请使用这个选项。否则不应该使用这个选项。

-O
--no-owner

不输出用于设置对象所有权以符合原始数据库的命令。默认情况下，ux_dumpall发出**ALTER OWNER**或**SET SESSION AUTHORIZATION**语句来设置被创建的模式元素的所有权。除非脚本是由一个超级用户（或者是拥有脚本中所有对象的同一个用户）所运行，这些语句在脚本运行时会失败。要使得一个脚本能被任意用户恢复，但又不想给予该用户所有对象的所有权，可以指定-O。

-r
--roles-only

只转储角色，不转储数据库和表空间。

--running-mode = standard | compatible | mysql

运行模式，默认模式是standard。standard表示标准模式，compatible表示兼容模式，mysql表示mysql模式。

-s
--schema-only

只转储对象定义（模式），不转储数据。

-S *username*
--superuser=*username*

指定要在禁用触发器时使用的超级用户的用户名。只有使用--disable-triggers时，这个选项才相关（通常，最好省去这个选项，而作为超级用户来启动结果脚本来取而代之）。

-t
--tablespaces-only

只转储表空间，不转储数据库和角色。

-v
--verbose

启用详细输出模式。这将导致ux_dumpall输出详细的对象注释以及转储文件的开始/停止时间，还有进度消息。它也会启用ux_dump中的细节输出。

-V
--version

打印ux_dumpall版本并退出。

-x
--no-privileges
--no-acl

防止转储访问特权（grant/revoke命令）。

--binary-upgrade

这个选项用于就地升级功能。不推荐也不支持把它用于其他目的。

--column-inserts
--attribute-inserts

将数据转储为带有显式列名的INSERT命令（INSERT INTO *table*(*column*, ...) VALUES ...）。这将使得恢复过程非常慢，这主要用于使转储能够被载入到非UXDB数据库中。

--disable-dollar-quoting

这个选项禁止在函数体中使用美元符号引用，并且强制它们使用SQL标准字符串语法被引用。

--disable-triggers

只有在创建一个只转储数据的转储时，这个选项才相关。它指示ux_dumpall在数据被重新载入时临时禁用目标表上的触发器的命令。如果你在表上有引用完整性检查或其他触发器，并且你在数据重新载入期间不想调用它们，可以使用这个选项。

当前，为--disable-triggers发出的命令必须作为超级用户来执行。因此，你还应当使用-S指定一个超级用户名，或者作为一个超级用户启动生成脚本。

--extra-float-digits=*ndigits*

在转储浮点数据时使用extra_float_digits规定的值，而不是最大可用精度。备份目的进行的常规转储不使用此选项。

--exclude-database=*pattern*

不要转储名字与*pattern*匹配的数据库。可以通过编写多个--exclude-database开关来排除多个模式。 pattern参数被解释为模式，根据uxsql的\d命令使用的相同规则，因此，通过在模式中编写通配符也可以排除多个数据库。 使用通配符时，请谨慎的引用模式，如果需要防止shell通配符扩展。

--if-exists

使用条件命令(例如，添加IF EXISTS子句)来删除数据库和其他对象。除非还指定了--clean，否则此选项无效。

--inserts

将数据转储为INSERT命令（而不是COPY）。这将使得恢复非常慢，这主要用于使转储能够被载入到非UXDB数据库中。注意如果你已经重新安排了列序，该恢复可能会一起失败。--column-inserts选项对于列序改变是安全的，但是会更慢。

--load-via-partition-root

在为一个分区表转储数据时，让COPY或INSERT语句以包含它的分区层次结构的根为目标，而不是分区本身。这将导致在加载数据时为每一行重新确定适当的分区。在恢复服务器上的数据时，这可能很有用，因为服务器上的行并不总是位于与原始服务器上相同的分区中。例如，如果分区列是文本类型并且两个系统中用于排序分区列的排序规则有着不同的定义，就会发生这种情况。

--lock-wait-timeout=*timeout*

不要永远等待在转储开始时获取共享表锁。相反，如果不能在指定的超时内锁定表，则失败。超时时长可以用SET statement_timeout接受的任何格式指定（允许的值取决于你从其转出的服务器版本）。

--no-comments

不转储注释。

--no-publications

不转储发布。

--no-role-passwords

不要为角色转储密码。恢复时，角色将具有空密码，并且在设置密码之前密码验证将始终失败。 由于指定此选项时不需要密码值，因此将从目录视图ux_roles中读取角色信息，而不是从ux_authid中读取角色信息。因此，如果访问ux_authid受到某些安全策略的限制，该选项也会有所帮助。

--no-security-labels

不转储安全标签。

--no-subscriptions

不转储订阅。

--no-sync

默认情况下，ux_dumpall将等待所有文件被安全地写入到磁盘。这个选项会导致ux_dumpall无需等待而返回，但意味着后续的操作系统崩溃可能会导致转储损坏。通常，此选项对于测试非常有用，但在从生产安装中转储数据时不应使用此选项。

--no-tablespaces

不要输出创建表空间的命令，也不要为对象选择表空间。使用此选项，所有对象将在恢复期间的缺省表空间中创建。

--no-unlogged-table-data

不转储非日志记录表的内容。这个选项对于表定义（模式）是否被转储没有影响，它只会限制转储表数据。

--on-conflict-do-nothing

添加ON CONFLICT DO NOTHING到INSERT命令。除非--inserts或--column-inserts也被指定，否则此选项不生效。

--quote-all-identifiers

强制引用所有标识符。当从UXDB主版本与ux_dumpall不同的服务器转储数据库时，或者当输出打算加载到不同主版本的服务器时，建议使用此选项。默认情况下，ux_dumpall只引用其主版本中的保留字标识符。在处理保留字集略有不同的其他版本的服务器时，这有时会导致兼容性问题。使用--quote-all-identifiers可以阻防止这类问题，但是代价是转储脚本会更加难以阅读。

--rows-per-insert=*nrows*

将数据转储为INSERT命令（而不是COPY）。控制每个INSERT命令的最大行数。 指定的值必须是大于零的数。重新加载期间的任何错误都将导致仅丢失有问题的INSERT的行，而不是整个表内容。

--use-set-session-authorization

输出SQL标准的SET SESSION AUTHORIZATION命令取代ALTER OWNER命令来确定对象的所有关系。这让该转储更加兼容标准，但是根据该转储中对象的历史，该转储可能无法正常恢复。

-?
--help

显示有关ux_dumpall命令行参数的帮助并退出。

下列命令行选项控制数据库连接参数。

-d *connstr*
--dbname=*connstr*

指定用于连接到服务器的参数。

这个选项被称为--dbname是为了和其他客户端应用一致，但是因为ux_dumpall需要连接多个数据库，连接字符串中的数据库名将被忽略。使用-l选项指定一个数据库，该数据库被用来转储全局对象并且发现需要转储哪些其他数据库。

-h *host*
--host=*host*

指定服务器正在运行的机器的主机名。如果该值开始于一个斜线，它被用作一个 Unix 域套接字的目录。默认是从UXHOST环境变量中取得（如果被设置），否则将尝试一次 Unix 域套接字连接。

-l *dbname*
--database=*dbname*

指定要连接到的数据库的名称，以转储全局对象并发现应该转储的其他数据库。如果没有指定，将会使用uxdb数据库，如果uxdb不存在，就使用 template1。

-p *port*
--port=*port*

指定服务器正在监听连接的 TCP 端口或本地 Unix 域套接字文件扩展名。默认是放在UXPORT环境变量中（如果被设置），否则使用编译在程序中的默认值。

-U *username*
--username=*username*

要作为哪个用户连接。

-w
--no-password

永远不要发出密码提示。如果服务器需要密码验证，而通过其他方式(如.pgpass文件)无法获得密码，则连接尝试将失败。此选项在没有用户可以输入密码的批处理作业和脚本中很有用。

-W
--password

在连接到数据库之前，强制ux_dumpall提示输入密码。

这个选项从来不是必须的，因为如果服务器要求输入密码，ux_dumpall将自动提示输入密码。但是，ux_dumpall将浪费一次连接尝试来发现服务器想要一个密码。在某些情况下，建议输入-W来避免额外的连接尝试。

注意对每个要被转储的数据库，密码提示都会再次出现。通常，最好设置一个~/.uxpass文件来减少手动密码输入。

--role=*rolename*

指定一个用来创建该转储的角色名。这个选项导致ux_dump在连接到数据库后发出一个**SET ROLE** *rolename*命令。当已认证用户（由-U指定）缺少ux_dump所需的特权但是能够切换到一个具有所需权利的角色时，这个选项很有用。一些安装有针对直接作为超级用户登录的策略，使用这个选项可以让转储在不违反该策略的前提下完成。

1.13.4. 环境变量

UXHOST
UXOPTIONS
UXPORT
UXUSER

默认连接参数。

UX_COLOR

规定在诊断消息中是否使用颜色。可能的值为always、auto、never。

和大部分其他UXDB工具相似，这个工具也使用libuxsql支持的环境变量。

1.13.5. 注解

因为ux_dumpall在内部调用ux_dump，所以，一些诊断消息可以参考ux_dump。

即使当用户的目的是把转储脚本恢复到一个空的集簇中，--clean选项也有用武之地。--clean的使用让该脚本删除并且重建内建的uxdb和template1数据库，确保这两个数据库保持与源集簇中相同的属性（例如locale和编码）。如果不用这个选项，这两个数据库将保持它们现有的数据库级属性以及任何已有的内容。

一旦恢复，建议在每个数据库上运行ANALYZE，这样优化器就可以得到有用的统计信息。你也可以运行vacuumdb -a -z来分析所有数据库。

不应该预期转储脚本运行到结束都不出错。特别是由于脚本将为源集簇中已有的每一个角色发出CREATE ROLE语句，对于超级用户当然会得到一个“role already exists”错误，除非目标集簇用一个不同的超级用户名完成的初始化。这种错误是无用的并且应该被忽略。--clean选项的使用很可能会产生额外的有关于不存在对象的不致命错误消息，不过可以通过加上--if-exists减少这类错误消息。

ux_dumpall要求所有需要的表空间目录在进行恢复之前就必须存在；否则，数据库创建就会由于在非默认位置创建数据库而失败。

1.13.6. 示例

要转储所有数据库：

```
$ ux_dumpall > db.out
```

要从这个文件重新载入数据库，你可以使用：

```
$ uxsql -f db.out uxdb
```

这里你连接哪一个数据库并不重要，因为由ux_dumpall创建的脚本将包含合适的命令来创建和连接到被保存的数据库。

1.14. ux_isready

ux_isready — 检查一个UXDB服务器的连接状态

1.14.1. 用法

ux_isready [*connection-option*...] [*option*...]

1.14.2. 描述

ux_isready是一个用来检查一个UXDB数据库服务器的连接状态的工具。其退出状态指定了连接检查的结果。

1.14.3. 选项

-d *dbname*
--dbname=*dbname*

指定要连接的数据库名。

如果这个参数包含一个=记号或者以一个合法的URI前缀（UXDB://或uxdb://）开始，它会被当作一个*conninfo*字符串。

-h *hostname*
--host=*hostname*

指定运行服务器的机器的主机名。如果该值以一个斜线开始，它被用作 Unix 域套接字的目录。

-p *port*
--port=*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。默认值取自UXPORT环境变量。如果环境变量没有设置，则默认值使用编译时指定的端口（通常是5432）。

-q
--quiet

不显示状态消息。当脚本编程时有用。

-t *seconds*
--timeout=*seconds*

尝试连接时，在返回服务器不响应之前等待的最大秒数。设置为0则禁用。默认值是3秒。

-U *username*
--username=*username*

作为用户*username*连接数据库，而不是用默认用户。

-V
--version

打印ux_isready版本并退出。

-?
--help

显示有关ux_isready命令行参数的帮助并退出。

1.14.4. 退出状态

如果服务器正常接受连接，ux_isready返回0给shell；如果服务器拒绝连接（例如处于启动阶段）则返回1；如果连接尝试没有被相应则返回2；如果没有尝试（例如由于非法参数）则返回3。

1.14.5. 环境变量

UX_COLOR

规定在诊断消息中是否使用颜色。可能的值为always、auto、never。

和大部分其他UXDB工具相似，ux_isready也使用libuxsql支持的环境变量。

1.14.6. 注解

要获得服务器状态，不一定需要提供正确的用户名、密码或数据库名。不过，如果提供了不正确的值，服务器将会记录一次失败的连接尝试。

1.14.7. 示例

标准用法：

```
$ ux_isready
/tmp:5432 - accepting connections
$ echo $?
0
```

使用连接参数运行连接到处于启动中的UXDB集群：

```
$ ux_isready -h localhost -p 5433
localhost:5433 - rejecting connections
$ echo $?
1
```

使用连接参数运行连接到无响应的UXDB集群：

```
$ ux_isready -h someremotehost
someremotehost:5432 - no response
$ echo $?
2
```

1.15. ux_probackup

ux_probackup — 用于管理UxsinoDB数据库集群的备份和恢复的实用程序

1.15.1. 用法

ux_probackup [*option*...]

- 查看ux_probackup命令的详细信息。

```
[uxdb@local65 bin]$./ux_probackup  --help
  ux_probackup - utility to manage backup/recovery of UXsinoDB database.

  ux_probackup help [COMMAND]

  ux_probackup version

  ux_probackup init -B backup-path
```

```
ux_probackup set-config -B backup-path --instance=instance_name
            [-D uxdata-path]
            [--external-dirs=external-directories-paths]
            [--log-level-console=log-level-console]
            [--log-level-file=log-level-file]
            [--log-filename=log-filename]
            [--error-log-filename=error-log-filename]
            [--log-directory=log-directory]
            [--log-rotation-size=log-rotation-size]
            [--log-rotation-age=log-rotation-age]
            [--retention-redundancy=retention-redundancy]
            [--retention-window=retention-window]
            [--wal-depth=wal-depth]
            [--compress-algorithm=compress-algorithm]
            [--compress-level=compress-level]
            [--archive-timeout=timeout]
            [-d dbname] [-h host] [-p port] [-U username]
            [--remote-proto] [--remote-host]
            [--remote-port] [--remote-path] [--remote-user]
            [--ssh-options]
            [--restore-command=cmdline] [--archive-host=destination]
            [--archive-port=port] [--archive-user=username]
            [--help]

ux_probackup set-backup -B backup-path --instance=instance_name
            -i backup-id [--ttl=interval] [--expire-time=timestamp]
            [--note=text]
            [--help]

ux_probackup show-config -B backup-path --instance=instance_name
            [--format=format]
            [--help]
ux_probackup delete -B backup-path --instance=instance_name
            [-j num-threads] [--progress]
            [--retention-redundancy=retention-redundancy]
            [--retention-window=retention-window]
            [--wal-depth=wal-depth]
            [-i backup-id | --delete-expired | --merge-expired | --status=backup_status]
            [--delete-wal]
            [--dry-run]
            [--help]

ux_probackup add-instance -B backup-path -D uxdata-path
            --instance=instance_name
            [--external-dirs=external-directories-paths]
            [--remote-proto] [--remote-host]
            [--remote-port] [--remote-path] [--remote-user]
            [--ssh-options]
            [--help]
      .
          .
          .
          .
          .
```

- 查看具体的参数的描述信息。

  以添加备份实例add-instance命令的详细信息为例。

```
[uxdb@local65 bin]$ ux_probackup add-instance --help

ux_probackup add-instance -B backup-path -D uxdata-path
              --instance=instance_name
              [-E external-directory-path]
              [--remote-proto] [--remote-host]
              [--remote-port] [--remote-path] [--remote-user]
              [--ssh-options]

 -B, --backup-path=backup-path    location of the backup storage area
 -D, --uxdata=uxdata-path         location of the database storage area
    --instance=instance_name     name of the new instance
 -E  --external-dirs=external-directories-paths
                           backup some directories not from uxdata
                           (example: --external-dirs=/tmp/dir1:/tmp/dir2)

 Remote options:
    --remote-proto=protocol      remote protocol to use
                           available options: 'ssh', 'none' (default: ssh)
    --remote-host=destination    remote host address or hostname
    --remote-port=port           remote host port (default: 22)
    --remote-path=path           path to directory with ux_probackup binary on remote host
                           (default: current binary path)
    --remote-user=username       user name for ssh connection (default: current user)
    --ssh-options=ssh_options    additional ssh options (default: none)
                           (example: --ssh-options='-c cipher_spec -F configfile')
```

1.15.2. 描述

ux_probackup旨在执行UxsinoDB实例的定期备份，能够在发生故障时还原服务器。

与其他备份解决方案相比，ux_probackup具有以下优点，可实施不同的备份策略并处理大量数据：

- 增量备份：页面级增量备份可以节省磁盘空间，加快备份和还原速度。使用三种不同的增量模式，可以根据数据流计划备份策略。
- 增量还原：页面级增量还原允许通过在目标目录中重用有效的未更改页面来极大地加快还原速度。
- 合并：使用此功能，可以实施“增量更新的备份”策略，而无需进行定期的完整备份。
- 验证：自动数据一致性检查和按需备份验证，无需实际数据恢复。
- 验证：使用checkdb命令按需验证UXsinoDB实例。
- 保留：根据保留策略管理WAL存档和备份。可以根据恢复时间或要保留的备份数量来配置保留策略，也可以time to live为特定备份指定（TTL）。过期的备份可以合并或删除。
- 并行化：在多个并行线程上运行备份，还原，合并，删除，验证和验证过程。

- 压缩：以压缩状态存储备份数据以节省磁盘空间。
- 重复数据删除：通过不复制未更改的非数据文件（例如_vm）来节省磁盘空间_fsm。
- 远程操作：备份位于远程系统上的UXsinoDB实例或远程还原备份，
- 从备用服务器进行备份：通过从备用服务器进行备份来避免主服务器上的额外负载。
- 外部目录：备份位于UxsinoDB data directory（UXDATA）外部的文件和目录，例如脚本，配置文件，日志或SQL转储文件。
- 备份目录：以纯文本或JSON格式获取备份列表和相应的元信息。
- 存档目录：以纯文本或JSON格式获取所有WAL时间轴的列表以及相应的元信息。
- 部分还原：仅还原指定的数据库或从还原中排除指定的数据库。

要管理备份数据，ux_probackup创建一个备份目录。该目录存储所有带有附加元信息的备份文件，以及时间点恢复所需的WAL存档。可以将不同实例的备份存储在单个备份目录的单独子目录中。

使用ux_probackup，可以进行完整或增量备份：

- Full备份包含从头还原数据库集群所需的所有数据文件。
- Incremental备份仅存储自上次备份以来已更改的数据，它允许减小备份大小并加快备份操作。支持以下增量备份模式：
  - PAGE备份。在此模式下，ux_probackup从上一次进行完整备份或增量备份开始扫描存档中的所有WAL文件。新创建的备份仅包含WAL记录中提到的页面。这要求自上次备份以来，所有WAL文件都存在于WAL存档中。如果这些文件的大小可与数据库集群文件的总大小相媲美，则加速会更小，但备份仍会占用较少的空间。
  - DELTA备份。在这种模式下，ux_probackup读取UXDATA目录中的所有数据文件，仅复制那些自上次备份以来已更改的页面。连续归档对于它的运行不是必需的。同样，此模式可能会施加等于Full备份的只读I/O压力。
  - PTRACK备份。在这种模式下，UxsinoDB会实时跟踪页面更改。连续归档对于它的运行不是必需的。每次更新关系页面时，都会在PTRACK此关系的特殊位图中标记该页面。由于一页仅需要PTRACK分叉中的一位，因此此类位图非常小。跟踪意味着数据库服务器操作会花费一些开销，但是会大大加快增量备份的速度。

无论选择哪种备份类型，所有使用ux_probackupWAL交付的备份都支持以下WAL交付策略：

- Autonomous backups通过复制协议传输所有WAL文件，以便在进行备份时将集群还原到一致状态。即使未设置连续归档，所需的WAL段也包含在备份中。
- Archive backups 依靠连续归档。

ux_probackup当前具有以下限制：

- 从中进行备份的服务器和还原的服务器必须与block_size和wal_block_size参数兼容，并且具有相同的主要发行版号。
- 目前不支持在Windows上通过ssh进行远程备份。
- 通过ssh运行远程操作时，远程和本地ux_probackup版本必须相同。

1.15.3. 选项

--instance

指定instance的名称。

--remote-host

指定远程备份实例IP。

--remote-port

指定远程备份实例SSH端口（默认值：22）。

--remote-user

指定远程SSH用户。

--remote-path

指定远程备份实例ux_probackup工具所在路径。

-B

备份路径位置。

-D

uxdata数据库实例的路径。

-b

指定备份模式（backup mode=FULL|PAGE|DELTA|PTRACK）。

-j

并行的线程数。

--archive-host

到存档主机的ssh连接的目标地址或主机名。

--archive-port

ssh连接到存档主机的端口端口（默认值：22）。

--archive-user

到存档主机的ssh连接的用户名（默认值：UXsinoDB user）。

--running-mode = standard | compatible | mysql

运行模式，默认模式是standard。standard表示标准模式，compatible表示兼容模式，mysql表示mysql模式。

1.15.4. 示例

1.15.4.1. 初始化备份目录

./ux_probackup init -B /home/uxdb/data/probackup

```
[uxdb@local65 bin]$  ux_probackup init -B /home/uxdb/data/probackup
INFO: Backup catalog '/home/uxdb/data/probackup' successfully inited
[uxdb@local65 bin]$
```

查看备份目录。

```
[uxdb@local65 probackup]$ pwd
/home/uxdb/data/probackup
[uxdb@local65 probackup]$ ls
backups  wal
[uxdb@local65 probackup]$
```

1.15.4.2. 添加需要备份实例信息

一个备份目录下可添加多个实例，可以是本地实例也可以是远程实例。

1. 初始化数据库实例。

   [uxdb@local65 bin]$ ./initdb -W -D test

2. 添加备份实例。

   [uxdb@local65 bin]$ ux_probackup add-instance -B /home/uxdb/data/probackup -D /home/uxdb/uxdbinstall/dbsql/bin/test/ --instance=test1

   命令执行成功之后，在备份路径下的backups目录下会有test1目录，该目录下是备份实例的备份配置文件，记录备份实例的相关信息。

```
[uxdb@local65 probackup]$ pwd
/home/uxdb/data/probackup
[uxdb@local65 probackup]$ ls
backups  wal
[uxdb@local65 probackup]$ cd backups/
[uxdb@local65 backups]$ ls
test1
[uxdb@local65 backups]$ cd test1/
[uxdb@local65 test1]$ ls
ux_probackup.conf
```

3. 修改本地备份实例的配置文件uxsinodb.conf。

   max_wal_senders = 10 #根据实际场景设置合理值，此处使用默认值10
   wal_level = 'replica'
   archive_mode = 'on'
   archive_command = '/home/uxdb/uxdbinstall/dbsql/bin/ux_probackup archive-push -B /home/uxdb/data/probackup --instance test1 --wal-file-path=%p --wal-file-name=%f '

4. 启动数据库并创建测试表。

   ./ux_ctl -D test/ start
   ./uxsql

```
uxdb=# create table t1(id int);
CREATE TABLE
uxdb=# insert into t1 values (1);
INSERT 0 1
uxdb=# insert into t1 values (2);
INSERT 0 1
uxdb=# insert into t1 values (3);
INSERT 0 1
uxdb=# insert into t1 values (4);
INSERT 0 1
uxdb=# insert into t1 values (5);
INSERT 0 1
uxdb=# insert into t1 values (6);
INSERT 0 1
uxdb=# insert into t1 values (7);
INSERT 0 1
uxdb=# insert into t1 values (8);
INSERT 0 1
uxdb=# select count(*) from t1;
 count
-------
     8
(1 row)

uxdb=#
```

1.15.4.3. 显示备份实例的配置信息

[uxdb@local65 bin]$ ./ux_probackup show-config -B /home/uxdb/data/probackup --instance test1

```
[uxdb@local65 bin]$ ./ux_probackup show-config -B /home/uxdb/data/probackup --instance test1
# Backup instance information
uxdata = /home/uxdb/uxdbinstall/dbsql/bin/test
system-identifier = 6886013058702885863
xlog-seg-size = 16777216
# Connection parameters
uxdatabase = uxdb
# Replica parameters
replica-timeout = 5min
# Archive parameters
archive-timeout = 5min
# Logging parameters
log-level-console = INFO
log-level-file = OFF
log-filename = ux_probackup.log
log-rotation-size = 0TB
log-rotation-age = 0d
# Retention parameters
retention-redundancy = 0
retention-window = 0
wal-depth = 0
# Compression parameters
compress-algorithm = none
compress-level = 1
# Remote access parameters
remote-proto = ssh
[uxdb@local65 bin]$
```

1.15.4.4. 全量备份

[uxdb@local65 bin]$ ./ux_probackup backup -B /home/uxdb/data/probackup --instance=test1 -b full

```
[uxdb@local65 bin]$ ./ux_probackup backup -B /home/uxdb/data/probackup --instance=test1 -b full
INFO: Backup start, ux_probackup version: 2.4.4, instance: test1, backup ID: QILA2N, backup mode: FULL, wal mode: ARCHIVE, remote: false, compress-algorithm: none, c
ompress-level: 1
Password:
WARNING: This UXsinoDB instance was initialized without data block checksums. ux_probackup have no way to detect data block corruption without them. Reinitialize UXD
ATA with option '--data-checksums'.
WARNING: Current UXsinoDB role is superuser. It is not recommended to run backup or checkdb as superuser.
INFO: Wait for WAL segment /home/uxdb/data/probackup/wal/test1/000000010000000000000003 to be archived
INFO: UXDATA size: 51MB
INFO: Start transferring data files
INFO: Data files are transferred, time elapsed: 3s
INFO: wait for ux_stop_backup()
INFO: ux_stop backup() successfully executed
INFO: Syncing backup files to disk
INFO: Backup files are synced, time elapsed: 0
INFO: Validating backup QILA2N
INFO: Backup QILA2N data files are valid
INFO: Backup QILA2N resident size: 51MB
INFO: Backup QILA2N completed
[uxdb@local65 bin]$
```

如果不想多次手动输入密码，可以通过.uxpass进行免密操作，查看备份路径下的内容、

```
[uxdb@local65 test1]$ pwd
/home/uxdb/data/probackup/backups/test1
[uxdb@local65 test1]$ ls
QILA2N  ux_probackup.conf
[uxdb@local65 test1]$ cd QILA2N/
[uxdb@local65 QILA2N]$ ls
backup_content.control  backup.control  database  page_header_map
[uxdb@local65 QILA2N]$ cd database/
[uxdb@local65 database]$ ls
backup_label      database_map  ux_commit_ts  ux_ident.conf  ux_notify    uxsinodb.auto.conf  ux_stat      ux_tblspc    ux_wal
base              global        ux_dynshmem   ux_logical     ux_replslot  uxsinodb.conf       ux_stat_tmp  ux_twophase  ux_xact
current_logfiles  log           ux_hba.conf   ux_multixact   ux_serial    ux_snapshots        ux_subtrans  UX_VERSION
[uxdb@local65 database]$
```

1.15.4.5. 增量备份

增量备份有3种模式：PAGE、DELTA和PTRACK。

- backup mode=PAGE

  [uxdb@local65 bin]$ ./ux_probackup backup -B /home/uxdb/data/probackup --instance=test1 -b page

```
[uxdb@local65 bin]$ ./ux_probackup backup -B /home/uxdb/data/probackup --instance=test1 -b page
INFO: Backup start, ux_probackup version: 2.4.4, instance: test1, backup ID: QILAD0, backup mode: PAGE, wal mode: ARCHIVE, remote: false, compress-algorithm: none, c
ompress-level: 1
Password:
WARNING: This UXsinoDB instance was initialized without data block checksums. ux_probackup have no way to detect data block corruption without them. Reinitialize UXD
ATA with option '--data-checksums'.
WARNING: Current UXsinoDB role is superuser. It is not recommended to run backup or checkdb as superuser.
INFO: Wait for WAL segment /home/uxdb/data/probackup/wal/test1/000000010000000000000006 to be archived
INFO: Parent backup: QILA2N
INFO: UXDATA size: 51MB
INFO: Extracting pagemap of changed blocks
INFO: Pagemap successfully extracted, time elapsed: 0 sec
INFO: Start transferring data files
INFO: Data files are transferred, time elapsed: 0
INFO: wait for ux_stop_backup()
INFO: ux_stop backup() successfully executed
INFO: Wait for LSN 0/70000F0 in archived WAL segment /home/uxdb/data/probackup/wal/test1/000000010000000000000007
INFO: Syncing backup files to disk
INFO: Backup files are synced, time elapsed: 0
INFO: Validating backup QILAD0
INFO: Backup QILAD0 data files are valid
INFO: Backup QILAD0 resident size: 172kB
INFO: Backup QILAD0 completed
[uxdb@local65 bin]$
```

查看备份内容：

```
[uxdb@local65 test1]$ ls
QILA2N  QILAD0  ux_probackup.conf
[uxdb@local65 test1]$ cd QILAD0/
[uxdb@local65 QILAD0]$ ls
backup_content.control  backup.control  database  page_header_map
[uxdb@local65 QILAD0]$ cd database/
[uxdb@local65 database]$ ls
backup_label  database_map  log          ux_dynshmem  ux_multixact  ux_replslot  ux_snapshots  ux_stat_tmp  ux_tblspc    ux_wal
base          global        ux_commit_ts  ux_logical   ux_notify     ux_serial    ux_stat       ux_subtrans  ux_twophase  ux_xact
[uxdb@local65 database]$
```

- backup mode=DELTA

  [uxdb@local65 bin]$ ./ux_probackup backup -B /home/uxdb/data/probackup --instance=test1 -b delta

```
[uxdb@local65 bin]$ ./ux_probackup backup -B /home/uxdb/data/probackup --instance=test1 -b DELTA
INFO: Backup start, ux_probackup version: 2.4.4, instance: test1, backup ID: QILAJ1, backup mode: DELTA, wal mode: ARCHIVE, remote: false, compress-algorithm: none,
compress-level: 1
Password:
WARNING: This UXsinoDB instance was initialized without data block checksums. ux_probackup have no way to detect data block corruption without them. Reinitialize UXD
ATA with option '--data-checksums'.
WARNING: Current UXsinoDB role is superuser. It is not recommended to run backup or checkdb as superuser.
INFO: Wait for WAL segment /home/uxdb/data/probackup/wal/test1/000000010000000000000009 to be archived
INFO: Parent backup: QILAD0
INFO: UXDATA size: 51MB
INFO: Start transferring data files
INFO: Data files are transferred, time elapsed: 0
INFO: wait for ux_stop_backup()
INFO: ux_stop backup() successfully executed
INFO: Syncing backup files to disk
INFO: Backup files are synced, time elapsed: 0
INFO: Validating backup QILAJ1
INFO: Backup QILAJ1 data files are valid
INFO: Backup QILAJ1 resident size: 140kB
INFO: Backup QILAJ1 completed
[uxdb@local65 bin]$
```

查看备份内容：

```
[uxdb@local65 QILA2N]$ cd ../QILAJ1/
[uxdb@local65 QILAJ1]$ ls
backup_content.control  backup.control  database
[uxdb@local65 QILAJ1]$ cd database/
[uxdb@local65 database]$ ls
backup_label  database_map  log          ux_dynshmem  ux_multixact  ux_replslot  ux_snapshots  ux_stat_tmp  ux_tblspc    ux_wal
base          global        ux_commit_ts  ux_logical   ux_notify     ux_serial    ux_stat       ux_subtrans  ux_twophase  ux_xact
[uxdb@local65 database]$
```

- backup mode=PTRACK

  备份模式为PTRACK时需要在初始化数据库实例时有数据块校验和，因此需要重新初始化数据库实例。

  1. 初始化数据库。

     ./initdb -k -W -D data

  2. 添加需要备份实例的信息。

     ux_probackup add-instance -B /home/uxdb/data/probackup -D /home/uxdb/uxdbinstall/dbsql/bin/data/ --instance=data1

  3. 修改实例data的uxsinodb.conf文件。

max_wal_senders = 10 #根据实际场景设置合理值，此处使用默认值10
wal_level = 'replica'
archive_mode = 'on'
archive_command = '/home/uxdb/uxdbinstall/dbsql/bin/ux_probackup archive-push -B /home/uxdb/data/probackup --instance data1 --wal-file-path=%p --wal-file-name=%f '

4. 启动数据库创建测试表。

```
uxdb=# create table t1(id int);
CREATE TABLE
uxdb=# insert into t1 values (11)
uxdb-# ;
INSERT 0 1
uxdb=# insert into t1 values (12);
INSERT 0 1
uxdb=# insert into t1 values (13);
INSERT 0 1
uxdb=# insert into t1 values (14);
INSERT 0 1
uxdb=# insert into t1 values (15);
INSERT 0 1
uxdb=# insert into t1 values (16);
INSERT 0 1
uxdb=# insert into t1 values (17);
INSERT 0 1
uxdb=# select count(*) from t1;
 count
-------
     7
(1 row)

uxdb=#
```

5. 先对data实例进行一次全备操作。

[uxdb@local65 bin]$ ./ux_probackup backup -B /home/uxdb/data/probackup --instance=data1 -b full

```
[uxdb@local65 bin]$ ./ux_probackup backup -B /home/uxdb/data/probackup --instance=data1 -b full
INFO: Backup start, ux_probackup version: 2.4.4, instance: data1, backup ID: QIMTZR, backup mode: FULL, wal mode: ARCHIVE, remote: false, compress-algorithm: none, c
ompress-level: 1
Password:
WARNING: Current UXsinoDB role is superuser. It is not recommended to run backup or checkdb as superuser.
INFO: Wait for WAL segment /home/uxdb/data/probackup/wal/data1/000000010000000000000002 to be archived
INFO: UXDATA size: 51MB
INFO: Start transferring data files
INFO: Data files are transferred, time elapsed: 1s
INFO: wait for ux_stop_backup()
INFO: ux_stop backup() successfully executed
INFO: Syncing backup files to disk
INFO: Backup files are synced, time elapsed: 0
INFO: Validating backup QIMTZR
INFO: Backup QIMTZR data files are valid
INFO: Backup QIMTZR resident size: 51MB
INFO: Backup QIMTZR completed
[uxdb@local65 bin]$
```

6. 数据库测试表t1中再次插入3条数据。

```
uxdb=# insert into t1 values (19);
INSERT 0 1
uxdb=# insert into t1 values (20);
INSERT 0 1
uxdb=# insert into t1 values (21);
INSERT 0 1
uxdb=# select count(*) from t1;
 count
-------
    10
(1 row)

uxdb=#
```

7. 以ptrack模式进行增量备份。

   [uxdb@local65 bin]$ ./ux_probackup backup -B /home/uxdb/data/probackup --instance=data1 -b ptrack

```
[uxdb@local65 bin]$ ./ux_probackup backup -B /home/uxdb/data/probackup --instance=data1 -b ptrack
INFO: Backup start, ux_probackup version: 2.4.4, instance: data1, backup ID: QIMU7Z, backup mode: PTRACK, wal mode: ARCHIVE, remote: false, compress-algorithm: none,
 compress-level: 1
Password:
WARNING: Current UXsinoDB role is superuser. It is not recommended to run backup or checkdb as superuser.
ERROR: This UXsinoDB instance does not support ptrack
WARNING: Backup QIMU7Z is running, setting its status to ERROR
```

   该问题的原因是使用ptrack模式备份时依赖ptrack这个插件，暂不提供。

```
[postgres@local63 bin]$ ./pg_probackup backup -B /home/postgres/probackup --instance=data1 -b PTRACK
INFO: Backup start, pg_probackup version: 2.4.4, instance: data1, backup ID: QIMWYX, backup mode: PTRACK, wal mode: ARCHIVE, remote: false, compress-algorithm: none,
 compress-level: 1
WARNING: Current PostgreSQL role is superuser. It is not recommended to run backup or checkdb as superuser.
ERROR: This PostgreSQL instance does not support ptrack
WARNING: Backup QIMWYX is running, setting its status to ERROR
[postgres@local63 bin]$ ./pg_probackup backup --help
```

1.15.4.6. 查看备份信息

[uxdb@local65 bin]$ ./ux_probackup show -B /home/uxdb/data/probackup

```
[uxdb@local65 bin]$ ./ux_probackup show -B /home/uxdb/data/probackup

BACKUP INSTANCE 'test1'
================================================================================================================================================
 Instance  Version  ID      Recovery Time           Mode   WAL Mode  TLI  Time   Data   WAL  Zratio  Start LSN  Stop LSN   Status
================================================================================================================================================
 test1     12       QILAJ1  2020-10-22 14:11:33+08  DELTA  ARCHIVE   1/1    9s  140kB  16MB    1.00  0/9000028  0/A0000B8  OK
 test1     12       QILAD0  2020-10-22 14:07:56+08  PAGE   ARCHIVE   1/1   10s  172kB  16MB    1.00  0/6000028  0/70000F0  OK
 test1     12       QILA2N  2020-10-22 14:01:47+08  FULL   ARCHIVE   1/0   13s   51MB  16MB    1.00  0/3000028  0/40000F0  OK
[uxdb@local65 bin]$
```

1.15.4.7. 验证备份

[uxdb@local65 bin]$ ./ux_probackup validate -B /home/uxdb/data/probackup --instance=test1

```
[uxdb@local65 bin]$ ./ux_probackup validate -B /home/uxdb/data/probackup --instance=test
INFO: Validate backups of the instance 'test1'
WARNING: Backup QILCGE has missing parent 0
WARNING: Backup QILBL8 has missing parent 0
INFO: Validating backup QILAJ1
INFO: Backup QILAJ1 data files are valid
INFO: Backup QILAJ1 WAL segments are valid
INFO: Validating backup QILAD0
INFO: Backup QILAD0 data files are valid
INFO: Backup QILAD0 WAL segments are valid
INFO: Validating backup QILA2N
INFO: Backup QILA2N data files are valid
INFO: Backup QILA2N WAL segments are valid
WARNING: Some backups are not valid
[uxdb@local65 bin]$
```

1.15.4.8. 并行备份

[uxdb@local65 bin]$./ux_probackup backup -B /home/uxdb/data/probackup --instance=data1 -b full -j 4

```
[uxdb@local65 bin]$ ./ux_probackup backup -B /home/uxdb/data/probackup --instance=data1 -b full -j 4
INFO: Backup start, ux_probackup version: 2.4.4, instance: data1, backup ID: QINE9H, backup mode: FULL, wal mode: ARCHIVE, remote: false, compress-algorithm: none, c
ompress-level: 1
Password:
WARNING: Current UXsinoDB role is superuser. It is not recommended to run backup or checkdb as superuser.
INFO: Wait for WAL segment /home/uxdb/data/probackup/wal/data1/000000010000000000000005 to be archived
INFO: UXDATA size: 51MB
INFO: Start transferring data files
INFO: Data files are transferred, time elapsed: 4s
INFO: wait for ux_stop_backup()
INFO: ux_stop backup() successfully executed
INFO: Wait for LSN 0/60000B8 in archived WAL segment /home/uxdb/data/probackup/wal/data1/000000010000000000000006
INFO: Syncing backup files to disk
INFO: Backup files are synced, time elapsed: 0
INFO: Validating backup QINE9H
INFO: Backup QINE9H data files are valid
INFO: Backup QINE9H resident size: 51MB
INFO: Backup QINE9H completed
[uxdb@local65 bin]$
```

1.15.4.9. 恢复

恢复时数据库实例目录必须要为空，即恢复新创建的实例目录下，或者需要将源实例目录重命名。

1. 根据备份集恢复到一个空的实例目录下。

   a. 在uxdbinstall/dbsql/bin目录下创建restore目录，恢复到该目录下。

   [uxdb@local65 bin]$ ./ux_probackup restore -B /home/uxdb/data/probackup --instance=test1 -D /home/uxdb/uxdbinstall/dbsql/bin/restore -j 4

```
[uxdb@local65 bin]$ ./ux_probackup restore -B /home/uxdb/data/probackup --instance=test1 -D /home/uxdb/uxdbinstall/dbsql/bin/restore -j 4
WARNING: Skipping backup QILCGE, because it has non-valid status: ERROR
WARNING: Skipping backup QILBL8, because it has non-valid status: ERROR
INFO: Validating parents for backup QILAJ1
INFO: Validating backup QILA2N
INFO: Backup QILA2N data files are valid
INFO: Validating backup QILAD0
INFO: Backup QILAD0 data files are valid
INFO: Validating backup QILAJ1
INFO: Backup QILAJ1 data files are valid
INFO: Backup QILAJ1 WAL segments are valid
INFO: Backup QILAJ1 is valid.
INFO: Restoring the database from backup at 2020-10-22 14:11:25+08
INFO: Start restoring backup files. UXDATA size: 51MB
INFO: Backup files are restored. Transfered bytes: 51MB, time elapsed: 0
INFO: Restore incremental ratio (less is better): 100% (51MB/51MB)
INFO: Syncing restored files to disk
INFO: Restored backup files are synced, time elapsed: 0
INFO: Restore of backup QILAJ1 completed.
[uxdb@local65 bin]$
```

   b. 启动恢复后的数据库。

   启动时会提示没有权限，需要修改restore目录的权限。

   chmod 0700 restore/

启动需要使用绝对路径，否则启动失败。

./ux_ctl -D /home/uxdb/uxdbinstall/dbsql/bin/restore/ start

c. 连接数据库查看t1表的内容。

```
[uxdb@local65 bin]$ ./uxsql
Password for user uxdb:
uxsql (2.1.1.3)
Type "help" for help.

uxdb=# select * from t1;
 id
----
  1
  2
  3
  4
  5
  6
  7
  8
 11
 12
 13
 14
 15
(13 rows)

uxdb=#
```

2. 按时间点恢复。

以将数据库实例test从备份中恢复到时间点2020-10-24 14:47:33为例。

a. 停掉test实例，并将其重命名。

.[uxdb@local65 bin]$ ./ux_ctl -D test/ stop
[uxdb@local65 bin]$ mv test test_bak
[uxdb@local65 bin]$ ./ux_probackup restore -B /home/uxdb/data/probackup --instance=test1
--recovery-target-time='2020-10-24 14:47:33'

恢复命令执行之后可以看到bin目录下除了之前重命名之后的test_bak目录之外，还有恢复后的实例test目录。

```
[uxdb@local65 bin]$ ls
clusterdb   dropdb    initdb     removedb  ux_archivecleanup  ux_checksums    ux_ctl
createdb    dropuser  oid2name   test      ux_basebackup      ux_config       uxdb
createuser  ecux      reindexdb  test_bak  uxbench            ux_controldata  ux_diagnose
```

b. 启动数据库（必须用绝对路径）。

[uxdb@local65 bin]$ ./ux_ctl -D /home/uxdb/uxdbinstall/dbsql/bin/test start

查看表数据库。

```
[uxdb@local65 bin]$ ./uxsql
Password for user uxdb:
uxsql (2.1.1.3)
Type "help" for help.

uxdb=# select * from  t1;
 id
----
  1
  2
  3
  4
  5
  6
  7
  8
 11
 12
 13
 14
 15
(13 rows)

uxdb=#
```

1.15.4.10. 远程操作

1. 配置ssh秘钥

   在两台机器上分别生成ssh，默认保存在~/.ssh目录中。

   环境：本地机器 192.71.0.88，远程机器 192.71.0.86。

   在本地机器上执行：

   ssh-keygen -t rsa
   cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys65
   chmod 600 ~/.ssh/authorized_keys65

```
[uxdb@local65 ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/uxdb/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/uxdb/.ssh/id_rsa.
Your public key has been saved in /home/uxdb/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:jRTABkDns5cEGgrEO1thWxD5sSOVWnSFpUD21an7mVw uxdb@local65
The key's randomart image is:
+---[RSA 2048]----+
|+o++XBoo=+. .    |
|.o O.B*.o. o     |
|. + @.+o. .      |
| o = B o +       |
|  + o + S o      |
| .   .   .   E   |
|          o +    |
|           =     |
|                 |
+----[SHA256]-----+
[uxdb@local65 ~]$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys65
[uxdb@local65 ~]$ chmod 600 ~/.ssh/authorized_keys65
[uxdb@local65 ~]$
```

在远程机器上执行：

ssh-keygen -t rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys64
chmod 600 ~/.ssh/authorized_keys64

```
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/uxdb/.ssh/id_rsa.
Your public key has been saved in /home/uxdb/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:LUHahx10H0u96l4RraKmt9u8WxcyWhx0/uY/8QPqeg4 uxdb@local64
The key's randomart image is:
+---[RSA 2048]----+
|        ..o ..oo |
|       + o o.oooo|
|      . + o  .ooo|
|         +  . .oo|
|        S . .=ooo|
|         . .o+o+o|
|         Eo.o o.=|
|         oo= o.+o|
|        .oB=*+  +|
+----[SHA256]-----+
[uxdb@local64 ~]$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys64
[uxdb@local64 ~]$ chmod 600 ~/.ssh/authorized_keys64
[uxdb@local64 ~]$
```

2. 复制公钥

   分别将本地机器的公钥复制到远程机器，远程机器的公钥复制到本地机器。

   scp ~/.ssh/authorized_keys65 192.71.0.86:~/.ssh/

```
[uxdb@local65 ~]$ scp ~/.ssh/authorized_keys65 192.71.0.86:~/.ssh/
The authenticity of host '192.71.0.86 (192.71.0.86)' can't be established.
ECDSA key fingerprint is SHA256:/swLcTLPAm+UhocyDZxkEANxMWm0dDTnWBJgXbY40Pw.
ECDSA key fingerprint is MD5:c6:56:ca:85:f6:83:25:3a:bb:bd:e3:4f:91:b7:e7:db.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.71.0.86' (ECDSA) to the list of known hosts.
uxdb@192.71.0.86's password:
authorized_keys65
[uxdb@local65 ~]$
```

```
[uxdb@local64 ~]$ scp ~/.ssh/authorized_keys64 192.71.0.88:~/.ssh/
The authenticity of host '192.71.0.88 (192.71.0.88)' can't be established.
ECDSA key fingerprint is SHA256:/swLcTLPAm+UhocyDZxkEANxMWm0dDTnWBJgXbY40Pw.
ECDSA key fingerprint is MD5:c6:56:ca:85:f6:83:25:3a:bb:bd:e3:4f:91:b7:e7:db.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.71.0.88' (ECDSA) to the list of known hosts.
uxdb@192.71.0.88's password:
authorized_keys64
[uxdb@local64 ~]$
```

3. 配置公钥

   在两台机器上配置公钥并验证。

   mv ~/.ssh/authorized_keys64 ~/.ssh/authorized_keys

```
[uxdb@local65 ~]$ mv ~/.ssh/authorized_keys64 ~/.ssh/authorized_keys
[uxdb@local65 ~]$
```

```
[uxdb@local64 ~]$ mv ~/.ssh/authorized_keys65 ~/.ssh/authorized_keys
[uxdb@local64 ~]$
```

验证是否成功。

```
[uxdb@local65 ~]$ ssh 192.71.0.86
Last login: Sat Oct 24 16:20:05 2020 from 192.71.1.101
[uxdb@local64 ~]$
```

4. 配置远程实例

   修改远程实例的配置文件uxsinodb.conf。

   max_wal_senders = 10 #根据实际场景设置合理值，此处使用默认值10
   wal_level = 'replica'
   archive_mode = 'on'
   archive_command = '/home/uxdb/uxdbinstall/dbsql/bin/ux_probackup archive-push -B /home/uxdb/data/probackup --instance test2 --remote-user=uxdb --remote-host=192.71.0.88 --remote-port=22 --remote-path=/home/uxdb/uxdbinstall/dbsql/bin/ --wal-file-path=%p --wal-file-name=%f '

5. 添加远程实例

   本地机器执行命令：

   ux_probackup add-instance -B /home/uxdb/data/probackup -D /home/uxdb/uxdbinstall/dbsql/bin/test/ --instance=test2 --remote-host=192.71.0.86 --remote-port=22 --remote-user=uxdb --remote-path=/home/uxdb/uxdbinstall/dbsql/bin/

6. 全量备份

   [uxdb@local65 bin]$ ./ux_probackup backup -B /home/uxdb/data/probackup --instance=test2 --remote-user=uxdb --remote-host=192.71.0.86 --remote-port=22 --remote-path=/home/uxdb/uxdbinstall/dbsql/bin/ -b full

```
[uxdb@local65 bin]$ ./ux_probackup backup -B /home/uxdb/data/probackup --instance=test2 --remote-user=uxdb --remote-host=192.71.0.86 --remote-port=22 --remote-path=/home/uxdb/uxdbinstall/dbsql/bin/ -b full
INFO: Backup start, ux_probackup version: 2.4.4, instance: test2, backup ID: QISC7U, backup mode: FULL, wal mode: ARCHIVE, remote: true, compress-algorithm: none, compress-level: 1
Password:
WARNING: This UXsinoDB instance was initialized without data block checksums. ux_probackup have no way to detect data block corruption without them. Reinitialize UXDATA with option '--data-checksums'.
WARNING: Current UXsinoDB role is superuser. It is not recommended to run backup or checkdb as superuser.
INFO: Wait for WAL segment /home/uxdb/data/probackup/wal/test2/000000010000000000000022 to be archived
INFO: UXDATA size: 51MB
INFO: Start transferring data files
INFO: Data files are transferred, time elapsed: 2s
INFO: wait for ux_stop_backup()
INFO: ux_stop backup() successfully executed
INFO: Wait for LSN 0/230000B8 in archived WAL segment /home/uxdb/data/probackup/wal/test2/000000010000000000000023
INFO: Syncing backup files to disk
INFO: Backup files are synced, time elapsed: 1s
INFO: Validating backup QISC7U
INFO: Backup QISC7U data files are valid
INFO: Backup QISC7U resident size: 51MB
INFO: Backup QISC7U completed
[uxdb@local65 bin]$
```

7. 增量备份

   [uxdb@local65 bin]$ ./ux_probackup backup -B /home/uxdb/data/probackup --instance=test2 --remote-user=uxdb --remote-host=192.71.0.86 --remote-port=22 --remote-path=/home/uxdb/uxdbinstall/dbsql/bin/ -b page

```
[uxdb@local65 bin]$ ./ux_probackup backup -B /home/uxdb/data/probackup --instance=test2 --remote-user=uxdb --remote-host=192.71.0.86 --remote-port=22 --remote-path=/
home/uxdb/uxdbinstall/dbsql/bin/ -b page
INFO: Backup start, ux_probackup version: 2.4.4, instance: test2, backup ID: QISEM5, backup mode: PAGE, wal mode: ARCHIVE, remote: true, compress-algorithm: none, co
mpress-level: 1
Password:
WARNING: This UXsinoDB instance was initialized without data block checksums. ux_probackup have no way to detect data block corruption without them. Reinitialize UXD
ATA with option '--data-checksums'.
WARNING: Current UXsinoDB role is superuser. It is not recommended to run backup or checkdb as superuser.
INFO: Wait for WAL segment /home/uxdb/data/probackup/wal/test2/000000010000000000000026 to be archived
INFO: Parent backup: QISC7U
INFO: UXDATA size: 51MB
INFO: Extracting pagemap of changed blocks
INFO: Pagemap successfully extracted, time elapsed: 0 sec
INFO: Start transferring data files
INFO: Data files are transferred, time elapsed: 1s
INFO: wait for ux_stop_backup()
INFO: ux_stop backup() successfully executed
INFO: Wait for LSN 0/270000B8 in archived WAL segment /home/uxdb/data/probackup/wal/test2/000000010000000000000027
INFO: Syncing backup files to disk
INFO: Backup files are synced, time elapsed: 0
INFO: Validating backup QISEM5
INFO: Backup QISEM5 data files are valid
INFO: Backup QISEM5 resident size: 204kB
INFO: Backup QISEM5 completed
[uxdb@local65 bin]$
```

8. 远程恢复

恢复到2020-10-26 09:32:25时间节点。

```
uxdb=# select * from student;
 id
----
 55
 55
 55
 55
(4 rows)

uxdb=# select now();
              now
-------------------------------
 2020-10-26 09:32:25.532345+08
(1 row)
```

停止远程机器上的实例test,并进行重命名。

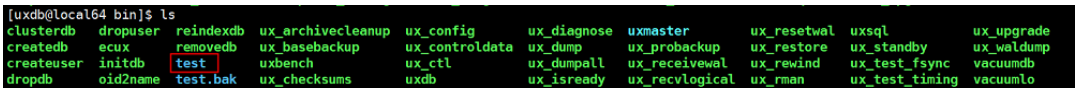
./ux_ctl -D test stop
mv test test.bak

本地机器执行恢复命令。

./ux_probackup restore -B /home/uxdb/data/probackup --instance=test2 --recovery-target-time='2020-10-26 09:32:25' --remote-user=uxdb --remote-host=192.71.0.86 --remote-port=22 --remote-path=/home/uxdb/uxdbinstall/dbsql/bin/ --archive-host=192.71.0.88 --archive-port=22 --archive-user=uxdb

```
[uxdb@local65 bin]$ ./ux_probackup restore -B /home/uxdb/data/probackup --instance=test2 --recovery-target-time='2020-10-26 09:32:25' --remote-user=uxdb --remote-ho
st=192.71.0.86 --remote-port=22 --remote-path=/home/uxdb/uxdbinstall/dbsql/bin/ --archive-host=192.71.0.88 --archive-port=22 --archive-user=uxdb
INFO: Validating backup QISC7U 远程机器ip                                                    本地归档机器的ip
INFO: Backup QISC7U data files are valid
INFO: Backup validation completed successfully on time 2020-10-26 10:09:21+08, xid 510 and LSN 0/24000250
INFO: Backup QISC7U is valid.
INFO: Restoring the database from backup at 2020-10-26 09:31:06+08
INFO: Start restoring backup files. UXDATA size: 51MB
INFO: Backup files are restored. Transfered bytes: 51MB, time elapsed: 3s
INFO: Restore incremental ratio (less is better): 100% (51MB/51MB)
INFO: Syncing restored files to disk
INFO: Restored backup files are synced, time elapsed: 1s
INFO: Restore of backup QISC7U completed.
[uxdb@local65 bin]$
```

远程机器查看恢复后的实例数据。



启动恢复后的实例时必须以绝对路径启动。

[uxdb@local64 bin]$ ./ux_ctl -D /home/uxdb/uxdbinstall/dbsql/bin/test start
[uxdb@local64 bin]$ ./uxsql

```
uxmaster status starting
uxmaster status starting
uxmaster status starting
uxmaster status starting
uxmaster status starting
uxmaster status ready
 done
server started
[uxdb@local64 bin]$ ./uxsql
Password for user uxdb:
uxsql (2.1.1.3)
Type "help" for help.

uxdb=# select * from student ;
 id
----
 55
 55
 55
 55
(4 rows)

uxdb=#
```

注意

恢复时间节点有限制范围，只能使执行备份的时间范围内，时间范围外恢复失败。

1.16. ux_receivewal

ux_receivewal — 以流的方式从一个UXDB服务器得到预写日志

1.16.1. 用法

ux_receivewal [*option*...]

1.16.2. 描述

ux_receivewal被用来从一个运行状态的UXDB集群以流的方式得到预写式日志。预写式日志会被使用流复制协议以流的方式传送，并且被写入到文件的一个本地目录。这个目录可以被用作归档位置来做一次使用时间点恢复的恢复。

当预写式日志在服务器上被产生时，ux_receivewal实时以流的方式传输预写式日志，并且不像archive_command那样等待段完成。由于这个原因，在使用ux_receivewal时不必设置archive_command。

与 UXDB 备用服务器上的 WAL 接收进程不同，ux_receivewal默认只在一个WAL文件被关闭时才刷入WAL数据。要实时刷入WAL数据，必须指定选项--synchronous。 由于ux_receivewal不应用于WAL，当synchronous_commit等于remote_apply时，不允许它成为同步备用。如果发生这样的情况，它将成为一个永远不能拉起的备用数据库，并且会导致事务提交阻塞。 为了避免这种情况，你应该为synchronous_standby_names配置一个适当的值，或规定为ux_receivewal的application_name 与它不匹配，或将synchronous_commit的值更改为remote_apply以外的内容。

预写式日志对一个常规UXDB连接进行流处理，并且使用复制协议。连接必须由一个超级用户或一个具有REPLICATION权限的用户建立，并且ux_hba.conf必须允许复制连接。服务器还必须将max_wal_senders设置得足够高，以便至少为流处理预留下一个可用的会话。

如果该连接丢失，或者最初无法建立连接，并且出现非致命性错误，ux_receivewal将无限期地重试连接并且尽可能重新建立流。为了避免这种行为，使用-n参数。

如果不出现致命错误，ux_receivewal将一直运行直至被SIGINT信号（Control+C）终止。

1.16.3. 选项

-D *directory*
--directory=*directory*

要把输出写到哪个目录。

这个参数是必需的。

-E *lsn*
--endpos=*lsn*

当接收到达指定的LSN时，自动停止复制并且以正常退出状态0退出。

如果有一个记录的LSN正好等于*lsn*，则该记录将会被处理。

--if-not-exists

当指定--create-slot并且具有指定名称的槽已经存在时不要抛出错误。

-n
--no-loop

不要在连接错误上循环。相反，碰到一个错误时立刻退出。

--no-sync

这个选项导致ux_receivewal不强制WAL数据被刷回磁盘。这样会更快，但是也意味着接下来的操作系统崩溃会让WAL段损坏。通常，这个选项对于测试有用，但不应该在对生产部署进行WAL归档时使用。

这个选项与--synchronous不兼容。

-s *interval*
--status-interval=*interval*

指定发送回服务器的状态包之间的秒数。这允许我们更容易地监控服务器的进度。0表示完全禁用这种周期性的状态更新，不过当服务器需要时还是会发送更新避免超时导致的断开连接。默认值是10秒。

-S *slotname*
--slot=*slotname*

要求ux_receivewal使用一个已有的复制槽。在使用这个选项时，ux_receivewal将会报告给服务器一个刷新位置，指示每一个段是何时被同步到磁盘的，这样服务器可以在不需要该段时移除它。

当ux_receivewal的复制客户端在服务器上被配置为一个同步备用时，那么使用复制槽将会向服务器报告刷新位置，但只在一个WAL文件被关闭时报告。因此，该配置将导致主服务器上的事务等待很长的时间并且无法令人满意地工作。要使配置正确，还必须同时使用选项--synchronous。

--synchronous

在WAL数据被收到后立即刷入到磁盘。还要在刷新后立即向服务器回送一个状态包（不考虑--status-interval）。

如果ux_receivewal的复制客户端在服务器上被配置为一个同步备用，应该指定这个选项来确保向服务器发送及时反馈。

-v
--verbose

启用详细输出模式。

-Z *level*
--compress=*level*

启用预写式日志上的gzip压缩，并且指定压缩级别（0到9，0是不压缩而9是最大压缩）。所有的文件名后都将被追加后缀.gz。

下列命令行选项控制数据库连接参数。

-d *connstr*
--dbname=*connstr*

指定用于连接到服务器的参数为一个连接字符串。

为了和其他客户端应用一致，该选项被称为--dbname。但是因为ux_receivewal并不连接到集群中的任何特定数据库，连接字符串中的数据库名将被忽略。

-h *host*
--host=*host*

指定运行服务器的机器的主机名。如果该值以一个斜线开始，它被用作 Unix 域套接字的目录。默认值取自UXHOST环境变量（如果设置），否则会尝试一个 Unix 域套接字连接。

-p *port*
--port=*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。默认用UXPORT环境变量中的值（如果设置），或者一个编译在程序中的默认值。

-U *username*
--username=*username*

要作为哪个用户连接。

-w
--no-password

永远不要发出密码提示。如果服务器需要密码验证，而通过其他方式(如.pgpass文件)无法获得密码，则连接尝试将失败。此选项在没有用户可以输入密码的批处理作业和脚本中很有用。

-W
--password

强制ux_receivewal在连接到一个数据库之前提示输入密码。

这个选项不是必需的，因为如果服务器要求输入密码，ux_receivewal将自动提示输入密码。但是，ux_receivewal将浪费一次连接尝试来发现服务器想要一个密码。在某些情况下建议用-W来避免额外的连接尝试。

为了控制物理复制槽，ux_receivewal可以执行下列两种动作之一：

--create-slot

用--slot中指定的名称创建一个新的物理复制槽，然后退出。

--drop-slot

删除--slot中指定的复制槽，然后退出。

其他选项也可用：

-V
--version

打印ux_receivewal版本并退出。

-?
--help

显示有关ux_receivewal命令行参数的帮助并退出。

1.16.4. 退出状态

在被SIGINT信号终止（没有正常的方式结束它。因此这不是一种错误）时，ux_receivewal将以状态0退出。 对于致命错误或者其他信号，退出状态将不是零。

1.16.5. 环境变量

UX_COLOR

规定在诊断消息中是否使用颜色。可能的值为always、auto、never。

和大部分其他UXDB工具相似，这个工具也使用libuxsql支持的环境变量。

1.16.6. 注解

在使用ux_receivewal替代archive-command作为主要的WAL备份方法时，强烈建议使用复制槽。否则，服务器可能会在预写式日志文件被备份好之前重用或者移除它们，因为没有任何信息（不管是来自archive-command或是复制槽）能够指示WAL流已经被归档到什么程度。不过要注意，如果接收者没有持续地取走WAL数据，一个复制槽将会填满服务器的磁盘空间。

如果在源集簇上启用了组权限，ux_receivewal将保留接收到的WAL文件上的组权限。

1.16.7. 示例

要从位于mydbserver的服务器流式传送预写式日志并且将它存储在本地目录/home/uxdb/uxdbinstall/dbsql/archive：

$ **ux_receivewal -h mydbserver -D /home/uxdb/uxdbinstall/dbsql/archive**

1.17. ux_recvlogical

ux_recvlogical — 控制UXDB逻辑解码流

1.17.1. 用法

ux_recvlogical [*option...*]

1.17.2. 描述

ux_recvlogical控制逻辑解码复制槽以及来自这种复制槽的流数据。

它会创建一个复制模式的连接，因此它受到和ux_receivewal相同的约束，还有逻辑复制的约束。

ux_recvlogical不同于逻辑解码SQL接口的监控和获取模式。它会在收到数据并退出时延迟地发送重播确认。检查插槽上未处理的数据而不使用它，使用ux_logical_slot_peek_changes。

1.17.3. 选项

必须至少要指定下列选项之一来选择一项操作：

--create-slot

为--dbname指定的数据库用--slot指定的名称创建一个新的逻辑复制槽，使用--plugin指定的输出插件。

--drop-slot

删除名称由--slot指定的复制槽，然后退出。

--start

从--slot指定的逻辑复制槽开始进行流式传送更改，一直继续到被一个信号终止。如果服务器端关机或者断开连接导致更改流结束，会进入一个循环一直重试，通过指定--no-loop可以防止这种情况下进入循环重试。

流格式由槽创建时指定的输出插件决定。

连接必须是连接到用于创建该槽的同一个数据库上。

--create-slot和--start可以被一起指定。 --drop-slot不能和另一个动作组合在一起。

下面的命令行选项控制输出的位置和格式以及其他复制行为：

-E *lsn*
--endpos=*lsn*

在--start模式中，当接收过程到达指定的LSN时会自动地停止复制并且以正常的退出状态0退出。如果不处于--start模式时指定这个选项，则会发生错误。

如果有一个记录的LSN正好等于*lsn*，则该记录将被输出。

--endpos选项不知道事务边界，并可能在事务中途截断输出。 任何部分输出的事务都不会被使用，并且在下一次读取插槽时将再次播放。个人消息不会被截断。

-f *filename*
--file=*filename*

把接收到并且解码好的事务数据写入到一个文件。使用-可以写到stdout。

-F *interval_seconds*
--fsync-interval=*interval_seconds*

指定ux_recvlogical发出 fsync()调用确保输出文件被安全地刷到磁盘的频度。

服务器将会间隔设置时间要求客户端执行一次刷新并且把刷新位置报告给服务器。这个设置可以在此之外更加频繁地执行刷新。

指定间隔为0会完全禁止发出fsync() 调用，但是仍会报告进度给服务器。在这种情况下，发生崩溃会导致数据丢失。

-I *lsn*
--startpos=*lsn*

在--start模式中，从给定的LSN开始复制。在其他模式中会忽略这个参数。

--if-not-exists

当指定--create-slot并且具有指定名称 的槽已经存在时不要抛出错误。

-n
--no-loop

当服务器连接丢失时，不要在循环中重试，直接退出。

-o *name*[=*value*]
--option=*name*[=*value*]

如果指定了输出插件，把选项值*value* 传递给选项*name*。存在哪些选项以及它们的效果 取决于使用的输出插件。

-P *plugin*
--plugin=*plugin*

在创建一个槽时使用指定的逻辑解码输出插件。如果该槽已经存在，这个选项没有效果。

-s *interval_seconds*
--status-interval=*interval_seconds*

这个选项和ux_receivewal中的同名选项效果相同。

-S *slot_name*
--slot=*slot_name*

在--start模式中，使用名为*slot_name* 的已有逻辑复制槽。在--create-slot模式中，使用这个名称 创建该槽。在--drop-slot模式中，删除这个名称指定的槽。

-v
--verbose

开启详细输出模式。

下列命令行选项控制数据库连接参数。

-d *database*
--dbname=*database*

要连接的数据库。它可以是一个libuxsql连接字符串。默认为用户名。

-h *hostname-or-ip*
--host=*hostname-or-ip*

指定服务器正在运行的机器的主机名。如果该值开始于一个斜线， 它被用作一个 Unix 域套接字的目录。默认是从 UXHOST环境变量中取得（如果被设置）， 否则将尝试一次 Unix 域套接字连接。

-p *port*
--port=*port*

指定服务器正在监听连接的 TCP 端口或本地 Unix 域套接字文件扩展名。 默认是放在UXPORT环境变量中（如果被设置）， 否则使用编译在程序中的默认值。

-U *user*
--username=*user*

要作为哪个用户连接。默认是用当前操作系统用户名。

-w
--no-password

永远不要发出密码提示。如果服务器需要密码验证，而通过其他方式(如.pgpass文件)无法获得密码，则连接尝试将失败。此选项在没有用户可以输入密码的批处理作业和脚本中很有用。

-W
--password

强制ux_dump在连接到一个数据库之前提示输入密码。

这个选项不是必须的，因为如果服务器要求输入密码，ux_dump将自动提示输入密码。但是，ux_dump将浪费一次连接尝试来发现服务器想要一个密码。在某些情况下，建议输入-W来避免额外的连接尝试。

还有下列附加选项可用：

-V
--version

打印ux_recvlogical的版本并且退出。

-?
--help

显示关于ux_recvlogical命令行参数的帮助，并且退出。

1.17.4. 环境变量

UX_COLOR

规定在诊断消息中是否使用颜色。可能的值为always、auto、never。

和大部分其他UXDB工具相似，这个工具也使用libuxsql支持的环境变量。

1.17.5. 注解

如果在源服务器上启用了组权限，ux_recvlogical将会在接收到的WAL文件上保留组权限。

1.17.6. 示例

$ **ux_recvlogical -d UXDB --slot test --start -f -**
Control+Z

$ **uxsql -d UXDB -c "INSERT INTO data(data) VALUES('4');"**
$**fg**
BEGIN 693
table public.data: INSERT: id[integer]:4 data[text]:'4'
COMMIT 693
Control+C

$ **ux_recvlogical -d UXDB --slot test --drop-slot**

1.18. ux_restore

ux_restore — 从一个由ux_dump创建的归档文件恢复一个UXDB数据库

1.18.1. 用法

ux_restore [*connection-option...*] [*option...*] [*filename*]

1.18.2. 描述

ux_restore是一个用来从ux_dump创建的非文本格式归档恢复UXDB数据库的工具。它将发出必要的命令把该数据库重建成它被保存时的状态。这些归档文件还允许ux_restore选择恢复哪些内容或者在恢复前对恢复项重排序。这些归档文件被设计为可以在不同的架构之间迁移。

ux_restore可以在两种模式下操作。如果指定了一个数据库名称，ux_restore会连接那个数据库并且把归档内容直接恢复到该数据库中。否则，会创建一个脚本，其中包含着重建该数据库所必要的SQL命令，它会被写入到一个文件或者标准输出。这个脚本输出等效于ux_dump的纯文本输出格式。因此，一些控制输出的选项与ux_dump的选项类似。

显然，ux_restore无法恢复不在归档文件中的信息。例如，如果归档使用“以INSERT命令转储数据”选项创建， ux_restore将无法使用COPY语句装载数据。

1.18.3. 选项

ux_restore接受下列命令行参数。

filename

指定要被恢复的归档文件（对于一个目录格式的归档则是目录）的位置。如果没有指定，则使用标准输入。

-a
--data-only

只恢复数据，不恢复模式（数据定义）。如果在归档中存在，表数据、大对象和序列值会被恢复。

这个选项类似于指定--section=data，但两者不完全相同。

-c
--clean

在重新创建数据库对象之前清除（丢弃）它们（除非使用了--if-exists，如果有对象在目标数据库中不存在，这可能会生成一些无害的错误消息）。

-C
--create

在恢复一个数据库之前先创建它。如果还指定了--clean，在连接到目标数据库之前丢弃并且重建它。

如果使用--create，ux_restore还会恢复数据库的注释（如果有）以及与其相关的配置变量设置，也就是任何提到过这个数据库的ALTER DATABASE ... SET ...和ALTER ROLE ... IN DATABASE ... SET ...命令。不管是否指定--no-acl，数据库本身的访问特权都会被恢复。

在使用这个选项时，-d提到的数据库只被用于发出初始的DROP DATABASE和CREATE DATABASE命令。所有要恢复到该数据库名中的数据都出现在归档中。

-d *dbname*
--dbname=*dbname*

连接到数据库*dbname*并且直接恢复到该数据库中。

--running-mode = standard | compatible | mysql

运行模式，默认模式是standard。standard表示标准模式，compatible表示兼容模式，mysql表示mysql模式。

-e
--exit-on-error

在发送SQL命令到该数据库期间如果碰到一个错误就退出。默认行为是继续并且在恢复结束时显示错误计数。

-f *filename*
--file=*filename*

为生成的脚本或列表（当使用-l时）指定输出文件。默认是标准输出。

-F *format*
--format=*format*

指定归档的格式。并不一定要指定该格式，因为ux_restore将会自动决定格式。如果指定，可以是下列之一：

c
custom

ux_dump的自定义格式。

d
directory

目录归档。

t
tar

tar归档。

-I *index*
--index=*index*

只恢复提及的索引的定义。可以通过写多个-I指定多个索引。

-j *number-of-jobs*
--jobs=*number-of-jobs*

使用并发任务运行ux_restore中最耗时的部分 — 载入数据、创建索引或者创建约束。对于一个运行在多处理器机器上的服务器，这个选项能够大幅度减少恢复一个大型数据库的时间。

每一个任务是一个进程或者一个线程，这取决于操作系统，它们都使用一个独立的服务器连接。

这个选项的最佳值取决于服务器、客户端以及网络的硬件设置。因素包括CPU的核心数和磁盘设置。一个好的建议是服务器上CPU的核心数，但是更大的值在很多情况下也能导致更快的恢复时间。当然，过高的值会由于超负荷反而导致性能降低。

这个选项只支持自定义和目录归档格式。输入必须是一个常规文件或目录（例如，不能是一个管道）。当发出一个脚本而不是直接连接到数据库服务器时会忽略这个选项。还有，多任务不能和选项--single-transaction一起用。

-l
--list

列出归档的内容的表。这个操作的输出能被用作-L选项的输入。注意如果把-n或-t这样的过滤开关与-l一起使用，它们将会限制列出的项。

-L *list-file*
--use-list=*list-file*

只恢复在*list-file*中列出的归档元素，并且按照它们出现在该文件中的顺序进行恢复。注意如果把-n或-t这样的过滤开关与-L一起使用，它们将会进一步限制要恢复的项。

*list-file*通常是编辑一个-l操作的输出来创建。行可以被移动或者移除，并且也可以通过在行首放一个（;）将其注释掉。

-n *shcema*
--schema=*schema*

只恢复在被提及的模式中的对象。可以用多个-n开关来指定多个模式。这可以与-t选项组合在一起只恢复一个指定的表。

-N *schema*
--exclude-schema=*schema*

不要恢复所提及方案中的对象。可以使用多个-N指定多个要被排除的模式。

如果对同一个方案名称同时给出了-n和-N后时，-N优先级更高，排除它指出的模式。

-O
--no-owner

不要输出将对象的所有权设置为与原始数据库匹配的命令。默认情况下，ux_restore会发出ALTER OWNER或者SET SESSION AUTHORIZATION语句来设置已创建的模式对象的所有权。除非到该数据库的初始连接是一个超级用户（或者拥有脚本中所有对象的同一个用户）建立的，这些语句将会失败。通过-O，任何用户名都可以被用于初始连接，并且这个用户将会拥有所有被创建的对象。

-P *function-name(argtype [, ...])*
--function=*function-name(argtype [, ...])*

只恢复被提及的函数。要拼写函数的名称和参数使它们正好就是出现在转储文件的内容表中的名称和参数。可以使用多个-P开关指定多个函数。

-s
--schema-only

只恢复归档中的模式（数据定义）不恢复数据。

这个选项是--data-only的逆选项。它与指定--section=pre-data --section=post-data相似，但并不完全相同。

（不要把这个选项和--schema选项弄混，后者把词“schema”用于一种不同的含义）。

-S *username*
--superuser=*username*

指定在禁用触发器时要用的超级用户名。只有使用--disable-triggers时这个选项才有含义。

--section=*sectionname*

pre-data、data或者post-data是将备份文件分成了3个小节，分别对应数据前的部分，数据部分，数据后的部分。

完整的恢复流程如下所示。

1. 先恢复--section=pre-data。
2. 再恢复--section=data。
3. 最后恢复--section=post-data。

如果没有恢复pre-data，而直接恢复data或者post-data，就会报没有表的错误，因为创建表、设置参数等都是在pre-data阶段完成。

-t *table*
--table=*table*

只恢复所提及的表的定义和数据。包括视图、物化视图、序列和外部表。可以写上多个-t可以选择多个表。这个选项可以和-n选项结合在一起指定一个特定模式中的表。

> **注意**
>
> 在指定-t时，ux_restore不会尝试恢复所选表可能依赖的任何其他数据库对象。因此，无法确保能成功地把一个特定表恢复到一个干净的数据库中。
>
> 这个标志的行为和ux_dump的-t标志不一样。在ux_restore中当前没有任何通配符匹配的规定，也不能在其-t选项中包括模式的名称。

-T *trigger*
--trigger=*trigger*

只恢复所提及的触发器。可以用多个-T指定多个触发器。

-v
--verbose

开启详细输出模式。

-V
--version

打印该ux_restore的版本并退出。

-x
--no-privileges
--no-acl

禁止恢复访问特权（grant/revoke命令）。

-1
--single-transaction

将恢复作为单一事务执行（即把发出的命令包裹在BEGIN/COMMIT中）。这可以确保要么所有命令完全成功，要么任何改变都不被应用。这个选项隐含了--exit-on-error。

--disable-triggers

只有在执行一个只恢复数据的恢复时，这个选项才相关。它指示ux_restore在加载数据时执行命令临时禁用目标表上的触发器。如果你在表上有参照完整性检查或者其他触发器并且你不希望在数据载入期间调用它们时，请使用这个选项。

目前，为--disable-triggers发出的命令必须以超级用户身份完成。因此你还应该用-S指定一个超级用户名，或者更好的方法是以一个UXDB超级用户运行ux_restore。

--enable-row-security

只有在恢复具有行安全性的表的内容时，这个选项才相关。默认情况下，ux_restore将把row-security设置为 off 来确保所有数据都被恢复到表中。如果用户不拥有足够绕过行安全性的特权，那么会错误。这个参数指示ux_restore把row-security设置为on允许用户尝试恢复启用了行安全性的表的内容。如果用户没有从转储向表中插入行的权限，这仍将失败。

注意当前这个选项还要求转储处于INSERT格式，因为COPY FROM不支持行安全性。

--if-exists

在清理数据库对象时使用条件命令（即增加一个IF EXISTS子句）。只有指定了--clean时，这个选项才有效。

--no-comments

即便归档中包含注释也不输出恢复注释的命令。

--no-data-for-failed-tables

默认情况下，即便表的创建命令失败（例如因为表已经存在），表数据也会被恢复。通过这个选项，对这类表的数据会被跳过。如果目标数据库已经包含了想要的表内容，这种行为很有有用。例如，UXDB扩展的辅助表可能已经被载入到目标数据库中，指定这个选项就能阻止把重复的或者废弃的数据载入到这些表中。

只有当直接恢复到一个数据库中时这个选项才有效，在产生SQL脚本输出时这个选项不会产生效果。

--no-publications

即便归档中包含发布也不输出恢复发布的命令。

--no-security-labels

不恢复安全标签，即使归档中包含安全标签。

--no-subscriptions

即便归档中包含订阅也不输出恢复订阅的命令。

--no-tablespaces

不输出命令选择表空间。通过这个选项，所有的对象都会被创建在恢复时的默认表空间中。

--section=*sectionname*

只恢复提及的节。节的名称可以是pre-data、data或者post-data。可以把这个选项指定多次来选择多个节。默认值是恢复所有节。

数据节包含实际的表数据以及大对象定义。Pre-data项由所有其他数据定义项构成。Post-data项由索引定义、触发器、规则和除已验证的检查约束之外的约束构成。

--strict-names

要求每一个模式（-n/--schema）以及表（-t/--table）限定词匹配备份文件中至少一个模式/表。

--use-set-session-authorization

输出SQL标准的SET SESSION AUTHORIZATION命令取代ALTER OWNER命令来决定对象拥有权。这会让转储更加兼容标准，但是依赖于转储中对象的历史，可能无法正确恢复。

-?
--help

显示有关ux_restore命令行参数的帮助，并且退出。

ux_restore也接受下列用于连接参数的命令行参数：

-h *host*
--host=*host*

指定服务器正在运行的机器的主机名。如果该值开始于一个斜线，它被用作一个 Unix 域套接字的目录。默认是从UXHOST环境变量中取得（如果被设置），否则将尝试一次 Unix 域套接字连接。

-p *port*
--port=*port*

指定服务器正在监听连接的 TCP 端口或本地 Unix 域套接字文件扩展名。默认是放在UXPORT环境变量中（如果被设置），否则使用编译在程序中的默认值。

-U *username*
--username=*username*

要作为哪个用户连接。

-w
--no-password

不发出一个密码提示。如果服务器要求输入密码并且没有其他方式提供密码（例如一个.uxpass文件），那么连接尝试将失败。这个选项对于批处理任务和脚本有用，因为在其中没有一个用户来输入密码。

-W
--password

强制ux_restore在连接到一个数据库之前提示输入密码。

这个选项是可选的。如果服务器要求输入密码，ux_restore将自动提示需要一个密码。但是，ux_restore将浪费一次连接来发现服务器需要一个密码。在某些情况下，使用-W来避免额外的连接。

--role=*rolename*

指定一个用来创建该转储的角色名。这个选项导致ux_restore在连接到数据库后发出一个SET ROLE *rolename*命令。当已认证用户（由-U指定）缺少ux_restore所需的特权但是能够切换到一个具有所需权利的角色时，这个选项很有用。一些安装有针对直接作为超级用户登录的策略，使用这个选项可以让转储在不违反该策略的前提下完成。

1.18.4. 环境变量

UXHOST
UXOPTIONS
UXPORT
UXUSER

默认连接参数。

UX_COLOR

规定在诊断消息中是否使用颜色。可能的值为always、 auto、never。

和大部分其他UXDB工具相似，这个工具也使用libuxsql支持的环境变量。

1.18.5. 诊断

当使用-d选项指定一个直接数据库连接时，ux_restore在内部执行SELECT语句。如果你运行ux_restore时出现问题，确定你能够从正在使用的数据库中选择信息，例如uxsql。此外，libuxsql前端-后端库所使用的任何默认连接设置和环境变量都将适用。

1.18.6. 注解

如果你的数据库集簇对于template1数据库有任何本地添加，要注意将ux_restore的输出载入到一个真正的空数据库。否则你很可能由于以增加对象的重复定义而得到错误。要创建一个不带任何本地添加的空数据库，从template0而不是template1复制它，例如：

```
CREATE DATABASE foo WITH TEMPLATE template0;
```

下面将详细介绍ux_restore的局限性。

- 在恢复数据到一个已经存在的表中并且使用了选项--disable-triggers时，ux_restore会在插入数据之前发出命令禁用用户表上的触发器，然后在完成数据插入后重新启用它们。如果恢复在中途停止，可能会让系统目录处于错误的状态。
- ux_restore不能有选择地恢复大对象，例如只恢复特定表的大对象。如果一个归档包含大对象，那么所有的大对象都会被恢复，如果通过-L、-t或者其他选项进行了排除，它们一个也不会被恢复。

一旦完成恢复，应该在每一个被恢复的表上运行ANALYZE，这样优化器能得到有用的统计信息。

1.18.7. 示例

假设我们已经以自定义格式转储了一个叫做mydb的数据库：

```
$ ux_dump -Fc mydb > db.dump
```

要删除该数据库并且从转储中重新创建它：

```
$ dropdb mydb
$ ux_restore -C -d uxdb db.dump
```

-d中提到的数据库可以是任何已经存在于集群中的数据库，ux_restore只会用它来为mydb发出CREATE DATABASE命令。通过-C，数据总是会被恢复到出现在归档文件的数据库名中。

要把转储重新载入到一个名为newdb的新数据库中：

```
$ createdb -T template0 newdb
$ ux_restore -d newdb db.dump
```

> **注意**
>
> 我们不使用-C，而是直接连接到要恢复到其中的数据库。还要注意我们是从template0而不是template1创建了该数据库，以保证它最初是空的。

要对数据库项重排序，首先需要转储归档的表：

```
$ ux_restore -l db.dump > db.list
```

列表文件由头部和下面的行组成，这些行每一个都用于一个项，例如：

```
;
; Archive created at Mon Sep 14 13:55:39 2009
;     dbname: DBDEMOS
;     TOC Entries: 81
;     Compression: 9
;     Dump Version: 1.10-0
;     Format: CUSTOM
;     Integer: 4 bytes
;     Offset: 8 bytes
;     Dumped from database version: 8.3.5
;     Dumped by ux_dump version: 8.3.8
;
;
; Selected TOC Entries:
;
3; 2615 2200 SCHEMA - public pasha
1861; 0 0 COMMENT - SCHEMA public pasha
```

```
1862; 0 0 ACL - public pasha
317; 1247 17715 TYPE public composite pasha
319; 1247 25899 DOMAIN public domain0 pasha
```

分号表示开始一段注释，行首的数字表明了分配给每个项的内部归档ID。

文件中的行可以被注释掉、删除以及重排序。例如：

```
10; 145433 TABLE map_resolutions uxdb
;2; 145344 TABLE species uxdb
;4; 145359 TABLE nt_header uxdb
6; 145402 TABLE species_records uxdb
;8; 145416 TABLE ss_old uxdb
```

把这样一个文件作为ux_restore的输入将会只恢复项10和6，并且先恢复10再恢复6。

```
$ ux_restore -L db.list db.dump
```

1.19. ux_rman

ux_rman — 备份工具，支持全量，增量和差异备份

1.19.1. 用法

ux_rman [*option*] [*init* | *backup* | *restore* | *show [DATE]* | *show detail [DATE]* | *validate [DATE]* | *delete DATE* | *purge...*]

1.19.2. 描述

ux_rman是一个实用程序，用于备份和还原数据库。 它对整个数据库集群进行存档WAL和服务器日志物理备份，支持全量，增量和差异备份。

ux_rman目前仅支持Linux系统。

1.19.3. 选项

1.19.3.1. 命令选项

-D
--uxdata=PATH

数据存储目录的路径。

-A
--arclog-path=PATH

归档WAL日志的路径。

-S
--srvlog-path=PATH

存储服务器日志的路径。

-B
--backup-path=PATH

备份数据的存储路径。

-c
--check

检查。

-v
--verbose

显示详细信息。

-P
--progress

显示已处理文件的进度。

--running-mode = standard | compatible | mysql

运行模式，默认模式是standard。standard表示标准模式，compatible表示兼容模式，mysql表示mysql模式。

1.19.3.2. 备份选项

-b
--backup-mode=MODE

全量、增量、差异和归档备份，可选参数：full、 difference、 incremental和archive。

-s
--with-serverlog

备份服务器日志文件。

-Z
--compress-data

使用zlib压缩数据备份。

-C
--smooth-checkpoint

备份前进行平滑检查点。

-F
--full-backup-on-error

切换到完全备份模式。

> **注意**
>
> 此选项仅用于“--backup-mode=incremental或archive”。

--keep-data-generations=NUM

保留NUM代完整数据备份。

--keep-data-days=NUM

保持足够的数据备份以恢复到N天前。

--keep-arclog-files=NUM

保留NUM个已归档的WAL日志。

--keep-arclog-days=DAY

保持存档的WAL在DAY天内修改。

--keep-srvlog-files=NUM

保留NUM个服务器日志。

--keep-srvlog-days=DAY

保留在DAY天内修改的服务器日志。

--standby-host=HOSTNAME

从待机状态进行备份时备用主机。

--standby-port=PORT

从待机状态进行备份时的备用端口。

1.19.3.3. 恢复选项

--recovery-target-time

恢复进行的时间戳。

--recovery-target-xid

恢复继续进行的事务ID。

--recovery-target-inclusive

是否在目标恢复完成之后就停止。

--recovery-target-timeline

恢复到特定的时间。

--hard-copy

复制archivelog而不是符号链接。

1.19.3.4. 目录选项

-a
--show-all

显示已删除的备份。

1.19.3.5. 删除选项

-f
--force

强制删除比规定日期更早的备份。

1.19.3.6. 连接选项

-d
--dbname=DBNAME

连接指定数据库。

-h
--host=HOSTNAME

数据库主机。

-P
--port=PORT

数据库端口。

-U
--username=USERNAME

数据库用户名。

-w
--no-password

不提示密码。

-W
--password

强制提示密码。

1.19.3.7. 通用选项

-q
--quiet

不显示任何info和debug信息。

--debug

查看debug信息。

--help

查看帮助信息。

--version

查看版本信息。

1.19.4. 环境变量

BACKUP_PATH

备份路径。

例如：

export BACKUP_PATH=/home/uxdb/backup

来设置备份路径。

> **注意**
>
> 如果不设置环境变量，则涉及路径的命令都需要添加 “-B backup_path” 指定备份路径。

1.19.5. 操作

1.19.5.1. 初始化操作

1. 创建归档目录。

   mkdir /home/uxdb/archivedir

2. 修改目标数据库配置文件。

   vi uxsinodb.conf

```
archive_mode = on       # enables archiving; off, on, or always
                # (change requires restart)
archive_command = 'test ! -f /home/uxdb/archivedir/%f && cp %p /home/uxdb/archivedir/%f'

archive_timeout = 1800      # force a logfile segment switch after this
                # number of seconds; 0 disables
```

3. 重启数据库。

4. 初始化备份目录。

   ux_rman -D PATH init

   PATH是备份目标（实例）的路径，例如：/home/uxdb/uxdbinstall/dbsql/bin/test。

   初始化成功后，如图所示：

```
[uxdb@localhost bin]$ ux_rman -D /home/uxdb/uxdbinstall/dbsql/bin/debug init
INFO: ARCLOG_PATH is set to '/home/uxdb/archivedir'
INFO: SRVLOG_PATH is set to '/home/uxdb/uxdbinstall/dbsql/bin/debug/ux_log'
```

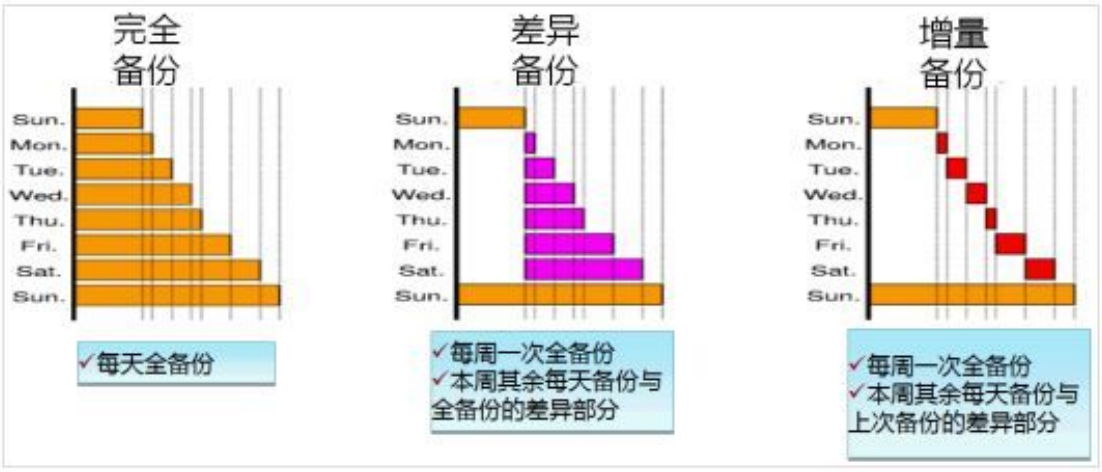
   初始化成功后，进入备份目录可以看到ux_rman配置文件ux_rman.ini ，初始配置文件内容如图所示：

```
ARCLOG_PATH='/home/uxdb/archivedir'
SRVLOG_PATH='/home/uxdb/uxdbinstall/dbsql/bin/debug/ux_log'
```

可以在此文件中添加其他配置参数。

```
COMPRESS_DATA = YES
KEEP_ARCLOG_FILES = 10
KEEP_ARCLOG_DAYS = 10
KEEP_DATA_GENERATIONS = 3
KEEP_DATA_DAYS = 120
KEEP_SRVLOG_FILES = 10
KEEP_SRVLOG_DAYS = 10
```

1.19.5.2. 备份操作



下面的PATH指的是备份目标（实例）的路径，例如：/home/uxdb/uxdbinstall/dbsql/bin/test。

- 全量备份

  ux_rman -D PATH backup --backup-mode=full --progress

  备份成功后，提示信息：

```
[uxdb@localhost bin]$ ux_rman -D /home/uxdb/uxd
Password:
INFO: copying database files
Processed 1044 of 1044 files, skipped 0
INFO: copying archived WAL files
Processed 3 of 3 files, skipped 0
INFO: backup complete
```

> **注意**
>
> 再次全备时，若备份目录中存在已校验的早期备份，则再次全备会跳过已备份的WAL日志以减少I/O操作。所以建议再次全备时手动将已存在的早期备份转移到其他目录。

- 增量备份

  ux_rman -D PATH backup --backup-mode=incremental --progress

  备份成功后，提示信息：

```
[uxdb@localhost bin]$ ux_rman -D /home/uxdb/debug backup --backup-mode=incremental
Password:
INFO: copying database files
Processed 1047 of 1047 files, skipped 991
INFO: copying archived WAL files
Processed 8 of 8 files, skipped 3
INFO: backup complete
INFO: Please execute 'ux_rman validate' to verify the files are correctly copied.
```

- 归档备份(增量模式)

  ux_rman -D PATH backup --backup-mode=archive --progress

  备份成功后，提示信息：

```
[uxdb@localhost bin]$ ux_rman -D /home/uxdb/debug backup --backup-mode=archive --pr
Password:
INFO: copying archived WAL files
Processed 11 of 11 files, skipped 8
INFO: backup complete
INFO: Please execute 'ux_rman validate' to verify the files are correctly copied.
```

- 差异备份

  ux_rman -D PATH backup --backup-mode=difference --progress

  备份成功后，提示信息：

```
[uxdb@localhost bin]$ ./ux_rman -D /home/uxdb/ux01 backup --backup-mode=difference --progress
Password:
INFO: copying database files
Processed 1048 of 1048 files, skipped 1012
INFO: copying archived WAL files
Processed 31 of 31 files, skipped 6
INFO: backup complete
INFO: Please execute 'ux_rman validate' to verify the files are correctly copied.
```

1.19.5.3. 校验操作

每次备份完成后必须进行校验操作，否则此次备份不可以作为下一次备份的基础，且不可用于还原操作。

ux_rman validate

```
[uxdb@localhost bin]$ ux_rman validate
INFO: validate: "2019-09-25 16:27:38" archive log files by CRC
INFO: backup "2019-09-25 16:27:38" is valid
```

1.19.5.4. 恢复操作

恢复时可选择原地恢复或异地恢复，原地恢复时需要停止数据库服务，以免发生意外造成数据丢失。

ux_rman restore -D PATH --recovery-target-time TIME

恢复到指定时间点TIME之前，原地恢复时，PATH为原库路径；异地恢复时，PATH为备库路径。

```
[uxdb@localhost bin]$ ux_rman restore -D /home/uxdb/debug --recovery-target-time "2019-09-25 16:25:03"
INFO: the recovery target timeline ID is not given
INFO: use timeline ID of current database cluster as recovery target: 1
INFO: calculating timeline branches to be used to recovery target point
INFO: searching latest full backup which can be used as restore start point
INFO: found the full backup can be used as base in recovery: "2019-09-25 16:18:14"
INFO: copying online WAL files and server log files
INFO: clearing restore destination
INFO: validate: "2019-09-25 16:18:14" backup and archive log files by SIZE
INFO: backup "2019-09-25 16:18:14" is valid
INFO: restoring database files from the full mode backup "2019-09-25 16:18:14"
INFO: searching incremental backup to be restored
INFO: searching backup which contained archived WAL files to be restored
INFO: backup "2019-09-25 16:18:14" is valid
INFO: restoring WAL files from backup "2019-09-25 16:18:14"
INFO: backup "2019-09-25 16:25:03" is valid
INFO: restoring WAL files from backup "2019-09-25 16:25:03"
INFO: validate: "2019-09-25 16:27:38" archive log files by SIZE
INFO: backup "2019-09-25 16:27:38" is valid
INFO: restoring WAL files from backup "2019-09-25 16:27:38"
INFO: restoring online WAL files and server log files
INFO: generating recovery.conf
INFO: restore complete
HINT: Recovery will start automatically when the PostgreSQL server is started.
```

1.20. uxsql

uxsql — UXDB的交互式终端

1.20.1. 用法

uxsql [*option*...] [*dbname* [*username*]]

1.20.2. 描述

uxsql是一个UXDB的基于终端的前端。它让你能交互式地输入查询，把它们发送给UXDB，并且查看查询结果。或者，输入可以来自于一个文件或者命令行参数。此外，uxsql还提供一些元命令和多种类似shell的特性来为编写脚本和自动化多种任务提供便利。

1.20.3. 选项

-a
--echo-all

把所有非空输入行按照它们被读入的形式打印到标准输出（不适用于交互式行读取）。这等效于把变量ECHO设置为all。

-A
--no-align

切换到非对齐输出模式（默认输出模式是对齐的）。这等效于\pset format unaligned。

-b
--echo-errors

把失败的SQL命令打印到标准错误输出。这等效于把变量ECHO设置为errors。

-c *command*
--command=*command*

指定uxsql执行一个给定的命令字符串*command*。这个选项可以重复多次并且以任何顺序与-f选项组合在一起。当-c或者-f被指定时，uxsql不会从标准输入读取命令，直到它处理完序列中所有的-c和-f选项之后终止。

*command*必须是一个服务器完全可解析的命令字符串（即不包含uxsql相关的特性）或者单个反斜线命令。因此不能在一个-c选项中混合SQL和uxsql元命令。要那样做，可以使用多个-c选项或者把字符串用管道输送到uxsql中，例如：

```
uxsql -c '\x' -c 'SELECT * FROM foo;'
```

或者

```
echo '\x \\ SELECT * FROM foo;' | uxsql
```

（\\是分隔符元命令）。

每一个被传递给-c的SQL命令字符串会被当做一个单独的查询请求发送给服务器。因此，即便该字符串包括多个SQL命令，服务器也会把它当做一个事务来执行，除非在该字符串中有显式的**BEGIN/COMMIT**命令把它划分成多个事务。此外，uxsql只会打印出该字符串中最后一个SQL命令的结果。这和从文件中读取同一字符串或者把同一字符串传给uxsql的标准输出时的行为不同，这两种情况下uxsql会独立地发送每一个SQL命令。

由于这种行为，把多于一个SQL命令放在-c字符串中通常会得到意料之外的结果。最好使用多个-c命令或者把多个命令输送给uxsql的标准输入，按照上文所说的使用echo或者通过一个shell，例如：

```
uxsql <<EOF
\x
SELECT * FROM foo;
EOF
```

--csv

切换到CSV（逗号分隔值）输出模式。 这相当于\pset format csv。

-d *dbname*
--dbname=*dbname*

指定要连接的数据库的名称。这等效于指定*dbname*为命令行上的第一个非选项参数。

如果这个参数包含一个=符号或者以一个合法的URI前缀（UXDB://或者uxdb://）开始，它会被当作一个*conninfo*字符串。

-e
--echo-queries

也把发送到服务器的所有SQL命令复制到标准输出。这等效于把变量ECHO设置为queries。

-E
--echo-hidden

回显\d以及其他反斜线命令生成的实际查询。可以用它来学习uxsql的内部操作。这等效于把变量ECHO_HIDDEN设置为on。

-f *filename*
--file=*filename*

从文件*filename*而不是标准输入中读取命令。这个选项可以被重复多次，也可以以任意顺序与-c选项组合。当-c或者-f被指定时，uxsql不会从标准输入读取命令，直到它处理完序列中所有的-c和-f选项之后终止。除此以外，这个选项很大程度上等价于元命令\i。

如果*filename*是-（连字符），那么会读取标准输入直到遇见一个EOF指示或者\q元命令。这种方式可以用把自多个文件的输入组合成一种交互式输入。不过注意在这种情况下不会使用Readline（类似指定了-n的情况）。

使用这个选项与uxsql < *filename*有细微的不同。通常，两种形式都可以做到我们所期望的，但是使用-f启用了一些好的特性，例如带有行号的错误消息。有机会可以降低启动开销。在另一方面，使用shell输入重定向的变体（理论上）保证会得到与手工输入时相同的输出。

-F *separator*
--field-separator=*separator*

使用*separator*作为非对齐输出的域分隔符。这等效于\pset fieldsep或者\f。

-h *hostname*
--host=*hostname*

指定运行服务器的机器的主机名。如果这个值由一个斜线开始，它会被用作 Unix 域套接字的目录。

-H
--html

打开HTML表格输出。这等效于\pset format html或者\H命令。

-l
--list

列出所有可用的数据库，然后退出。其他非连接选项会被忽略。这与元命令\list类似。

在使用这个选项时，uxsql将连接到数据库uxdb，除非在命令行上提及一个不同的数据（选项-d或非选项参数，可能是通过一个服务项，但不能通过一个环境变量）。

-L *filename*
--log-file=*filename*

除了把所有查询输出写到普通输出目标之外，还写到文件*filename*中。

-n
--no-readline

不使用Readline做行编辑并且不使用命令历史。在剪切和粘贴时，关掉Tab展开会有所帮助。

-o *filename*
--output=*filename*

把所有查询输出放到文件*filename*中。这等效于命令\o。

-p *port*
--port=*port*

指定服务器用于监听连接的 TCP 端口或者本地 Unix 域套接字文件扩展。默认是UXPORT环境变量的值，如果没有设置，则默认为编译时指定的端口号（通常是5432）。

-P *assignment*
--pset=*assignment*

以\pset的形式指定打印选项。注意，这里你必须用一个等号而不是空格来分隔名称和值。例如，要设置输出格式为LaTeX，应该写成-P format=latex。

-q
--quiet

不显示进度信息。默认情况下，它会打印欢迎消息以及多种输出。如果使用了这个选项，都不会输出。在使用-c选项时，配合这个选项很有用。这等效于设置变量QUIET为on。

--running-mode = standard | compatible | mysql

运行模式，默认模式是standard。standard表示标准模式，compatible表示兼容模式，mysql表示mysql模式。

-R *separator*
--record-separator=*separator*

把*separator*用作非对齐输出的记录分隔符。这等效于\pset recordsep命令。

-s
--single-step

运行在单步模式中。这意味着在每个命令被发送给服务器之前都会提示用户一个可以取消执行的选项。使用这个选项可以调试脚本。

-S
--single-line

运行在单行模式中，新行会终止一个SQL命令，就像分号的作用一样。

> **注意**
>
> 这种模式被提供给那些坚持使用它的用户，但是并不一定要使用它。特别地，如果在一行中混合了SQL和元命令，那对于没有经的用户来说，它们的执行顺序可能不总是那么清晰。

-t
--tuples-only

关闭打印列名和结果行计数页脚等。这等效于\t或者\pset tuples_only命令。

-T *table_options*
--table-attr=*table_options*

指定要替换HTML table标签的选项。

-U *username*
--username=*username*

作为用户*username*而不是默认用户连接到数据库。

-v *assignment*
--set=*assignment*
--variable=*assignment*

执行一次变量赋值，和\set元命令相似。注意你必须在命令行上用等号分隔名字和值。要重置一个变量，去掉等号就行。要把一个变量置为空值，使用等号但是去掉值。这些分配是在命令行处理期间完成的，因此反映连接状态的变量稍后将被覆盖。

-V
--version

打印uxsql版本并且退出。

-w
--no-password

永远不要发出密码提示。如果服务器需要密码验证，而通过其他方式(如.pgpass文件)无法获得密码，则连接尝试将失败。此选项在没有用户可以输入密码的批处理作业和脚本中很有用。

> **注意**
>
> 注意这个选项将对整个会话保持设置，并且因此它会影响元命令\connect的使用，就像初始的连接尝试那样。

-W
--password

强制uxsql在连接到一个数据库之前提示输入密码。

这个选项不是必需的，因为如果服务器要求输入密码，uxsql将自动提示输入密码。但是，uxsql将浪费一次连接尝试来发现服务器想要一个密码。在某些情况下建议用-W来避免额外的连接尝试。

> **注意**
>
> 注意这个选项将对整个会话保持设置，并且因此它会影响元命令\connect的使用，就像初始的连接尝试那样。

> 用户输入的密码会在数据传输前以默认加密算法MD5进行加密，加密后与用户名（明文形式）一起传输到服务器端进行身份鉴别。如果用户想修改默认密码加密算法，只需要修改ux_hba.conf中的METHOD列认证方式即可。

-x
--expanded

打开扩展表格式模式。这等效于\x或者\pset expanded命令。

-X
--no-uxsqlrc

不读取启动文件（要么是系统范围的uxsqlrc文件，要么是用户的~/.uxsqlrc文件）。

-Y *password*
--uxsql-password=password

指定连接数据库时要使用的密码。

-z
--field-separator-zero

设置非对齐输出的域分隔符为零字节。这等效于\pset fieldsep_zero。

-0
--record-separator-zero

设置非对齐输出的记录分隔符为零字节。例如，这对与xargs -0配合有关。这等效于\pset recordsep_zero。

-1
--single-transaction

这个选项只能被用于与一个或者多个-c或者-f选项组合。它会让uxsql在第一个-c或者-f选项之前发出一条BEGIN命令并且在最后一个-c/或者-f选项之后发出一条COMMIT命令，这样就把所有的命令都包裹在一个事务中。这个选项可以保证要么所有的命令都成功地完成，要么不应用任何更改。

如果命令本身包含BEGIN、COMMIT或者ROLLBACK，这个选项将不会得到想要的效果。还有，如果其中存在命令不能在一个事务块中执行，指定这个选项将导致整个事务失败。

-?
--help[=*topic*]

显示有关uxsql的帮助并且退出。可选的*topic*参数（默认为options）选择要解释uxsql的哪一部分：commands描述uxsql的反斜线命令；options描述可以被传递给uxsql的命令行选项；而variables则显示有关uxsql配置变量的帮助。

1.20.4. 退出状态

如果uxsql正常完成，它会向shell返回0。如果它自身发生一个致命错误（例如内存用完、找不到文件），它会返回1。如果到服务器的连接出问题并且事务不是交互式的，它会返回2。如果在脚本中发生错误，它会返回3并且变量ON_ERROR_STOP会被设置。

1.20.5. 功能

1.20.5.1. 连接到数据库

uxsql是一个常规UXDB客户端应用。为了连接到数据库，你需要知道你的目标数据库的名称、主机名和该服务器的端口号，还有要作为哪个用户名连接。可以通过命令行选项告知uxsql这些参数，分别是-d、-h、-p以及-U。如果发现一个参数不属于任何选项，它将被解释为数据库名称（如果已经给出数据库名称，就解释为用户名）。并非所有这些选项都是必需的，它们都有可用的默认值。如果省略主机名，uxsql将通过一个 Unix 域套接字连接到本地主机上的服务器，或者通过 TCP/IP 连接到没有 Unix 域套接字的主机上的localhost。默认端口号则在编译时决定。由于数据库服务器使用相同的默认值，大多数情况下你将不必指定端口。默认的用户名是你的操作系统用户名，它也会是默认的数据库名。注意你不一定能连接到任意用户名下的任何数据库。你的数据库管理员应该已经告知过你有关你的访问权限。

当默认值不是很符合实际时，可以把环境变量UXDATABASE、UXHOST、UXPORT以及UXUSER设置为适当的值，这样也能节省一些键盘输入。用~/.uxpass文件来省略输入密码也很方便。

另一种指定连接参数的方法是用一个*conninfo*字符串或者一个URI，它可以被用来替代数据库名。这种机制可以让我们对连接具有很广的控制权。例如：

```
$ uxsql "service=myservice sslmode=require"
$ uxsql uxdb://dbmaster:5433/mydb?sslmode=require
```

用这种方式，你也可以把LDAP用于连接查找。

如果由于一些原因（例如权限不足、服务器没有在目标主机上运行等）导致连接无法建立，uxsql将返回一个错误并且终止。

如果标准输入和标准输出都是一个终端，那么uxsql会把客户端编码设置成“auto”，这会使uxsql从区域设置（Unix 系统上的LC_CTYPE环境变量）中检测合适的客户端编码。如果这样不起作用，可以使用环境变量UXCLIENTENCODING覆盖客户端编码。

1.20.5.2. 输入SQL命令

在正常操作时，uxsql会提供一个提示符，该提示符是uxsql当前连接到的数据库名称后面跟上字符串=#。例如：

```
$ uxsql testdb
uxsql (10.0)
Type "help" for help.

testdb=#
```

在提示符下，用户可以输入SQL命令。正常情况下，当碰到一个表示命令终结的分号时，输入的行会被发送给服务器。一行的结束并不表示命令的完结。因此，为了清晰，可以把命令散布在多个行上。如果命令被发送并且执行而不产生错误，该命令的结果将会显示在屏幕上。

如果不可信用户对还没有采用安全方案使用模式的一个而数据库拥有访问，通过从search_path移除公共可写的方案来开始你的会话。可以在连接字符串中加入options=-csearch_path=或者在其他SQL命令之前发出SELECT ux_catalog.set_config('search_path', '', false)。这种考虑并非专门针对uxsql，它适用于每一种执行任意SQL命令的接口。

只要执行命令，uxsql还会测试LISTEN和NOTIFY产生的异步通知。

C风格的注释块会被传给服务器处理并且移除，uxsql会自己移除掉SQL标准的注释。

1.20.5.3. 输入元命令

输入到uxsql中的任何以未加引用的反斜线开始的东西都是一个uxsql元命令，它们由uxsql自行处理。这些命令让uxsql对管理和编写脚本更有用。元命令常常被称作斜线或者反斜线命令。

uxsql命令的格式是用反斜线后面直接跟上一个命令动词，然后是一些参数。参数与命令动词和其他参数之间用任意多个空白字符分隔开。

要在一个参数中包括空白，可以将它加上单引号。要在一个参数中包括一个单引号，则需要在文本中写上两个单引号。任何包含在单引号中的东西都服从与C语言中\n（换行）、\t（制表符）、\b（退格）、\r（回车）、\f（换页）、*digits*（10 进制）以及\x*digits*（16 进制）类似的替换规则。单引号内文本中的其他任何字符前面的反斜线都没有实际意义（会被忽略）。

如果在一个参数中出现一个未加引号的冒号（:）后面跟着一个uxsql变量名，如SQL中插入变量中所述，它会被该变量的值所替换。形如:'*variable_name*'和:"*variable_name*"也有同样的效果。:{?*variable_name*}语法允许测试一个变量是否被定义。它会被TRUE或FALSE替换。用一个反斜线转义该冒号可以防止它被替换。

在一个参数中，包含在反引号（`）中的文本会被当做一个传递给shell的命令行。该命令的输出（移除任何拖尾的新行）会替换反引号文本。在封闭在反引号的文本，不会有特别的引号或者其他处理发生，:*variable_name*的出现除外，其中*variable_name*是一个会被其值替换的uxsql变量名。此外，Also, appearances of :'*variable_name*'的出现会被替换为该变量的值，而值会被适当地加以引用以变成一个单一shell命令参数（后一种形式几乎总是优先，除非你非常确定变量中有什么）。因为回车和换行字符在所有的平台上都不能被安全地引用，:'*variable_name*'形式会打印一个错误消息并且在这类字符出现在值中时不替换该变量值。

有些命令把SQL标识符（例如一个表名）当作参数。这些参数遵循SQL的语法规则：无引号的字母被强制变为小写，而双引号（"）可以保护字母避免大小写转换并且允许在标识符中包含空白。在双引号内，成对的双引号会被缩减为结果名称中的单个双引号。例如，FOO"BAR"BAZ会被解释成fooBARbaz，而"A weird"" name"会变成A weird" name。

对参数的解析会在行尾或者碰到另一个未加引号的反斜线时停止。一个未加引号的反斜线被当做新元命令的开始。特殊的序列\\（两个反斜线）表示参数结束并且应继续解析SQL命令（如果还有）。使用这种方法，SQL命令和uxsql命令可以被自由地混合在一行中。但是无论在何种情况中，元命令的参数都无法跨越一行。

很多元命令作用在当前查询缓冲区上。这就是一个缓冲区而已，它保存任何已经被输入但是还没有发送到服务器执行的SQL命令文本。这将包括之前输入的行以及在该元命令同一行上出现在前面的任何文本。

可以使用下列元命令：

\a

如果当前的表输出格式是非对齐的，则切换成对齐格式。如果不是非对齐格式，则设置成非对齐格式。保留这个命令是为了向后兼容性。

\c or \connect [-reuse-previous=*on*|*off*] [*dbname* [*username*] [*host*] [*port*] | *conninfo*]

与一台UXDB服务器建立一个新连接。可以使用位置语法指定要使用的连接参数，或者使用*conninfo*连接串。

在省略了数据库名、用户、主机或者端口的命令中，新的连接将会重用之前一个连接的值。默认情况下，前一个连接的值将会被重用，除非给出了一个*conninfo*串。给出第一个参数-reuse-previous=on或者-reuse-previous=off可以覆盖默认行为。当这个命令既没有指定一个参数也没有重用它时，将使用libuxsql的默认值。把*dbname*、*username*、*host*或者*port*中的任何一个指定为-等价于省略该参数。

如果新连接成功地被建立，之前的连接会被关闭。如果连接尝试失败（错误的用户名、访问被拒绝等），只有在uxsql处于交互模式的情况下才会保留之前的连接。当执行一个非交互式脚本时出现连接尝试失败，处理将被立即停止，并且报出一个错误。这种区别一方面可以帮助用户发现打字错误，另一方面也可以作为一种安全机制防止脚本在错误的数据库上执行动作。

例子：

```
=> \c mydb myuser host.dom 6432
=> \c service=foo
=> \c "host=localhost port=5432 dbname=mydb connect_timeout=10 sslmode=disable"
=> \c UXDB://tom@localhost/mydb?application_name=myapp
```

\C [*title*]

设置查询结果的任何表的标题，或者重置这类标题。这个命令等效于\pset title *title*（这个命令的名称来自于“caption”，因为它之前只被用来在HTML表格中设置标题）。

\cd [*directory*]

把当前工作目录改为*directory*。如果不带参数，则切换到当前用户的主目录。

> **提示**
>
> 要打印当前的工作目录，可以使用\! pwd。

\conninfo

输出有关当前数据库连接的信息。

\copy { *table* [(*column_list*)] | (*query*) } { from | to } { *'filename'* | program *'command'* | stdin | stdout | pstdin | pstdout } [[with] (*option* [, ...])]

执行一次前端拷贝。这个操作会运行一个SQL **COPY**命令，不过不是服务器读取或者写入指定的文件，而是由uxsql读写文件并且把数据从本地文件系统导向服务器。这意味着文件的可访问性和权限是本地用户的而非服务器上的，并且不需要SQL超级用户特权。

当program被指定时，*command*被uxsql执行并且传给*command*的数据或者从*command*传出的数据会在服务器和客户端之间流动。同样地，执行特权是本地用户的而非服务器上的，并且不需要SQL超级用户特权。

对于\copy ... from stdin，数据行从发出该命令的同一来源读取，一直到读到\.或者数据流到达EOF。这个选项可以用来填充内嵌在一个SQL脚本文件中的表。对于\copy ... to stdout，输出被发送到与uxsql命令输出相同的位置，并且COPY *count*命令的状态不会被打印（因为它会被一个数据行搞乱）。要读/写uxsql的标准输入或者输出而不管当前命令的来源或者\o选项，可以写from pstdin或者to pstdout。

这个命令的语法和SQL COPY命令类似。所有除开数据来源/目的地的选项都和COPY指定的一样。因此，\copy元命令由特殊的解析规则。与大部分其他元命令不同，该行的所有剩余部分总是会被当做\copy的参数，并且在参数中不会执行变量篡改以及反引号展开。

> **提示**
>
> 获得与\copy ... to相同结果的另一种方法是使用SQL COPY ... TO STDOUT命令并使用\g *filename*或\g | *program* 终止它。 \copy不同，此方法允许命令跨越多行；此外，可以使用变量插值和反引号扩展。
>
> 这个操作不如带有文件或程序数据源或目标的SQL COPY命令有效， 因为所有数据都必须通过客户端/服务器连接。 对于大量数据，SQL命令可能更可取。

\copyright

显示UXDB的版权以及发布条款。

\crosstabview [*colV* [*colH* [*colD* [*sortcolH*]]]]

执行当前的查询缓冲区（像\g那样）并且在一个交叉表格子中显示结果。该查询必须返回至少三列。由*colV*标识的输出列会成为垂直页眉并且*colH*所标识的输出列会成为水平页眉。*colD*标识显示在格子中的输出列。*sortcolH*标识用于水平页眉的可选的排序列。

每一个列说明可以是一个列编号（从1开始）或者一个列名。常用的SQL大小写折叠和引用规则适用于列名。如果省略，*colV*被当做列1并且*colH*被当做列2。*colH*必须和*colV*不同。如果没有指定*colD*，那么在查询结果中必须正好有三列，并且*colV*和*colH*之外的那一列会被当做*colD*。

垂直页眉显示为最左边的列，它包含列*colV*中找到的值，值的顺序和查询结果中的顺序相同，但是重复值会被移除。

水平页眉显示为第一行，它包含列*colH*中找到的值，其中的重复值被移除。默认情况下，这些值会以查询结果中相同的顺序出现。但是如果给出了可选的*sortcolH*参数，它标识一个值必须为整数编号的列，并且来自*colH*的值将会根据相应的*sortcolH*值排序后出现在水平页眉中。

在交叉表格子中，对于*colH*的每一个可区分的值x以及*colV*的每一个可区分的值y，位于交叉点(x,y)的单元包含*colH*值为x且*colV*值为y的查询结果行中colD列的值。如果没有这样的行，则该单元为空。如果有多个这样的行，则会报告一个错误。

\d[S+] [*pattern*]

对于每一个匹配*pattern*的关系（表、视图、物化视图、索引、序列或者外部表）或者组合类型，显示所有的列、它们的类型、表空间（如果非默认表空间）以及任何诸如NOT NULL或者默认值的特殊属性。相关的索引、约束、规则以及触发器也会被显示。对于外部表，还会显示相关的外部服务器（下文的模式（Pattern）中定义了“匹配模式”）。

对于某些类型的关系，\d会为每一列显示额外的信息：对于序列会显示列值，对于索引显示被索引的表达式，对于外部表显示外部数据包装器选项。

命令形式\d+是一样的，不过会显示更多信息：与该表的列相关的任何注释，表中是否存在OID，如果关系是视图则显示视图定义，非默认的replica identity设置。

默认情况下只会显示用户创建的对象，提供一个模式或者S修饰符可以把系统对象包括在内。

> **注意**
>
> 如果使用\d但不带有*pattern*参数，它等价于\dtvmsE，后者将显示所有可见的表、视图、物化视图、序列和外部表的列表。这纯粹是一种便利措施。

\da[S] [*pattern*]

列出聚集函数，以及它们的返回类型和它们所操作的数据类型。如果指定了*pattern*，只显示名称匹配该模式的聚集。默认情况下只会显示用户创建的对象，提供一个模式或者S修饰符可以把系统对象包括在内。

\dA[+] [*pattern*]

列出访问方法。如果指定了*pattern*，只显示名称匹配该模式的访问方法。如果在命令名称后面追加+，则与访问方法相关的处理器函数和描述也会和访问方法本身一起被列出。

\db[+] [*pattern*]

列出表空间。如果指定了*pattern*，只显示名称匹配该模式的表空间。如果在命令名称后面追加+，则与表空间相关的选项、磁盘上的尺寸、权限以及描述也会和表空间本身一起被列出。

\dc[S+] [*pattern*]

列出字符集编码之间的转换。如果指定了*pattern*，只列出名称匹配该模式的转换。默认情况下只会显示用户创建的对象，提供一个模式或者S修饰符可以把系统对象包括在内。如果在命令名称后面追加+，则每一个对象相关的描述也会被列出。

\dC[+] [*pattern*]

列出类型转换。如果指定了*pattern*，只列出源类型和目标类型匹配该模式的转换。如果在命令名称后面追加+，则每一个对象相关的描述也会被列出。

\dd[S] [*pattern*]

显示约束、操作符类、操作符族、规则以及触发器类型对象的描述。所有其他注释可以通过那些对象类型相应的反斜线命令查看。

\dd显示匹配*pattern*的对象的描述，如果没有给出参数则显示合适类型的可见对象的描述。但是在任一种情况下都只列出具有描述的对象。默认情况下只会显示用户创建的对象，提供一个模式或者S修饰符可以把系统对象包括在内。

对象的描述可以用**COMMENT**SQL命令创建。

\dD[S+] [*pattern*]

列出域。如果指定了*pattern*，只有名称匹配该模式的域会被显示。默认情况下，只有用户创建的对象会被显示，可以提供一个模式或者S修饰符以包括系统对象。如果+被追加到命令名称上，每一个被列出的对象会带有其相关的权限和描述。

\ddp [*pattern*]

列出默认的访问特权设置。对那些默认特权设置已经被改变得与内建默认值不同的角色（以及模式，如果适用），为每一个角色（以及模式）显示一项。如果指定了*pattern*，只列出角色名称或者模式名称匹配该模式的项。

`sql-alterdefaultprivileges`命令被用来设置默认访问特权。在GRANT中解释了显示的特权的含义。

\dE[S+] [*pattern*]
\di[S+] [*pattern*]
\dm[S+] [*pattern*]
\ds[S+] [*pattern*]
\dt[S+] [*pattern*]
\dv[S+] [*pattern*]

在这一组命令中，字母E、i、m、s、t和v分别对应着外部表、索引、物化视图、序列、表和视图。你可以以任何顺序指定这些字母中的任意一个或者多个，这样可以得到这些类型的对象的列表。例如，\dit会列出索引和表。如果在命令名称后面追加+，则每一个对象的物理尺寸以及相关的描述也会被列出。如果指定了*pattern*，只列出名称匹配该模式的对象。默认情况下只会显示用户创建的对象，提供一个模式或者S修饰符可以把系统对象包括在内。

\des[+] [*pattern*]

列出外部服务器。如果指定了*pattern*，只列出名称匹配该模式的那些服务器。如果使用了\des+形式，将显示每个服务器的完整描述，包括该服务器的ACL、类型、版本、选项和描述。

\det[+] [*pattern*]

列出外部表（助记：“外部表”）。如果指定了*pattern*，只列出表名称或者模式名称匹配该模式的项。如果使用了\det+选项，一般选项和外部表描述也会被显示。

\deu[+] [*pattern*]

列出用户映射。如果指定了*pattern*，只列出用户名匹配该模式的那些映射。如果使用了\deu+形式，有关每个映射的额外信息也会被显示。

> **注意**
>
> \deu+可能也会显示远程用户的用户名和密码，所以要注意不要把它们泄露出去。

\dew[+] [*pattern*]

列出外部数据封装。如果指定了*pattern*，只列出名称匹配该模式的那些外部数据包装器。如果使用了\dew+形式，外部数据包装器的ACL、选项和描述也会被显示。

\df[anptwS+] [*pattern*]

列出函数，以及它们的结果数据类型、参数数据类型和函数类型，函数类型被分为“agg”（聚集）、“normal”、“procedure”、“trigger”以及“window”。如果要只显示指定类型的函数，可以在该命令上增加相应的字母a、n、p、t或者w。如果指定了*pattern*，只显示名称匹配该模式的函数。默认情况下只会显示用户创建的对象，提供一个

模式或者S修饰符可以把系统对象包括在内。如果使用了\df+形式，则有关每个函数的额外信息也会被显示，包括易失性、并行安全性、拥有者、安全性分类、访问特权、语言、源代码和描述。

> **提示**
>
> 如果要查找接收指定数据类型参数或者返回指定类型值的函数，可以使用分页器的搜索能力来滚动显示\df输出。

\dF[+] [*pattern*]

列出文本搜索配置。如果指定了*pattern*，只显示名称匹配该模式的配置。如果使用了\dF+形式，每种配置的完整描述也会被显示，包括底层的文本搜索解析器和用于每一种解析器记号类型的字典列表。

\dFd[+] [*pattern*]

列出文本搜索字典。如果指定了*pattern*，只显示名称匹配该模式的字典。如果使用了\dFd+形式，有关每一种选中的字典的额外信息也会被显示，包括底层的文本搜索模板和选项值。

\dFp[+] [*pattern*]

列出文本搜索解析器。如果指定了*pattern*，只显示名称匹配该模式的解析器。如果使用了\dFp+形式，每一种解析器的完整描述也会被显示，包括底层的函数和可识别的记号类型列表。

\dFt[+] [*pattern*]

列出文本搜索模板。如果指定了*pattern*，只显示名称匹配该模式的模板。如果使用了\dFt+形式，每一种模板有关的额外信息也会被显示，包括底层的函数名称。

\dg[S+] [*pattern*]

列出数据库角色（因为“用户”和“组”的概念已经被统一成“角色”，这个命令现在等价于\du）。默认情况下只会显示用户创建的角色，提供一个模式或者S修饰符可以把系统角色包括在内。如果指定了*pattern*，只列出名称匹配该模式的那些角色。如果使用了\dg+形式，有关每种角色的额外信息也将被显示，当前这种形式会为角色增加显示注释。

\dl

这是\lo_list的一个别名，它显示大对象的列表。

\dL[S+] [*pattern*]

列出过程语言。如果指定了*pattern*，只列出名称匹配该模式的语言。默认情况下只会显示用户创建的语言，提供一个模式或者S修饰符可以把系统对象包括在内。如果向命令名称追加+，则每一种语言会和它的调用处理器、验证器、访问特权以及它是否为系统对象一起列出。

\dn[S+] [*pattern*]

列出模式（名字空间）。如果指定了*pattern*，只列出名称匹配该模式的模式。默认情况下只会显示用户创建的对象，提供一个模式或者S修饰符可以把系统对象包括在内。如果向命令名称追加+，每个对象会与它相关的权限及描述（如果有）一起被列出。

\do[S+] [*pattern*]

列出操作符及其操作数和结果类型。如果指定了*pattern*，只列出名称匹配该模式的操作符。默认情况下只会显示用户创建的对象，提供一个模式或者S修饰符可以把系统对象包括在内。如果向命令名称追加+，有关每个操作符的额外信息也将被显示，当前只包括底层函数的名称。

\dO[S+] [*pattern*]

列出排序规则。如果指定了*pattern*，只列出名称匹配该模式的排序规则。默认情况下只会显示用户创建的对象，提供一个模式或者S修饰符可以把系统对象包括在内。如果向命令名称追加+，每个排序规则将和它相关的描述（如果有）一起被列出。注意只有可用于当前数据库编码的排序规则会被显示，因此在同一个安装下的不同数据库中执行此命令可能会得到不同的结果。

\dp [*pattern*]

列出表、视图和序列，包括与它们相关的访问特权。如果指定了*pattern*，只列出名称匹配该模式的表、视图以及序列。

GRANT和REVOKE命令被用来设置访问特权。

\dP[itn+] [*pattern*]

列出分区关系。如果指定了*pattern*，则仅列出名称与模式匹配的条目。　　修改符t（表）和i（索引）可以追加到命令中，筛选要列出的关系类型。默认会列出分区表和索引。

如果用了修饰符n（“nested”）或指定了模式，则包括非根分区关系，并显示每个分区关系的父级的列。

如果+被附加到命令名中，那么还会显示每个关系分区的大小总和，以及关系的描述。　　如果n与_相结合，将显示两种大小：一种包含直接连接的叶分区的总大小，另一种显示所有分区的总大小，包括间接附加的子分区。

\drds [*role-pattern* [*database-pattern*]]

列出已定义的配置设置。这些设置可以是针对角色的、针对数据库的或者同时针对两者的。*role-pattern*和*database-pattern*分别被用来选择要列出的角色和数据库。如果省略它们或者指定了*，则会列出所有设置，分别会包括针对角色和针对数据库的设置。

ALTER ROLE以及ALTER DATABASE命令可以用来定义一个角色以及一个数据库的配置设置。

\dRp[+] [*pattern*]

列出复制发布。如果指定有*pattern*，则只列出名称与模式匹配的发布。如果将+附加到命令名称，则也会显示与每个发布相关联的表。

\dRs[+] [*pattern*]

列出复制的订阅。如果指定有*pattern*，只有那些名字匹配该模式的订阅才会被列出。如果+被追加到命令的名称上，订阅的额外属性会被显示。

\dT[S+] [*pattern*]

列出数据类型。如果指定了*pattern*，只列出名称匹配该模式的类型。如果向命令名称追加+，每一种类型、其内部名称和尺寸、允许的值（如果是一种enum类型）以及相关权限会

被一同列出。默认情况下只会显示用户创建的对象，提供一个模式或者S修饰符可以把系统对象包括在内。

\du[S+] [*pattern*]

列出数据库角色（因为“用户”和“组”的概念已经被统一成“角色”，这个命令现在等价于\dg）。默认情况下只会显示用户创建的角色，提供一个模式或者S修饰符可以把系统角色包括在内。如果指定了*pattern*，只列出名称匹配该模式的那些角色。如果使用了\du+形式，有关每一种角色的额外信息也会被显示，当前只会多显示角色的注释。

\dx[+] [*pattern*]

列出已安装的扩展。如果指定了*pattern*，只列出名称匹配该模式的那些扩展。如果使用了\dx+形式，所有属于每个匹配扩展的对象会被列出。

\dy[+] [*pattern*]

列出事件触发器。如果指定了*pattern*，只列出名称匹配该模式的事件触发器。如果在命令名称后面加上+，还会为每个列出的对象显示其相关的描述。

\eor\edit [*filename*] [*line_number*]

如果指定了*filename*，则它是被编辑的文件，在编辑器退出后，该文件的内容会被拷贝到当前查询缓冲区中。如果没有给定*filename*，当前查询缓冲区会被拷贝到一个临时文件中，并且接着以相同的方式编辑。或者，如果当前查询缓冲区为空，则最近被执行的查询会被拷贝到一个临时文件并且以同样的方式编辑。

查询缓冲区的新内容接下来将被重新根据uxsql的规则进行解析， 将整个缓冲区视为一条线。任何完整的查询都会立即执行；也就是说，如果查询缓冲区包含分号或以分号结尾，那么直到该点的所有内容都会被执行。 无论什么都会在查询缓冲区中等待。输入分号或\g发送它，或者\r通过清除查询缓冲区来取消它。将缓冲区视为单行处理主要影响元命令：元行命令之后的任何内容都将作为元命令的参数，即使它跨越多行。（因此你不能用这种方式制作使用元命令的脚本，而是使用\i。）

如果指定了一个行号，uxsql将会把游标（注意不是服务器端的游标）定位到文件或者查询缓冲区的指定行上。注意如果给出了一个全是数字的参数，uxsql就会假定它是行号而不是文件名。

\echo *text* [...]

把参数打印到标准输出，参数之间用一个空格分隔，最后加上一个新行。这可以用来在脚本的输出中间混入信息，例如：

```
=> \echo `date`
2020年 07月 15日 星期三 10:40:52 CST
```

如果第一个参数是一个没有加引号的-n，则不会加上最后的新行。

> **提示**
>
> 如果使用\o命令来重定向查询的输出，你可能希望使用\qecho来取代这个命令。

\ef [*function_description* [*line_number*]]

这个命令会以一个CREATE OR REPLACE FUNCTION或CREATE OR REPLACE PROCEDURE命令的形式取出并且编辑指定函数或过程的定义。编辑的方式与\edit完全相同。在编辑器退出后，更新过的命令将在查询缓冲区中等待，可以输入分号或者\g把它发出，也可以用\r取消。

目标函数可以单独用名称或者用名称和参数（例如foo(integer, text)）来指定。如果有多于一个函数具有同样的名称，则必须给出参数的类型。

如果没有指定函数，将会给出一个空白的CREATE FUNCTION模板来编辑。

如果指定了一个行号，uxsql将把游标定位在该函数体的指定行上（注意函数体通常不是开始于文件的第一行）。

和大部分其他元命令不同，该行的整个剩余部分总是会被当作\ef的参数，并且在参数中不会执行变量篡改以及反引号展开。

\encoding [*encoding*]

设置客户端字符集编码。如果没有参数，这个命令会显示当前的编码。

\errverbose

以最详细的程度重复最近的服务器错误消息，就好像VERBOSITY被设置为verbose且SHOW_CONTEXT被设置为always。

\ev [*view_name* [*line_number*]]

这个命令会以一个CREATE OR REPLACE VIEW的形式取出并且编辑指定函数的定义。编辑的方式与\edit完全相同。在编辑器退出后，更新过的命令将在查询缓冲区中等待，可以输入分号或者\g把它发出，也可以用\r取消之。

如果没有指定函数，将会给出一个空白的CREATE VIEW模板来编辑。

如果指定了一个行号，uxsql将把游标定位在该视图定义的指定行上。

和大部分其他元命令不同，该行的整个剩余部分总是会被当作\ev的参数，并且在参数中不会执行变量篡改以及反引号扩展。

\f [*string*]

设置用于非对齐查询输出的域分隔符。默认值是竖线（|）。它等效于\pset fieldsep。

\g [*filename*]
\g [|*command*]

把当前查询缓冲区发送给服务器执行。如果给出一个参数，查询的输出将被写到提到的文件或者用管道导向给定的shell命令，而不是像往常一样显示它。仅当查询成功返回零个或多个元组时，才会写入文件或命令，查询失败或不是数据返回的SQL命令时不写入。

如果当前查询缓冲区为空，则重新执行最近一次被发送的查询。除了这种行为之外，没有参数的\g实际上等效于一个分号。一个带有参数的\g是\o命令的一种“一次性”替换方案。

如果参数以|开始，则该行的所有剩余部分总是会被当做要执行的*command*，并且在参数中不会执行变量篡改以及反引号展开。该行的剩余部分会被简单地按字面传给shell。

\gdesc

显示当前查询缓冲区的结果的描述（即列名和数据类型）。查询不会被实际执行，不过，如果它含有某种类型的语法错误，该错误将被以通常的方式报出。

如果当前查询缓冲区为空，则会描述最近被发送的查询。

\gexec

把当前查询缓冲区发送到服务器，然后该查询输出（如果有）中的每一行的每一列都当作一个要被执行的SQL语句。例如，要在my_table的每一列上都创建一个索引：

```
=> SELECT format('create index on my_table(%I)', attname)
-> FROM ux_attribute
-> WHERE attrelid = 'my_table'::regclass AND attnum > 0
-> ORDER BY attnum
-> \gexec
CREATE INDEX
CREATE INDEX
CREATE INDEX
CREATE INDEX
```

产生的查询会按照其所在行被返回的顺序执行，如果有多个列，则同一行中按照从左至右的顺序执行。NULL域会被忽略。产生的查询会被原样发送给服务器处理，因此它们即不能是uxsql元命令，也不能包含uxsql变量引用。如果其中任何一个查询失败，剩余查询的执行将会继续，除非设置了ON_ERROR_STOP。每个查询的执行都遵照ECHO的处理（在使用\gexec时，通常建议设置ECHO为all或者queries）。查询日志、单步模式、计时以及其他查询执行特性也适用于每一个生成的查询。

如果当前查询缓冲区为空，则会重新执行最近被发送的查询。

\gset [*prefix*]

把当前查询输入缓冲区发送给服务器并且将查询的输出存储在uxsql变量中（见变量）。被执行的查询必须只返回一行。该行的每一列会被存储到一个单独的变量中，变量和该列的名字一样。例如：

```
=> SELECT 'hello' AS var1, 10 AS var2
-> \gset
=> \echo :var1 :var2
hello 10
```

如果指定了一个*prefix*，那么该字符串会被追加在该查询的输出列名称之前用来创建要使用的变量名：

```
=> SELECT 'hello' AS var1, 10 AS var2
-> \gset result_
=> \echo :result_var1 :result_var2
hello 10
```

如果一个列的结果为NULL，那么对应的变量会被重置而不是被设置。

如果查询失败或者没有返回一行，则不会有任何变量被更改。

如果当前查询缓冲区为空，则重新执行最近被发送的查询。

\gx [*filename*]
\gx [|*command*]

\gx等效于\g，但会为这个查询强制扩展的输出模式。

\h or \help [*command*]

给出指定SQL命令的语法帮助。如果没有指定*command*，则uxsql会列出可以显示语法帮助的所有命令。如果*command*是一个星号（*），则会显示所有SQL命令的语法帮助。

与大部分其他元命令不同，该行的所有剩余部分总是会被当做\help的参数，并且在参数中不会执行变量篡改以及反引号扩展。

> **注意**
>
> 为了简化输入，由几个词构成的命令不需要被加上引号。因此，输入**\help alter table**是可以的。

\H or \html

开启HTML查询输出格式。如果HTML格式已经开启，这会把它切换回默认的对齐文本格式。这个命令是为了兼容性和方便。

\i or \include *filename*

从文件*filename*读取输入并且把它当作从键盘输入的命令来执行。

如果*filename*是-（连字符），那么会一直读取标准输入直到碰到一个EOF指示符或者\q元命令。这可以用来把交互式输入与文件输入混杂。注意只有在最外层激活了readline行为的情况下才将会使用readline行为。

> **注意**
>
> 如果想在屏幕上看到被读入的行，必须把变量ECHO设置成all。

\if *expression*
\elif *expression*
\else
\endif

这组命令实现可嵌套的条件块。条件块必须以一个\if开始并且以一个\endif结束。两者之间可能有任意数量的\elif子句，后面也可能有选择地跟着一个单一的\else子句。一般查询以及其他类型的反斜线命令可以出现在这些命令之间构成条件块。

\if和\elif命令读取它们的参数并且将它们作为布尔表达式进行计算。如果表达式得到真则处理正常继续下去，否则会跳过下面的行直到到达一个匹配的\elif、\else或者\endif。一旦一个\if或者\elif测试成功，同一个块中后面的\elif命令的参数将不会被计算但会被当作为假。\else之后的行仅在前面的\if或\elif都未成功时才进行。

\if或者\elif命令的*expression*参数受到可变插值和反引用扩展的影响，就像任何其他反斜杠命令参数一样。之后，它被评估为开/关选项变量的值。所以有效值是以下之一的任何明

确的不区分大小写的匹配：true、false、1、0、on、off、yes、no。例如，t、T以及tR都将被认为是true。

无法被正确计算为真或假的表达式将产生一个警告并且被当做假。

被跳过的行还是会被正常地解析以标识查询和反斜线命令，但是查询不会被发送到服务器，并且非条件（\if、\elif、\else、\endif）反斜线命令会被忽略。条件命令会被检查以判断嵌套是否有效。被跳过的行中的变量引用不会被扩展，并且也不会执行反引号扩展。

给定条件块中的所有反斜线命令必须出现在相同的源文件中。如果在所有的本地\if块被关闭之前，主输入文件或者一个\include进来的文件上就达到了EOF，则uxsql将产生一个错误。

这里是一个例子：

```
-- 检查数据库中两个单独记录的存在性并且把结果存在单独的uxsql变量中
SELECT
   EXISTS(SELECT 1 FROM customer WHERE customer_id = 123) as is_customer,
   EXISTS(SELECT 1 FROM employee WHERE employee_id = 456) as is_employee
\gset
\if :is_customer
   SELECT * FROM customer WHERE customer_id = 123;
\elif :is_employee
   \echo 'is not a customer but is an employee'
   SELECT * FROM employee WHERE employee_id = 456;
\else
   \if yes
      \echo 'not a customer or employee'
   \else
      \echo 'this will never print'
   \endif
\endif
```

\ir or \include_relative *filename*

\ir命令类似于\i，但是以不同的方式处理相对路径文件名。在交互模式中执行时，这两个命令的行为相同。不过，当被从脚本中调用时，\ir相对于脚本所在的目录而不是根据当前工作目录来解释文件名。

\l[+] or \list[+] [*pattern*]

列出服务器中的数据库并且显示它们的名称、拥有者、字符集编码以及访问特权。如果指定了*pattern*，则只列出名称匹配该模式的数据库。如果向命令名称追加+，则还会显示数据库的尺寸、默认表空间以及描述（尺寸信息只对当前用户能连接的数据库可用）。

\lo_export *loid filename*

从数据库中读取具有OID *loid*的大对象并且将它写入到*filename*。注意这和服务器函数lo_export有微妙的不同，后者会以运行数据库服务器的用户权限来执行并且运行在服务器的文件系统上。

> **提示**
>
> 使用\lo_list可以找出大对象的OID。

\lo_import *filename* [*comment*]

把该文件存储到UXDB大对象。可选地，它可以把给定的注释关联到该对象。例如：

```
foo=> \lo_import '/home/peter/pictures/photo.xcf' 'a picture of me'
lo_import 152801
```

该响应表示该大对象得到的对象ID是　152801，未来可以用这个ID来访问这个新创建的大对象。为了便于阅读，推荐总是给每一个对象都关联人类可读的注释。OID和注释都可以用\lo_list命令查看。

注意这个命令和服务器端的lo_import有微妙的不同，因为它以本地文件系统上的本地用户的身份运行，而不是服务器用户和文件系统。

\lo_list

显示当前存储在数据库中的所有UXDB大对象，同时显示它们的注释。

\lo_unlink *loid*

从数据库中删除OID为*loid*的大对象。

> **提示**
>
> 使用\lo_list可以找出该大对象的OID。

\o or \out [*filename*]
\o or \out [|*command*]

安排把未来的查询结果保存到文件*filename*中或者用管道导向到shell命令*command*。如果没有指定参数，查询输出会被重置到标准输出。

如果参数以|开始，则该行的所有剩余部分总是会被当做要执行的*command*，并且在参数中不会执行变量篡改以及反引号展开。该行的剩余部分会被简单地按字面传给shell。

“查询结果”包括从数据库服务器得到的所有表、命令响应和提示，还有查询数据库的各种反斜线命令（如\d）的输出，但不包括错误消息。

> **提示**
>
> 要在查询结果之间混入文本输出，可以使用\qecho。

\p or \print

把当前查询缓冲区打印到标准输出。如果当前查询缓冲区为空，会打印最近被执行的查询。

\password [*username*]

更改指定用户（默认情况下是当前用户）的密码。这个命令会提示要求输入新密码、对密码加密然后把加密后的密码作为一个ALTER ROLE命令发送到服务器。这确保新密码不会以明文的形式出现在命令历史、服务器日志或者其他地方。

\prompt [*text*] *name*

提示用户提供一个文本用于分配给变量*name*。可以指定一个可选的提示字符串*text*（对于多个词组成的提示，把文本包裹在单引号中）。

默认情况下，\prompt使用终端进行输入和输出。不过，如果使用了-f命令行开关，\prompt会使用标准输入和标准输出。

\pset [*option* [*value*]]

这个命令设置影响查询结果表输出的选项。*option*表示要设置哪个选项。*value*的语义取决于选中的选项。对于某些选项，如果省略*value*会导致该选项值被切换或者被重置，具体是哪些选项可见特定选项的描述。如果没有上面提到的那种行为，那么省略*value*只会导致当前设置被显示。

不带任何参数的\pset显示所有打印选项的当前状态。

可调整的打印选项有：

border

*value*必须是一个数字。通常，数字越大，表格就会有更多的边框和线条，但具体要看是哪一种格式。在HTML格式中，这会直接被转换成border=...属性。在大部分其他格式中，只有值 0（没有边框）、1（内部分隔线）和 2（表格边框）有意义，并且 2 以上的值会被视为与border = 2相同。latex和latex-longtable格式会额外地允许一个值3表示在数据行之间增加分隔线。

columns

为wrapped格式设置目标宽度，还有扩展自动模式中决定输出是否足够多到需要分页器或者切换到垂直显示的宽度限制。零（默认）导致目标宽度由环境变量COLUMNS所控制，如果没有设置COLUMNS则使用检测到的屏幕宽度。此外，如果columns为零则wrapped格式只影响屏幕输出。如果columns为非零则文件和管道输出也会被包裹成该宽度。

expanded（or x）

如果*value*被指定，它必须是on或者off，它们分别会启用或者禁用扩展模式，也可以是auto。如果*value*被省略，则该命令会在开启和关闭设置之间切换。当扩展模式被启用时，查询结果被显示在两列中，第一列是列名而第二列是列值。如果在通常的“水平”模式中数据不适合屏幕，则可以用这种模式。在自动设置中，只要查询输出有多于一列并且比屏幕宽，就会使用扩展模式。否则，将使用常规模式。只有在对齐格式和wrapped格式中自动设置才有效。在其他格式中，它的行为总是像扩展模式被关闭一样。

fieldsep

指定在非对齐输出格式中使用的域分隔符。用那种方式，用户可以创建tab或者逗号分隔的输出，这种形式其他程序可能更喜欢。要设置tab为域分隔符，可以输入\pset fieldsep '\t'。默认的域分隔符是'|'（一个竖线）。

fieldsep_zero

把用在非对齐输出格式中的域分隔符设置为一个零字节。

footer

如果*value*被指定，它必须是on或者off，它们分别会启用或者禁用表格页脚（(*n* rows)计数）的显示。如果*value*被省略，则该命令会切换页脚显示为打开或者关闭。

format

设置输出格式为unaligned、aligned、wrapped、html、asciidoc、latex（使用tabular）、latex-longtable或者troff-ms之一。也可以使用不造成歧义的缩写（这意味着一个字母就够了）。

unaligned格式把一个数据行的所有列都写在一行上，之间用当前活动的域分隔符分隔。这可用于生成意图由其他程序读取的输出（例如，tab分隔或者逗号分隔格式）。

aligned格式是标准的、人类可读的、格式化好的文本输出，这是默认格式。

csv 格式写入以逗号分隔的列值，应用引号规则描述在RFC 4180。此输出与服务器COPY命令的 CSV 格式兼容。除非tuples_only参数为on，否则将生成具有列名称的标头行。不打印标题和页脚。每行都由系统相关的行尾字符结束，该字符通常是类似 Unix 的系统的单一换行符（\n）或应用于微软Windows的回车和换行顺序（\r\n）。除了逗号之外的\pset csv_fieldsep字段分隔符，还可以使用\pset csv_fieldsep选择。

wrapped格式和aligned相似，但是前者会把过宽的数据值分成多个行以便输出能够适合目标行的宽度。目标行的宽度由columns选项决定。注意uxsql将不会尝试对列头部标题进行换行，因此如果列头部需要的总宽度超过目标宽度，wrapped格式的行为就变得和aligned一样了。

html、asciidoc、latex、latex-longtable和troff-ms格式分别用相应的标记语言把要输出的表格放在文档中，不过它们的输出并不是完整的文档。在HTML中这可能并不重要，但是在LaTeX中必须有完整的文档。latex-longtable还要求有LaTeX的longtable以及booktabs包。

linestyle

设置边框线的绘制样式为ascii、old-ascii或者unicode之一。允许不产生歧义的缩写（这意味着一个字母就足够了）。默认的设置是ascii。这个选项只影响aligned以及wrapped输出格式。

ascii样式使用纯ASCII字符。数据中的新行使用一个+符号在右手边的空白处显示。当在wrapped格式中包裹两行中间没有新行字符的数据时，会在第一行右手边空白处显示一个点号（.），并且在下一行的左手边空白处也显示一个点号（.）。

old-ascii样式使用纯ASCII字符，数据中的新行使用:符号来代替左手边的列分隔符显示。在包裹两行中间没有新行字符的数据时，会用一个;符号取代左手边的列分隔符。

unicode样式使用Unicode的方框绘制字符。数据中的新行会使用一个回车符号显示在右手边的空白处。在包裹两行中间没有新行字符的数据时，会在第一行的右手边空白处显示一个省略号，并且在下一行的左手边空白处也显示一个省略号。

当border设置大于零时，linestyle选项也决定边框线用什么字符绘制。纯ASCII字符到处都可以使用，但是在识别Unicode字符的显示上使用Unicode字符会更好看。

null

设置要用来替代空值被打印的字符串。默认是什么也不打印，对于一个空字符串这很容易弄错。例如，有人可能更想用\pset null '(null)'。

numericlocale

如果*value*被指定，它必须是on或者off，它们将分别启用或者禁用一个与区域相关的字符来分隔数字和左边的十进制标记。如果*value*被省略，该命令会在常规输出和区域相关的数字输出之间切换。

pager

控制对查询和uxsql的帮助输出使用分页器程序。如果环境变量PAGER被设置，输出会被用管道输送到指定的程序。否则将使用与平台相关的默认分页器程序（例如more）。

如果pager选项被设为off，则不会使用分页器程序。如果pager选项被设为on，则会在适当的时候使用分页器，即当输出到终端并且无法适合屏幕时就会使用分页器。pager选项也可以被设置为always，这会导致对所有的终端输出都是用分页器而不管输出是否适合屏幕。不带*value*的\pset pager会切换分页器开、关状态。

pager_min_lines

如果pager_min_lines被设置为一个大于页面高度的数字，在至少这么多输出行被显示之前都不会调用分页器程序。默认设置为0。

recordsep

指定用在非对齐输出格式中的记录（行）分隔符。

recordsep_zero

把用在非对齐输出格式中的记录分隔符设置为一个零字节。

tableattr（or T）

在HTML格式中，这会指定要放在table标记内的属性。例如，这可能是cellpadding或者bgcolor。注意你可能不想在这里指定border，因为那由\pset border负责。如果没有给出*value*，则表属性会被重置。

在latex-longtable格式中，这个选项控制每个包含左对齐数据类型的列的宽度比例。这个选项的值是一个由空格分隔的值列表，例如'0.2 0.2 0.6'。没有指定的输出列会使用最后一个指定的值。

title（or C）

设置用于任何后续被打印表的表标题。这可以用来给输出加上描述性的标签。如果没有给出*value*，这个标题会被复原。

tuples_only（or t）

如果*value*被指定，它必须是on或者off，这个选项将启用或者禁用只显示元组的模式。如果*value*被省略，则该命令会在常规输出和只显示元组输出之间切换。常规输出包括列头、标题以及多种页脚之类的额外信息。在只显示元组的模式中，只会显示实际的表数据。

unicode_border_linestyle

设置unicode线型的边框绘制风格为single或者double之一。

unicode_column_linestyle

设置unicode线型的列绘制风格为single或者double之一。

unicode_header_linestyle

设置unicode线型的页眉绘制风格为single或者double之一。

这些不同格式的外观可以在示例小节的图示中看到。

> **提示**
>
> \pset有多种快捷命令。请参见\a、\C、\f、\H、\t、\T以及\x。

\q or \quit

退出uxsql程序。在一个脚本文件中，只有该脚本的执行会被终止。

\qecho *text* [...]

这个命令和\echo一样，不过输出将被写到\o所设置的查询输出通道。

\r or \reset

重置（清除）查询缓冲区。

\s [*filename*]

打印uxsql的命令行历史到*filename*。如果省略*filename*，该历史会被写入到标准输出（如果适用则使用分页器）。如果编译uxsql时没有加上Readline支持，则这个命令不可用。

\set [*name* [*value* [...]]]

设置uxsql变量*name*为*value*，如果给出了多于一个值，则把该变量的值设置为所有给出的值的串接。如果只给了一个参数，该变量会被设置为空字符串值。要重置一个变量，可以使用\unset 命令。

不带任何参数的\set显示所有当前设置的uxsql变量的名称和值。

合法的变量名可以包含字母、数字和下划线。详见下文的变量。变量名是大小写敏感的。

某些变量是特殊的，它们控制uxsql的行为或者会被自动设置以反映连接状态。这些变量在下文的变量中记录。

> **注意**
>
> 这个命令和SQL命令SET无关。

\setenv *name* [*value*]

把环境变量*name*设置为*value*，如果没有提供*value*，则会重置该环境变量。例如：

```
testdb=> \setenv PAGER less
testdb=> \setenv LESS -imx4F
```

\sf[+] *function_description*

这个命令以一个CREATE OR REPLACE FUNCTION命令取出并且显示指定函数或者过程的定义。定义会被打印到当前的查询输出渠道，就像\o所作的那样。

目标函数可以单独用名称指定，也可以用名称和参数指定，例如foo(integer, text)。如果有多于一个函数具有相同的名字，则必须给出参数的类型。

如果向命令名称追加+，那么输出行会被编号，函数体的第一行会被编为1。

与大部分其他元命令不同，该行的所有剩余部分总是会被当做\sf的参数，并且在参数中不会执行变量篡改以及反引号展开。

\sv[+] *view_name*

这个命令以一个CREATE OR REPLACE VIEW命令取出并且显示指定视图的定义。定义会被打印到当前的查询输出渠道，就像\o所作的那样。

如果在命令名称上追加+，那么输出行会从1开始编号。

与大部分其他元命令不同，该行的所有剩余部分总是会被当做\sv的参数，并且在参数中不会执行变量篡改以及反引号扩展。

\t

切换输出列名标题和行计数页脚的显示。这个命令等效于\pset tuples_only，提供它只是为了使用方便而已。

\T *table_options*

指定在HTML输出格式中，要放在table标签内的属性。这个命令等效于\pset tableattr *table_options*。

\timing [*on* | *off*]

使用参数，可以显示每个SQL语句打开或关闭的时间。没有参数，可以在开启和关闭之间切换显示。显示以毫秒为单位；时间间隔超过1秒也会以分钟：秒格式显示，如果需要，可添加小时和天数字段。

\unset *name*

重置（删除）uxsql变量*name*。

大部分控制uxsql行为的变量不能被重置，相反，\unset命令会被解释为把它们设置为其默认值。请参考下文的变量。

\w or \write *filename*
\w or \write |*command*

把当前查询缓冲区写到文件*filename*或者用管道导出到shell命令*command*。如果当前查询缓冲区为空，则写最近被执行的查询。

如果参数以|开始，则该行的整个剩余部分会被当做要执行的*command*，并且其中既不执行可变内插也不反向引用扩展。该行的其余部分只是简单地字面传递给shell。

\watch [*seconds*]

反复执行当前的查询缓冲区（就像\g那样）直到被中止或者查询失败。两次执行之间等待指定的秒数（默认是2秒）。显示每个查询结果时带上一个由\pset title字符串（如果有）、从查询开始起的时间以及延时间隔组成的页眉。

如果当前查询缓冲区为空，则会重新执行最近被发送的查询。

\x [*on* | *off* | *auto*]

设置或者切换扩展表格格式化模式。究其本身而言，这个命令等效于\pset expanded。

\z [*pattern*]

列出表、视图和序列，以及它们相关的访问特权。如果指定了*pattern*，则只会列出名称匹配该模式的表、视图和序列。

这是\dp（“display privileges”）的一个别名。

\! [*command*]

如果没有参数，就跳出到一个子shell，当子shell退出时uxsql会继续。如果有一个参数，则执行shell命令*command*。

与大部分其他元命令不同，该行的所有剩余部分总是会被当做\!的参数，并且在参数中不会执行变量篡改以及反引号展开。该行的剩余部分会被简单地按字面传递给shell。

\? [*topic*]

显示帮助信息。可选的*topic*参数（默认是commands）选择解释uxsql的哪一部分：commands表示uxsql的反斜线命令；options表示可以传递给uxsql的命令行选项；而variables显示有关uxsql配置变量的帮助。

\;

反斜线分号并非和前述命令相同的元命令，它只是会把一个分号加入到查询缓冲区且不会进一步执行。

通常，只要psql达到了命令结束的分号，它就将分发一个SQL命令给服务器，即使在当前行上还留有更多输入。因此，例如输入

```
select 1; select 2; select 3;
```

将导致三个SQL命令被逐个发送给服务器，在继续到下一个命令前会显示每一个命令的结果。不过，被输入为\;的分号将不会触发命令处理，这样在它之前的命令以及其后的命令实际上会被组合在一个请求中发送给服务器。例如

```
select 1\; select 2\; select 3;
```

会导致在到达非反斜线分号时用一个单一的请求把三个SQL命令发送给服务器。服务器会把这样一个请求当作单一的事务执行，除非该字符串中有显式的BEGIN/COMMIT命令把它划分成多个事务。psql对每个请求仅打印出它接收到的最后一个查询结果。在这个例子中，尽管所有三个SELECT确实都被执行了，但psql只会打印出3。

1.20.5.3.1. 模式（Pattern）

很多\d命令都可以用一个*pattern*参数来指定要被显示的对象名称。在最简单的情况下，模式正好就是该对象的准确名称。在模式中的字符通常会被变成小写形式（就像在SQL名称中那样），例如\dt FOO将会显示名为foo的表。就像在 SQL 名称中那样，把模式放在双引号中可以阻止它被转换成小写形式。如果需要在一个模式中包括一个真正的双引号字符，则需要把它写成两个相邻的双引号，这同样是符合 SQL 引用标识符的规则。例如，\dt "FOO""BAR"将显示名为FOO"BAR（不是foo"bar）的表。和普通的 SQL 名称规则不同，你不能只在模式的一部分周围放上双引号，例如\dt FOO"FOO"BAR将会显示名为fooFOObar的表。

只要*pattern*参数被完全省略，\d命令会显示在当前schema搜索路径中可见的全部对象 — 这等价于用*作为模式（如果一个对象所在的schema位于搜索路径中并且没有同类且同名的对象出现在搜索路径中该schema之前的schema中，则说该对象是可见的。这表示可以直接用名称引用该对象，而不需要用schema来进行限定）。要查看数据库中所有的对象而不管它们的可见性，可以把*.*用作模式。

如果放在一个模式中，*将匹配任意字符序列（包括空序列），而?会匹配任意的单个字符（这种记号方法就像 Unix shell 的文件名模式一样）。例如，\dt int*会显示名称以int开始的表。但是如果被放在双引号内，*和?就会失去这些特殊含义而变成普通的字符。

包含一个点号（.）的模式被解释为一个schema名称模式后面跟上一个对象名称模式。例如，\dt foo*.*bar*会显示名称以foo开始的 schema 中所有名称包括bar的表。如果没有出现点号，那么模式将只匹配当前schema搜索路径中可见的对象。同样，双引号内的点号会失去其特殊含义并且变成普通的字符。

高级用户可以使用字符类等正则表达式记法，如[0-9]可以匹配任意数字。如[0-9]可以匹配任意数字。.会作为一种分隔符，*会被翻译成正则表达式记号.*，?会被翻译成.，而$则按字面意思匹配。根据需要，可以通过书写?、(*R*+|)、(*R*|)和*R*?来分别模拟模式字符.、*R**和*R*?。$不需要作为一个正则表达式字符，因为模式必须匹配整个名称，而不是像正则表达式的常规用法那样解释（换句话说，$会被自动地追加到模式上）。如果不希望该模式的匹配位置被固定，可以在开头或者结尾写上*。注意在双引号内，所有的正则表达式特殊字符会失去其特殊含义并且按照其字面意思进行匹配。还有，在操作符名称模式中（即作为\do的参数），正则表达式特殊字符也按照字面意思进行匹配。

1.20.5.4. 高级特性

1.20.5.4.1. 变量

uxsql提供了和普通 Unix 命令shell相似的变量替换特性。变量简单来说就是一对名称/值，其中值可以是任意长度的任意字符串。名称必须由字母（包括非拉丁字母）、数字和下划线构成。

要设置一个变量，可以使用uxsql的元命令\set。例如，

```
testdb=> \set foo bar
```

会设置foo为值bar。要检索该变量的内容，可以在名称前放一个分号，例如：

```
testdb=> \echo :foo
bar
```

这在常规SQL命令和元命令中均有效，下文的SQL中插入变量中有更多细节。

如果调用\set时没有第二个参数，该变量会被设置为一个空字符串值。要重置（即删除）一个变量，可以使用命令\unset。要显示所有变量的值，在调用\set时不带任何参数即可。

> **注意**
>
> \set bar :foo是一种很好的拷贝变量的方法。

有一些变量会被uxsql特殊对待。它们表示特定的选项设置，运行时这类选项设置可以通过修改该变量的值来改变，或者在某些情况下它们表示uxsql的可更改的状态。按照惯例，所有被特殊对待的变量的名称由全部大写形式的 ASCII 字母（还有可能是数字和下划线）组成。为了确保未来最大的兼容性，最好避免把这类变量名用于自己的目的。

控制uxsql行为的变量通常不能被重置或者设置为无效值。允许\unset命令，但它会被解释为将变量设置为它的默认值。没有第二参数的\set命令会被解释为将变量设置为on（对于接受该值的控制变量），对不接受该值的变量则会拒绝这个命令。此外，接受值on和off的控制变量也能接受其他常见的布尔值拼写方式，例如true和false。

被特殊对待的变量是：

AUTOCOMMIT

在被设置为on（默认）时，每一个SQL命令在成功完成时会被自动提交。在这种模式中要推迟提交，必须输入一个**BEGIN**或者**START TRANSACTION** SQL命令。当被设置为off或者被重置时，在显式发出**COMMIT**或者**END**之前，SQL命令不会被提交。自动提交打开模式会为你发出一个隐式的**BEGIN**，这会发生在任何不在一个事务块中且本身即不是**BEGIN**及其他事务控制命令且不是无法在事务块中执行的命令（例如**VACUUM**）之前。

> **注意**
>
> 在自动提交关闭模式中，必须通过**ABORT**或者**ROLLBACK**显式地放弃任何失败的事务。还要记住，如果退出会话时没有提交，则所有的工作都会丢失。

> **注意**
>
> 自动提交打开模式是UXDB的传统行为，但是自动提交关闭模式更接近于SQL的规范。如果更喜欢自动提交关闭模式，可以在系统级的uxsqlrc文件或者个人的~/.uxsqlrc文件中设置它。

COMP_KEYWORD_CASE

确定在补全一个SQL关键词时要使用的大小写形式。如果被设置为lower或者upper，补全后的词将分别是小写或者大写形式。如果被设置为preserve-lower或者preserve-upper（默认），补全后的词将会保持该词已输入部分的大小写形式，但是如果被补全的词还没有被输入，则它会被分别补全成小写或者大写形式。

DBNAME

当前已连接的数据库名称。每次连接到一个数据库时都会设置该变量（包括程序启动时），但是可以被更改或者重置。

ECHO

如果被设置为all，所有非空输入行会被按照读入它们的样子打印到标准输出（不适用于交互式读取的行）。要在程序开始时选择这种行为，可以使用开关-a。如果被设置为queries，uxsql会在发送每个查询给服务器时将它们打印到标准输出。选择这种行为的开关是-e。如果被设置为errors，那么只有失败的查询会被显示在标准错误输出上。这种行为的开关是-b。如果被重置或者设置为none（默认值）则不会显示任何查询。

ECHO_HIDDEN

当这个变量被设置为on且一个反斜线命令查询数据库时，相应的查询会被先显示。这种特性可以帮助我们学习UXDB的内部并且在自己的程序中提供类似的功能（要在程序开始时选择这种行为，可以使用开关-E）。如果把这个变量设置为值noexec，则对应的查询只会被显示而并不真正被发送给服务器执行。默认值是off。

ENCODING

当前的客户端字符集编码。每一次你连接到一个数据库（包括程序启动）时以及当你用\encoding更改编码时，这个变量都会被设置，但它可以被更改或者重置。

ERROR

如果上一个SQL查询失败则为true，如果成功则是false。另见SQLSTATE。

FETCH_COUNT

如果这个变量被设置为一个大于零的整数值，SELECT查询的结果会以一组一组的方式取出并且显示（而不是像默认的那样把整个结果集拿到以后再显示），每一组就会包括这么多个行。因此，这种方式只会使用有限的内存量，而不管整个结果集的大小。在启用这个特性时，通常会使用100到1000的设置。记住在使用这种特性时，一个查询可能会在已经显示了一些行之后失败。

> **提示**
>
> 尽管可以把这种特性用于任何的输出格式，但是默认的aligned格式看起来会比较糟糕，因为每一组的FETCH_COUNT个行将被单独格式化，这就会导致不同的行组的列宽不同。其他的输出格式会更好。

HIDE_TABLEAM

如果此变量设置为true，则不显示表的访问方法的详细信息。这主要用于回归测试。

HISTCONTROL

如果这个变量被设置为ignorespace，则以一个空格开始的行不会被放入到历史列表中。如果被设置为值ignoredups，则匹配之前的历史行的行不会被放入。值ignoreboth组合了上述两种值。如果被重置或者被设置为none（默认值），所有在交互模式中被读入的行都会保存在历史列表中。

HISTFILE

该文件名将被用于存储历史列表。如果被重设，文件名将从UXSQL_HISTORY环境变量中取得。如果该环境变量也没有被设置，则默认值是~/.UXSQL_HISTORY，在Windows上是%APPDATA%\UXDB\UXSQL_HISTORY。例如，

\set HISTFILE ~/.uxsql_history- :DBNAME

放在~/.uxsqlrc中将会导致uxsql为每一个数据库维护一个单独的历史。

HISTSIZE

存储在命令历史中的最大命令数（默认值是500）。如果被设置为一个负值，则不会应用限制。

HOST

当前连接到的数据库服务器端口。每次连接到一个数据库时都会设置该变量（包括程序启动时），但是可以被更改或者重置。

IGNOREEOF

如果被设置为1或者更小，向一个uxsql的交互式会话发送一个EOF字符（通常是Control+D）将会终止应用。如果设置为一个较大的数字值，则必须输入多个连续的EOF字符才能让交互式会话终止。如果该变量被设置为一个非数字值，则它会被解释为10。默认值为0。

LAST_ERROR_MESSAGE
LAST_ERROR_SQLSTATE

当前uxsql会话中最近一个失败查询的主错误消息和相关的SQLSTATE代码，如果在当前会话中没有发生错误，则是一个空字符串和00000。

LASTOID

最后被影响的OID的值，这可能会由INSERT或者\lo_import命令返回。这个变量只保证在下一个SQL命令被显示完之前有效。

ON_ERROR_ROLLBACK

当被设置为on时，如果事务块中的一个语句产生一个错误，该错误会被忽略并且该事务会继续。当被设置为interactive时，只在交互式会话中忽略这类错误，而读取脚本文件时则不会忽略错误。当被重置或者设置为off（默认值）时，事务块中产生错误的一个语句会中止整个事务。错误回滚模式的工作原理是在事务块的每个命令之前都为你发出一个隐式的SAVEPOINT，然后在该命令失败时回滚到该保存点。

ON_ERROR_STOP

默认情况下，出现一个错误后命令处理会继续下去。当这个变量被设置为on后，出现错误后命令处理会立即停止。在交互模式下，uxsql将会返回到命令提示符；否则，uxsql将会退出并且返回错误代码 3 来把这种情况与致命错误区分开来，致命错误会被报告为错误代码1。在两种情况下，任何当前正在运行的脚本（顶层脚本以及任何它已经调用的其他脚本）将被立即中止。如果顶层命名字符串包含多个SQL命令，将在当前命令处停止处理。

PORT

当前连接到的数据库服务器端口。每次连接到一个数据库时都会设置该变量（包括程序启动时），但是可以改变或重置。

PROMPT1
PROMPT2
PROMPT3

这些变量指定uxsql发出的提示符的模样。见下文的提示符。

QUIET

把这个变量设置为on等效于命令行选项-q。在交互模式下可能用处不大。

SERVER_VERSION_NAME
SERVER_VERSION_NUM

服务器的版本号作为字符串，例如2.0.4.10、2.0.4.11，和以数字形式，例如02000410或02000411。每次连接到数据库（包括程序启动）时都会设置这些值，但可以更改或取消设置。

SHOW_CONTEXT

这个变量可以被设置为值never、errors或者always来控制是否在来自服务器的消息中显示CONTEXT域。默认是errors（表示在错误消息中显示上下文，但在通知和警告消息中不显示）。当VERBOSITY被设置为terse时，这个设置无效（另见\errverbose，它可以用来得到刚遇到的错误的详细信息）。

SINGLELINE

设置这个变量为on等效于命令行选项-S。

SINGLESTEP

设置这个变量为on等效于命令选项-s。

SQLSTATE

与上一个SQL查询的失败相关的错误代码，如果上一个查询成功则为00000。

USER

当前连接的数据库用户。每次连接到一个数据库时都会设置该变量（包括程序启动时），但是可以被更改或者重置。

VERBOSITY

这个变量可以被设置为值default、verbose或者terse来控制错误报告的详细程度（另见\errverbose，在想得到之前的错误的详细版本时使用）。

VERSION
VERSION_NAME
VERSION_NUM

这些变量在程序启动时被设置的，分别将将uxsql的版本反映为一个字符串（例如2.0.4.11）和一个数字（例如02000411）。他们可以更改或取消设置。

1.20.5.4.2. SQL 中插入变量

uxsql变量的一个关键特性是可以把它们替换（“插入”）到常规SQL语句中，也可以把它们作为元命令的参数。此外，uxsql还提供了功能来确保被用作SQL文字和标识符的变量值会被正确地引用。插入一个值而不需要加引用的语法是在变量名前面加上一个冒号（:）。例如，

```
testdb=> \set foo 'my_table'
testdb=> SELECT * FROM :foo;
```

将查询表my_table。注意这可能会不安全：该变量的值会被按字面拷贝，因此它可能包含不平衡的引号甚至反斜线命令。必须确保把它放在那里是有意义的。

当一个值被用作SQL文本或者标识符时，最安全的是把它加上引用。要引用一个变量的值作为SQL文本，可以把变量名称放在单引号中并且在引号前面写一个冒号。要引用作为SQL标识符，则可以把变量名称放在双引号中并且在引号前面写一个冒号。这种结构可以正确地处理变量值中嵌入的引号和其他特殊字符。之前的例子用这种方法写会更安全：

```
testdb=> \set foo 'my_table'
testdb=> SELECT * FROM :"foo";
```

在被引用的SQL文本和标识符中将不会执行变量插入。因此，一个诸如':foo'的结构不会从一个变量的值产生一个被引用的文本（即便能够也会不安全，因为无法正确地处理嵌入在值中的引号）。

使用这种机制的一个例子是把一个文件的内容拷贝到一个表列中。首先把该文件载入到一个变量，然后把该变量的值作为一个被引用的字符串插入：

```
testdb=> \set content `cat my_file.txt`
testdb=> INSERT INTO my_table VALUES (:'content');
```

> **注意**
>
> （注意如果my_file.txt包含NUL字节，这样也不行。uxsql不支持在变量值中嵌入 NUL 字节）。

因为冒号可以合法地出现在SQL命令中，一次明显的插入尝试（即:name、:'name'或者:"name"）不会被替换，除非所提及的变量就是当前被设置的。在任何情况下，可以用一个反斜线对冒号进行转义以避免它被替换。

:{?*name*}特殊语法根据该变量存在与否返回TRUE或者FALSE，并且因此总是会被替换，除非分号被反斜线转义。

变量的冒号语法对嵌入式查询语言（例如ECUX）来说是标准的SQL。用于数组切片和类型造型的冒号语法是UXDB扩展，它有时可能会与标准用法冲突。把一个变量值转义成SQL文本或者标识符的冒号引用语法是一种uxsql扩展。

1.20.5.4.3. 提示符

uxsql发出的提示符可以根据用户的喜好自定义。PROMPT1、PROMPT2和PROMPT3这三个变量包含了描述提示符外观的字符串和特殊转义序列。Prompt 1是当uxsql等待新命令时发出的常规提示符。Prompt 2是在命令输入时需要更多输入时发出的提示符，例如因为当命令没有被分号终止或者引用没有被关闭时就会发出这个提示符。在运行一个SQL **COPY FROM STDIN**命令并且需要在终端上输入一个行值时，会发出 Prompt3。

被选中的提示符变量会被原样打印，除非碰到一个百分号（%）。百分号的下一个字符会被特定的其他文本替换。预定义好的替换有：

%M

数据库服务器的完整主机名（带有域名），或者当该连接是建立在一个 Unix 域套接字上时则是[local]，或者当 Unix 域套接字不在编译在系统内的默认位置上时则是[local:*/dir/name*]。

%m

数据库服务器的主机名称（在第一个点处截断），或者当连接建立在一个 Unix 域套接字上时是[local]。

%>

数据库服务器正在监听的端口号。

%n

数据库会话的用户名（在数据库会话期间，这个值可能会因为命令**SET SESSION AUTHORIZATION**的结果而改变）。

%/

当前数据库的名称。

%~

和%/类似，但是如果数据库是默认数据库时输出是~（波浪线）。

%#

如果会话用户时一个数据库超级用户，则是#，否则是一个>（在数据库会话期间，这个值可能会因为命令SET SESSION AUTHORIZATION的结果而改变）。

%p

当前连接到的后端的进程ID。

%R

在提示符1下通常是=，但如果会话位于一个条件块的一个非活动分支中则是@，如果会话处于单行模式中则是^，如果会话从数据库断开连接（\connect失败时会发生这种情况）则是!。在提示符2中，根据为什么uxsql期待更多的输入，%R会被一个相应的字符替换：如果命令还没有被终止是-，如果有一个未完的/* ... */注释则是*，如果有一个未完的被引用字符串则是一个单引号，如果有一个未完的被引用标识符则是一个双引号，如果有一个未完的美元引用字符串则是一个美元符号，如果有一个还没有被配对的左圆括号则是(。在提示符3中%R不会产生任何东西。

%x

事务状态：当不在事务块中时是一个空字符串，在一个事务块中时是*，在一个失败的事务块中时是!，当事务状态是未判定时（例如因为没有连接）为?。

%l

当前语句中的行号，从1开始。

%*digits*

带有指定的八进制码的字符会被替换。

%:*name*:

uxsql变量*name*的值。详见变量。

%`*command*`

*command*的输出，类似于平常的“反引号”替换。

%[... %]

提示符可以包含终端控制字符，例如改变提示符文本的颜色、背景或者风格以及更改终端窗口标题的控制字符。为了让Readline的行编辑特性正确工作，这些不可打印的控制字符必须被包裹在%[和%]之间以指定它们是不可见的。在提示附中可以出现多个这样的标识对。例如：

testdb=> \set PROMPT1 '%[%033[1;33;40m%]%n@%/%R%[%033[0m%]%# '

会导致一个在兼容 VT100 的彩色终端上的粗体（1;）的、黑底黄字（33;40）的提示符。

要在你的提示符中插入一个百分号，可以写成%%。提示符1和2的默认提示是'%/%R%# '，提示符3的提示是'>> '。

1.20.5.4.4. 命令行编辑

为了方便的行编辑和检索，uxsql支持Readline库。uxsql退出时命令历史会被自动保存，而当uxsql启动时命令历史会被重新载入。uxsql也支持tab补全，不过补全逻辑绝不是一个SQL解析器。tab补全产生的查询也可能会受其他SQL命令干扰，例如SET TRANSACTION ISOLATION LEVEL。如果出于某种原因不想用tab键补全，可以把下面的代码放在主目录下的名为.inputrc文件中关闭该特性：

```
$if uxsql
set disable-completion on
$endif
```

（这不是uxsql特性而是Readline的特性。）

1.20.6. 环境变量

COLUMNS

如果\pset columns为零，这个环境变量控制用于wrapped格式的宽度以及用来确定是否输出需要用到分页器或者切换到扩展自动模式中的垂直格式的宽度。

PAGER

如果屏幕无法容纳查询结果，则查询结果会通过这个命令导出。典型的值是more或者less。它的默认值与平台相关。通过将PAGER设置为空或使用\pset命令的分页器相关选项，可以禁用分页器的使用。

UXDATABASE
UXHOST
UXPORT
UXUSER

默认连接参数。

UX_COLOR

规定在诊断消息中是否使用颜色。可能的值为always、 auto、never。

UXSQL_EDITOR
EDITOR
VISUAL

\e、\ef以及\ev命令所使用的编辑器。会按照列出的顺序检查这些变量，第一个被设置的变量将被使用。如果都没有被设置，默认是使用Unix系统上的vi或者Windows系统上的notepad.exe。

UXSQL_EDITOR_LINENUMBER_ARG

当\e、\ef或者\ev带有一个行号参数时，这个变量指定用于传递起始行号给用户编辑器的命令行参数。对于Emacs或者vi之类的编辑器，这个变量是一个加号。如果需要在选项名称和行号之间有空格，可以在该变量的值中包括一个结尾的空格。例如：

```
UXSQL_EDITOR_LINENUMBER_ARG='+'
UXSQL_EDITOR_LINENUMBER_ARG='--line '
```

在 Unix 系统上默认是+（对应于默认编辑器vi，且对很多其他常见编辑器可用）。在 Windows 系统上没有默认值。

UXSQL_HISTORY

命令历史文件的替代位置。波浪线（~）扩展会被执行。

UXSQL_PAGER
PAGER

如果一个查询的结果在屏幕上放不下，它们会通过这个命令分页显示。典型的值是more或less。通过把UXSQL_PAGER或PAGER设置为空字符串可以禁用分页器的使用，调整\uxset命令与分页器相关的选项也能达到同样的效果。会按照列出的顺序检查这些变量，第一个被设置的将被使用。如果都没有被设置，则大部分平台上默认使用more，但在Cygwin上使用less。

UXSQLRC

用户的.uxsqlrc文件的替代位置。波浪线（~）扩展会被执行。

SHELL

被\!命令执行的命令。

TMPDIR

存储临时文件的目录。默认是/tmp。

和大部分其他UXDB工具一样，这个工具也使用libuxsql所支持的环境变量。

1.20.7. 文件

uxsqlrc and ~/.uxsqlrc

如果没有-X选项，在连接到数据库后但在接收正常的命令之前，uxsql会尝试依次从系统级的启动文件（uxsqlrc）和用户的个人启动文件（~/.uxsqlrc）中读取并且执行命令。这些文件可以被用来设置客户端或者服务器，通常是一些\set和SET命令。

系统级的启动文件是uxsqlrc，它应该在安装好的UXDB的“系统配置”目录中，最可靠的定位方法是运行ux_config --sysconfdir。默认情况下，这个目录将是../etc/（相对于包含UXDB可执行文件的目录）。可以通过UXSYSCONFDIR环境变量显式地设置这个目录的名称。

用户个人的启动文件是.uxsqlrc，它应该在调用用户的主目录中。在Windows上，由于没有用户主目录的概念，个人的启动文件是%APPDATA%\UXDB\uxsqlrc.conf。用户启动文件的位置可以通过uxsqlrc环境变量设置。

系统级和用户个人的启动文件都可以弄成是针对特定uxsql版本的，方法是在文件名后面加上一个横线以及UXDB的版本号，例如~/.uxsqlrc-2.0.4.11或者~/.uxsqlrc-2.0.4.11。版本最为匹配的文件会优先于不那么匹配的文件读入。

.uxsql_history

命令行历史被存储在文件~/.uxsql_history中，或者是 Windows 的文件%APPDATA%\uxsinodb\uxsql_history中。

历史文件的位置可以通过uxsql变量HISTFILE 或者UXSQL_HISTORY环境变量显示的设置。

1.20.8. 注解

- uxsql和具有相同主版本或者更老的主版本服务器最为匹配。如果服务器的版本比uxsql本身要高，则反斜线命令尤其容易失败。运行SQL命令并且显示查询结果的一般功能应该也能和具有更新主版本的服务器一起使用，但是并非在所有的情况下都能保证如此。

  如果你想用uxsql连接到多个具有不同主版本的服务器，推荐使用最新版本的uxsql。或者，你可以为每一个主版本保留一份uxsql拷贝，并且针对相应的服务器使用匹配的版本。但实际上，这种额外的麻烦是不必要的。

1.20.9. 给Windows用户的注解

uxsql是一个“控制台应用”。由于Windows的控制台窗口使用的是一种和系统中其他应用不同的编码，在uxsql中使用8位字符时要特别注意。如果uxsql检测到一个有问题的控制台代码页，它将会在启动时警告你。要更改控制台代码页，有两件事是必要的：

- 输入**cmd.exe /c chcp 1252**可以设置代码页（1252 是适用于德语的一个代码页，请在这里替换成你的值）。如果正在使用 Cygwin，可以把这个命令放在/etc/profile中。
- 把控制台字体设置为Lucida Console，因为栅格字体无法与ANSI代码页一起使用。

1.20.10. 示例

第一个例子展示了如何如何跨越多行输入一个命令。注意提示符的改变：

```
testdb=> CREATE TABLE my_table (
testdb(>  first integer not null default 0,
testdb(>  second text)
testdb-> ;
CREATE TABLE
```

现在再看看表定义：

```
testdb=> \d my_table
            Table "public.my_table"
 Column |  Type   | Collation | Nullable | Default
--------+---------+-----------+----------+---------
 first  | integer |           | not null | 0
 second | text    |           |          |
```

现在我们把提示符改一改：

```
testdb=> \set PROMPT1 '%n@%m %~%R%# '
user@[local] testdb=>
```

假定已经用数据填充了这个表并且想看看其中的数据：

```
peter@localhost testdb=> SELECT * FROM my_table;
 first | second
-------+--------
     1 | one
     2 | two
     3 | three
     4 | four
(4 rows)
```

你可以用\pset命令以不同的方式显示表：

```
peter@localhost testdb=> \pset border 2
Border style is 2.
peter@localhost testdb=> SELECT * FROM my_table;
+-------+--------+
| first | second |
+-------+--------+
|     1 | one    |
|     2 | two    |
|     3 | three  |
|     4 | four   |
+-------+--------+
(4 rows)

peter@localhost testdb=> \pset border 0
Border style is 0.
peter@localhost testdb=> SELECT * FROM my_table;
first second
----- ------
    1 one
    2 two
    3 three
    4 four
(4 rows)

peter@localhost testdb=> \pset border 1
Border style is 1.
peter@localhost testdb=> \pset format unaligned
Output format is unaligned.
peter@localhost testdb=> \pset fieldsep ","
Field separator is ",".
peter@localhost testdb=> \pset tuples_only
Showing only tuples.
peter@localhost testdb=> SELECT second, first FROM my_table;
one,1
two,2
```

three,3
four,4

或者使用短命令：

```
peter@localhost testdb=> \a \t \x
Output format is aligned.
Tuples only is off.
Expanded display is on.
peter@localhost testdb=> SELECT * FROM my_table;
-[ RECORD 1 ]-
first  | 1
second | one
-[ RECORD 2 ]-
first  | 2
second | two
-[ RECORD 3 ]-
first  | 3
second | three
-[ RECORD 4 ]-
first  | 4
second | four
```

如果需要，可以用\crosstabview命令以交叉表的形式显示查询结果：

```
testdb=> SELECT first, second, first > 2 AS gt2 FROM my_table;
 first | second | gt2
-------+--------+-----
     1 | one    | f
     2 | two    | f
     3 | three  | t
     4 | four   | t
(4 rows)

testdb=> \crosstabview first second
 first | one | two | three | four
-------+-----+-----+-------+------
     1 | f   |     |       |
     2 |     | f   |       |
     3 |     |     | t     |
     4 |     |     |       | t
(4 rows)
```

这第二个例子展示了表的“乘法”（连接），行按照序号降序排序且列按照独立的、升序的方式排序。

```
testdb=> SELECT t1.first as "A", t2.first+100 AS "B", t1.first*(t2.first+100) as "AxB",
testdb(> row_number() over(order by t2.first) AS ord
testdb(> FROM my_table t1 CROSS JOIN my_table t2 ORDER BY 1 DESC
testdb(> \crosstabview "A" "B" "AxB" ord
 A | 101 | 102 | 103 | 104
---+-----+-----+-----+-----
```

```
 4 | 404 | 408 | 412 | 416
 3 | 303 | 306 | 309 | 312
 2 | 202 | 204 | 206 | 208
 1 | 101 | 102 | 103 | 104
(4 rows)
```

1.21. vacuumdb

vacuumdb — 对一个UXDB数据库进行垃圾清理和分析

1.21.1. 用法

vacuumdb [*connection-option...*] [*option...*] [--table | -t *table* [(*column* [,...])]] ... [*dbname*]

vacuumdb [*connection-option...*] [*option...*] --all | -a

1.21.2. 描述

vacuumdb是用于清理一个UXDB数据库的工具。vacuumdb也将产生由UXDB查询优化器所使用的内部统计信息。

vacuumdb是SQL命令**VACUUM**的封装。在通过这个工具和其他方法访问服务器来清理和分析数据库之间没有实质性的区别。

1.21.3. 选项

vacuumdb接受下列命令行参数：

-a
--all

清理所有数据库。

[-d] *dbname*
[--dbname=]*dbname*

指定要被清理或分析的数据库名。如果没有被指定并且没有使用-a（或--all），数据库名将从环境变量UXDATABASE中读出。如果环境变量也没有设置，指定给该连接的用户名将用作数据库名。

--disable-page-skipping

根据可见性地图的内容禁用跳过页面。

-e
--echo

回显vacuumdb生成并发送给服务器的命令。

-f
--full

执行“完全”清理。

-F
--freeze

强有力地“冻结”元组。

-j *njobs*
--jobs=*njobs*

通过同时运行*njobs*个命令来并行执行清理或者分析命令。这个选项会减少处理的时间，但是它也会增加数据库服务器的负载。

vacuumdb将开启*njobs*个到数据库的连接，因此请确认你的max-connections设置足够高以容纳所有的连接。

> **注意**
>
> 注意如果某些系统目录被并行处理，使用这种模式加上-f（FULL）选项可能会导致死锁失败。

--min-mxid-age *mxid_age*

仅在multixact ID 年龄至少为*mxid_age*的表上执行清空或分析命令。此设置对于确定要处理的表的优先级比较有用，以防止multixact ID 回绕。

对于此选项的用途，关系的multixact ID年龄是主关系及其关联的TOAST表的年龄中最大的，如果存在的话。由于vacuumdb发出的命令在需要时还将处理TOAST表的关系，它无需单独考虑。

--min-xid-age *xid_age*

仅在事务ID 年龄至少为*xid_age*的表上执行清空或分析命令。此设置对于确定要处理的表的优先级比较有用，以防止事务ID回绕。

对于此选项的用途，关系的事务ID年龄是主关系及其关联的TOAST表的年龄中最大的，如果存在的话。由于vacuumdb发出的命令在需要时还将处理TOAST表的关系，它无需单独考虑。

-q
--quiet

不显示进度消息。

--skip-locked

跳过无法立即锁定以进行处理的关系。

-t *table* [(*column* [,...])]
--table=*table* [(*column* [,...])]

只清理或分析*table*。列名只能和--analyze或--analyze-only选项一起被指定。通过写多个-t开关可以清理多个表。

-v
--verbose

在处理期间打印详细信息。

-V
--version

打印vacuumdb版本并退出。

-z
--analyze

计算优化器使用的统计信息。

-Z
--analyze-only

只计算优化器使用的统计信息（不清理）。

--analyze-in-stages

与--analyze-only相似，只计算优化器使用的统计信息（不做清理）。使用不同的配置设置运行分析的几个（目前是3个）阶段以更快地产生可用的统计信息。

这个选项对分析一个刚从转储恢复或者通过ux_upgrade得到的数据库有用。这个选项将尝试尽可能快地创建一些统计信息来让该数据库可用，然后在后续的阶段中产生完整的统计信息。

-?
--help

显示有关vacuumdb命令行参数的帮助并退出。

vacuumdb也接受下列命令行参数用于连接参数：

-h *host*
--host=*host*

指定运行服务器的机器的主机名。如果该值以一个斜线开始，它被用作 Unix 域套接字的目录。

-p *port*
--port=*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*
--username=*username*

要作为哪个用户连接。

-w
--no-password

永远不要发出密码提示。如果服务器需要密码验证，而通过其他方式(如.pgpass文件)无法获得密码，则连接尝试将失败。此选项在没有用户可以输入密码的批处理作业和脚本中很有用。

-W
--password

强制vacuumdb在连接到一个数据库之前提示输入密码。

这个选项不是必需的，因为如果服务器要求输入密码，vacuumdb将自动提示输入密码。但是，vacuumdb将浪费一次连接尝试来发现服务器想要一个密码。在某些情况下建议用-W来避免额外的连接尝试。

--maintenance-db=*dbname*

指定要连接到来发现哪些其他数据库应该被清理的数据库名。如果没有指定，将使用uxdb数据库。而如果它也不存在，将使用template1。

1.21.4. 环境变量

UXDATABASE
UXHOST
UXPORT
UXUSER

默认连接参数。

UX_COLOR

规定在诊断消息中是否使用颜色。可能的值为always、 auto、never。

和大部分其他UXDB工具相似，这个工具也使用libuxsql支持的环境变量。

1.21.5. 诊断

在有困难时，可以在VACUUM和uxsql中找潜在问题和错误消息的论述。数据库服务器必须运行在目标主机上。同样，任何libuxsql前端库使用的默认连接设置和环境变量都将适用于此。

1.21.6. 注解

vacuumdb可能需要多次连接到UXDB服务器，每次都询问一个密码。在这种情况下有一个~/.uxpass文件会很方便。

1.21.7. 示例

要清理数据库test：

$ **vacuumdb test**

要清理和为优化器分析一个名为bigdb的数据库：

$ **vacuumdb --analyze bigdb**

要清理在名为xyzzy的数据库中的一个表foo，并且为优化器分析该表的bar列：

$ **vacuumdb --analyze --verbose --table 'foo(bar)' xyzzy**

1.22. vacuumlo

vacuumlo — 从UXDB数据库中移除孤立的大对象

1.22.1. 用法

vacuumlo［*option*...］ *dbname*...

1.22.2. 描述

vacuumlo是一个从UXDB数据库中移除“孤立”大对象的简单使用程序。一个孤立的大对象（LO）是指其OID不出现在数据中任何oid或lo数据列中的LO。

如果你使用该程序，你也许还会对lo模块中的lo_manage触发器感兴趣。lo_manage对于避免创建孤立LO有用处。

在命令行中提到的所有数据库都将被处理。

1.22.3. 选项

vacuumlo接受下列命令行参数：

-l *limit*
--limit= *limit*

在每一个事务中移除不超过*limit*个大对象（默认值为1000）。因为移除每一个LO时服务器都将要求一个锁，所以在一个事务中移除过多的LO会有超过max_locks_per_transaction的风险。如果你想在一个事务中就完成所有的移除工作，请将这个限制设置为0。

-n
--dry-run

不移除任何东西，只是显示下一步操作。

-v
--verbose

显示进度消息。

-V
--version

打印vacuumlo的版本并退出。

-?
--help

显示关于vacuumlo的命令行参数，并且退出。

vacuumlo也接受下列命令行参数用于连接：

-h *host*
--host=*host*

数据库服务器的主机名。

-p *port*
--port=*port*

数据库服务器的端口。

-U *username*
--username= *username*

用于连接的用户名。

-w
--no-password

不要发出一个密码提示。如果服务器要求输入密码并且没有其他方式可以提供密码（例如一个.uxpass文件），连接尝试将会失败。这个选项可用于批处理任务以及脚本，因为在这些情况下不会有用户输入密码。

-W
--password

强制vacuumlo在连接到数据库之前提示输入密码。

这个选项不是不可缺少的，因为如果服务器要求输入密码，vacuumlo会自动提示输入密码。但是，vacuumlo将会浪费一次连接尝试来了解到服务器需要密码。在某些情况，建议用-W来避免这种额外的连接尝试。

1.22.4. 环境变量

UXHOST
UXPORT
UXUSER

默认连接参数。

和大部分其他UXDB工具相似，这个工具也使用libuxsql支持的环境变量。

1.22.5. 说明

vacuumlo按照下列方法工作：首先vacuumlo建立一个临时表，其中包含所选择数据库中所有大对象的OID。然后它会扫描数据库中所有类型为oid或lo的列，并且从临时表中移除在这些列中出现过的OID（注意：只有类型为这些名字的才被考虑，特别的，在这些类型之上的域是不被考虑的）。临时表中剩下的项就标识了孤立LO。它们将被移除。

第 2 章 UXDB服务器应用

本节包含UXDB服务端应用程序和支持应用程序的参考信息。这些命令只能在数据库服务器所在的主机上有效地运行。

2.1. initdb

initdb — 创建一个新的UXDB数据库集群

2.1.1. 用法

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

2.1.2. 描述

initdb创建一个新的UXDB数据库集群。一个数据库集群是由一个单一服务器实例管理的数据库的集合。

一个数据库集群的创建包括创建存放数据库数据的目录、生成共享目录表（属于整个集群而不是任何特定数据库的表）并且创建template1和uxdb数据库。当你后来创建一个新的数据库时，任何在template1数据库中的东西都会被复制（因此，任何已安装在template1中的东西都会被自动地复制到后来创建的每一个数据库中）。uxdb数据库是便于用户、工具和第三方应用使用的默认数据库。

尽管initdb将尝试创建指定的数据目录，它可能没有权限（如果想要的数据目录的父目录被根用户拥有）。要在这样一种设置中初始化，作为root创建一个空数据目录，然后使用chown将该目录赋予给数据库用户账户，再然后su成为该数据库用户并运行initdb。

initdb必须以将拥有该服务器进程的用户运行，因为该服务器需要访问initdb创建的文件和目录。因为该服务器不能作为root运行，你不能以root运行initdb（事实上它会拒绝这样做）。

由于安全原因，由initdb创建的新集群默认将只能由集群拥有者访问。--allow-group-access选项允许与集群拥有者同组的任何用户读取集群中的文件。这对非特权用户执行备份很有用。

initdb初始化该数据库集群的默认区域和字符集编码。当一个数据库被创建时，其字符集编码、排序顺序（LC_COLLATE）和字符集类（LC_CTYPE，例如大写、小写、数字）可以被独立设置。initdb为template1数据库确定那些设置，它们将作为所有其他数据库的默认值。

要修改默认排序顺序或字符集类，使用--lc-collate和--lc-ctype选项。除C或POSIX之外的排序顺序也有性能罚值。由于这些原因，在运行initdb时选择正确的区域很重要。

余下的区域分类可以在服务器启动之后改变。你也可以使用--locale来为所有区域分类设置默认值，包括排序顺序和字符集类。所有服务器区域值（lc_*）可以通过SHOW ALL显示。

要修改默认编码，使用--encoding。

大小写敏感，使用--ignore-case，可以选择查询结果按照原文还是转化成小写输出创建时的表名、列名、索引名等。

2.1.3. 选项

-a
--security

初始化带有安全功能的UXDB实例。安全功能请参见《优炫数据库安全功能手册 2.1》。

-A *authmethod*
--auth=*authmethod*

这个选项为本地用户指定在ux_hba.conf中使用的默认认证方法（host和local行）。initdb将使用指定的认证方法为非复制连接以及复制连接填充ux_hba.conf项。

除非你信任你系统上的所有本地用户，不要使用trust。为了安装方便，trust是默认值。

--auth-host=*authmethod*

这个选项为通过 TCP/IP 连接的本地用户指定在ux_hba.conf中使用的认证方法（host行）。

--auth-local=*authmethod*

这个选项为通过 Unix 域套接字连接的本地用户指定在ux_hba.conf中使用的认证方法（local行）。

-D *directory*
--uxdata=*directory*

这个选项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。

-E *encoding*
--encoding=*encoding*

选择模板数据库的编码。这也将是后来创建的任何数据库的默认编码，除非你覆盖它。默认值来自于区域，或者如果该值不起作用则为SQL_ASCII。

-g
--allow-group-access

允许与集群拥有者同组的用户读取initdb创建的所有集群文件。Windows会忽略此选项，因为它不支持POSIX样式的组权限。

-k
--data-checksums

在数据页面上使用校验码来帮助检测 I/O 系统造成的损坏。启用校验码将会引起显著的性能惩罚。这个选项只能在初始化期间被设置，并且以后不能修改。如果被设置，在所有数据库中会为所有对象计算校验码。

--locale=*locale*

为数据库集群设置默认区域。如果这个选项没有被指定，该区域将从initdb所运行的环境中继承。

--lc-collate=*locale*
--lc-ctype=*locale*
--lc-messages=*locale*
--lc-monetary=*locale*
--lc-numeric=*locale*
--lc-time=*locale*

和--locale相似，但是只在指定的分类中设置区域。

--no-locale

等效于--locale=C。

-N
--no-sync

默认情况下，initdb将等待所有文件被安全地写到磁盘。这个选项会导致initdb不等待就返回，这当然更快，但是也意味着一次后续的操作系统崩溃可能让数据目录损坏。通常，这个选项对测试有用，但是不应该在创建生产安装时使用。

--pwfile=*filename*

让initdb从一个文件读取数据库用户的密码。初始化不带安全功能的实例（不带-a或--security选项），该文件的第一行被当作密码。初始化带安全功能的实例（带-a或--security选项），则文件中至少包含三行有效密码，有效密码是指符合UXDB口令强度限制，详情请参见《优炫数据库安全功能手册 2.1》中“用户标识与鉴别”章节中的“强化口令鉴别”小节。

--running-mode = standard | compatible | mysql

运行模式，默认模式是standard。standard表示标准模式，compatible表示兼容模式，mysql表示mysql模式。

-S
--sync-only

安全地把所有数据库文件写入到磁盘并退出。这不会执行任何正常的initdb操作。

-T *config*
--text-search-config=*config*

设置默认的文本搜索配置。

-U *username*
--username=*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。超级用户的名字是什么真的不重要，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。

-W
--pwprompt

让initdb提示要求操作者为数据库用户设置一个密码，该密码会被程序以MD5默认加密算法加密后存储至系统表ux_authid中的rolepassword字段中。当数据库集群启动后，如果数据库用户请求登录，则服务器程序根据初始化数据库用户创建时保存的密码进行身份鉴别。

-X *directory*
--waldir=*directory*

这个选项指定预写式日志会被存储在哪个目录中。

--ignore-case

该选项只在initdb初始化数据库时设置有效。

在数据库中使用show ignore_case来查询“大小写敏感”的情况，on为元组中数据检索大小的不敏感，off为元组中数据检索大小的敏感。

--wal-segsize=*size*

设置WAL段尺寸，以兆字节为单位。这是WAL日志中每个文件的尺寸。默认的尺寸为16兆字节。该值必须位于2的1次幂和1024次幂（兆字节）之间。这个选项只能在初始化期间设置，并且之后不能更改。

调整这个值来控制WAL日志传送或者归档可能会有用。此外，在有大量WAL的数据库中，每个目录中数量巨大的WAL文件可能会成为性能和管理问题。增加WAL文件尺寸将会降低WAL文件的数量。

其他较少使用的选项：

-d
--debug

打印来自引导后端的调试输出以及普通大众不那么感兴趣的一些消息。引导后端被程序initdb用来创建目录表。这个选项会生成大量极端无聊的输出。

-L *directory*

指定initdb应从哪里寻找它的输入文件来初始化数据库集群。这通常没有必要。如果你需要显式指定它们的位置，你应该被告知。

-n
--no-clean

默认情况下，当initdb确定有一个错误阻止它完整地创建数据库集群，它会移除在它发现无法完成任务之前创建的任何文件。这个选项会抑制这种整理并且对调试有用。

-s
--show

显示内部设置。

-I

初始化数据库级审计，创建数据库审计员uxad。

让initdb提示要求操作者为数据库审计员uxad设置一个密码，该密码会被程序以MD5默认加密算法加密后存储至系统表ux_authid中的rolepassword字段中。当数据库集群以带审计模式启动后，如果审计员uxad请求登录，则服务器程序根据初始化uxad用户创建时保存的密码进行身份鉴别。

-M

启用全数据库级加密，提示用户输入加密秘钥，程序对秘钥采用SHA256算法进行加密，保存至进程中，未在磁盘文件中进行存储。进程退出，此秘钥释放并失效。

--encrypted-pwfile=*filename*

从文件中读取密码，启动加密的数据库。

其他选项：

-V
--version

打印initdb版本并退出。

-?
--help

显示有关initdb命令行参数的帮助并退出。

2.1.4. 环境变量

UXDATA

指定数据库集群应该被存放的目录，可以使用-D选项覆盖。

UX_COLOR

规定在诊断消息中是否使用颜色。可能的值为always、 auto、never。

TZ

指定创建的数据集群的默认时区。值应该是一个完整的时区名称。

和大部分其他UXDB工具相似，这个工具也使用libuxsql支持的环境变量。

2.1.5. 注解

initdb可以通过ux_ctl initdb被调用。

2.2. initdb图形化

initdb图形化 — 与initdb的功能相同，此工具支持initdb的所有参数项

2.2.1. 用法

配置数据库工具图形化工具支持当前配置文件uxsinodb.conf中的所有配置项的修改，且若实例处于启动状态，修改配置信息之后，会提供使其生效的重载或重启功能。

- 环境要求

  要求jdk1.8以上的版本。

- 启动

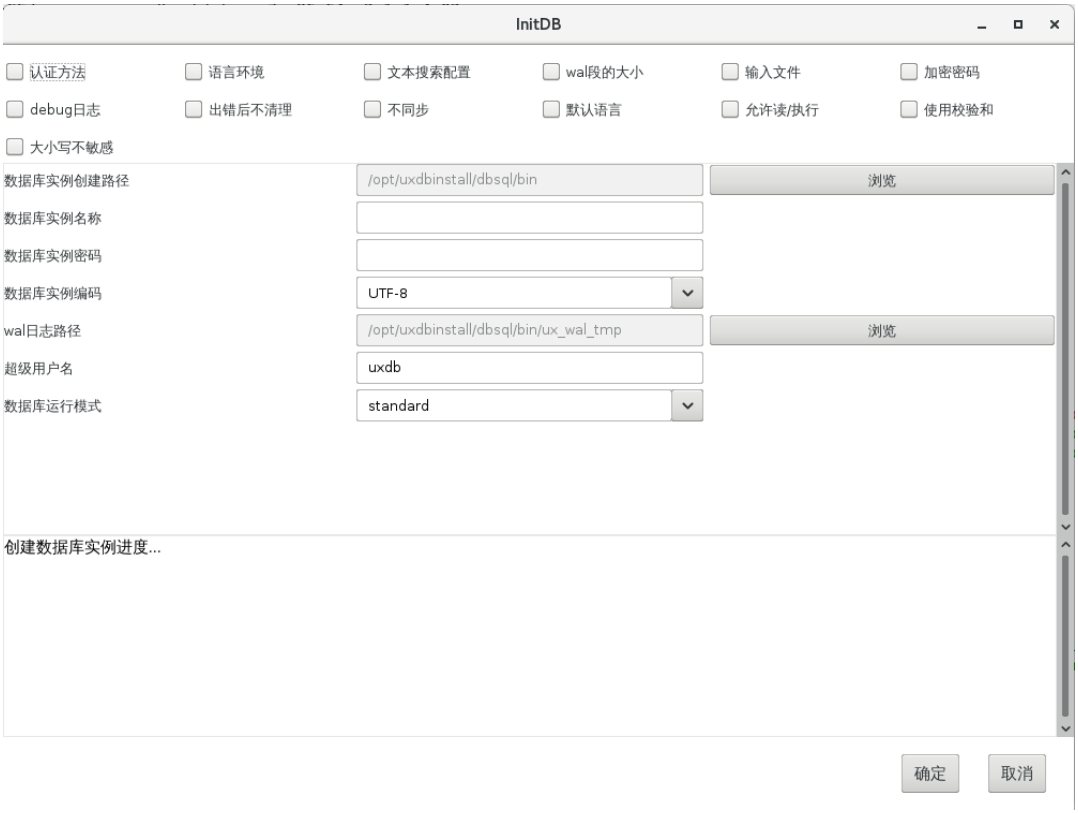
  Linux系统：启动提供的Initdb.sh或ModifyConf.sh脚本。

  Windows系统：启动提供的Initdb.bat或ModifyConf.bat脚本。

2.2.2. 示例

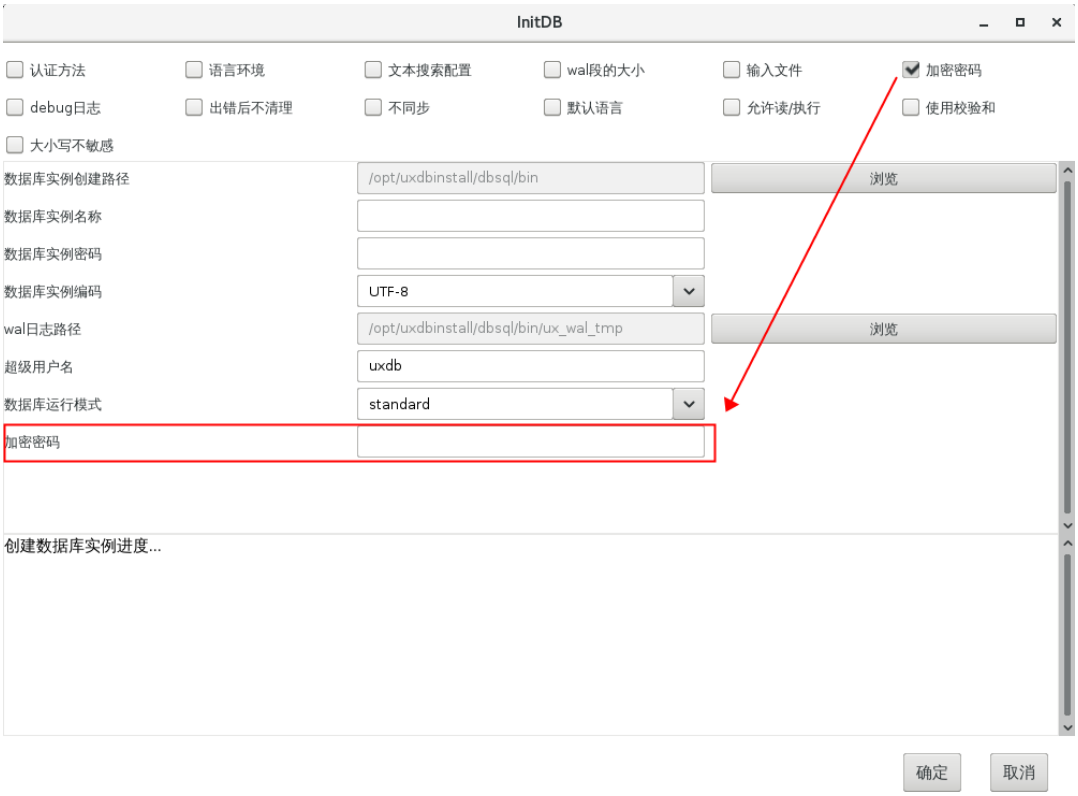
1. 初始化数据库实例

使用脚本Initdb.bat或Initdb.sh启动后，页面分为四个部分，从上到下依次为：参数多选框，当前可配置参数输入框或选择框，初始化实例日志框和确认框。



启动后，用户可勾选配置参数多选框中的参数项，此处展示的是当前版本initdb工具的可选参数。可选参数勾选后，会根据当前参数需要填写的信息，在中间块添加一行/几行参数框，如下图所示：

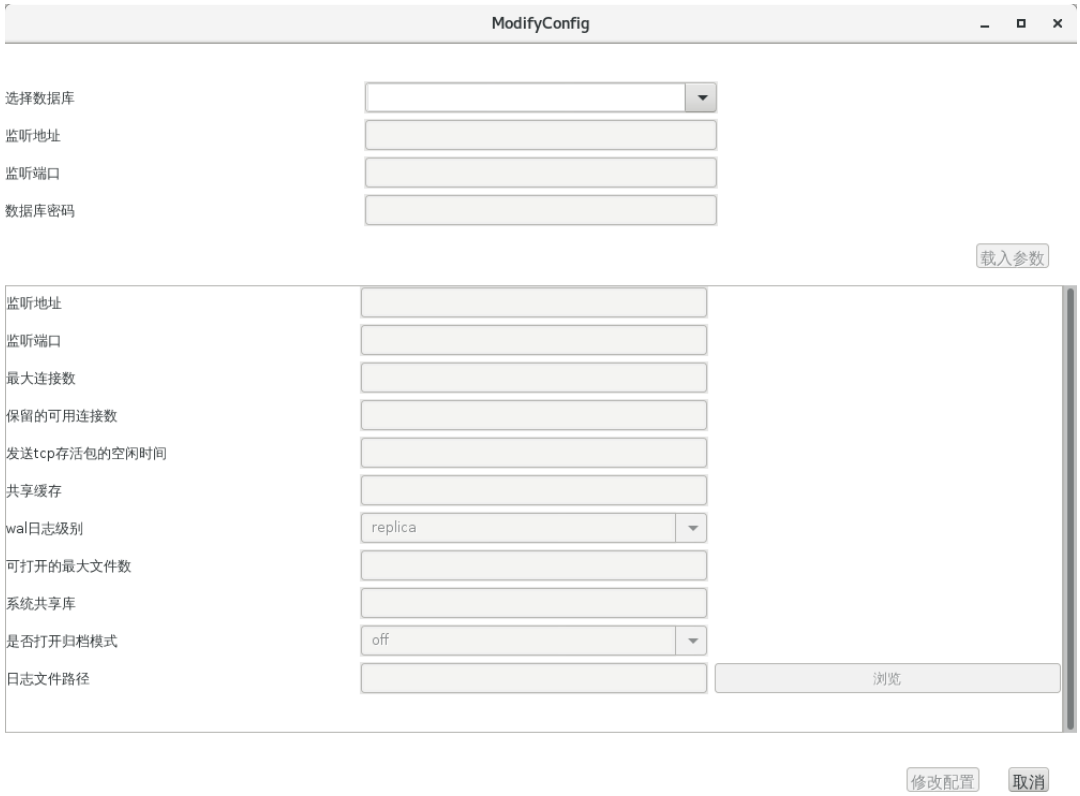
在“加密密码”复选框之后，增加了“加密密码”参数框。



填写完所有信息之后，单击“确定”按钮，即可创建数据库实例。创建完成后，会弹框“数据库实例**创建完成！”

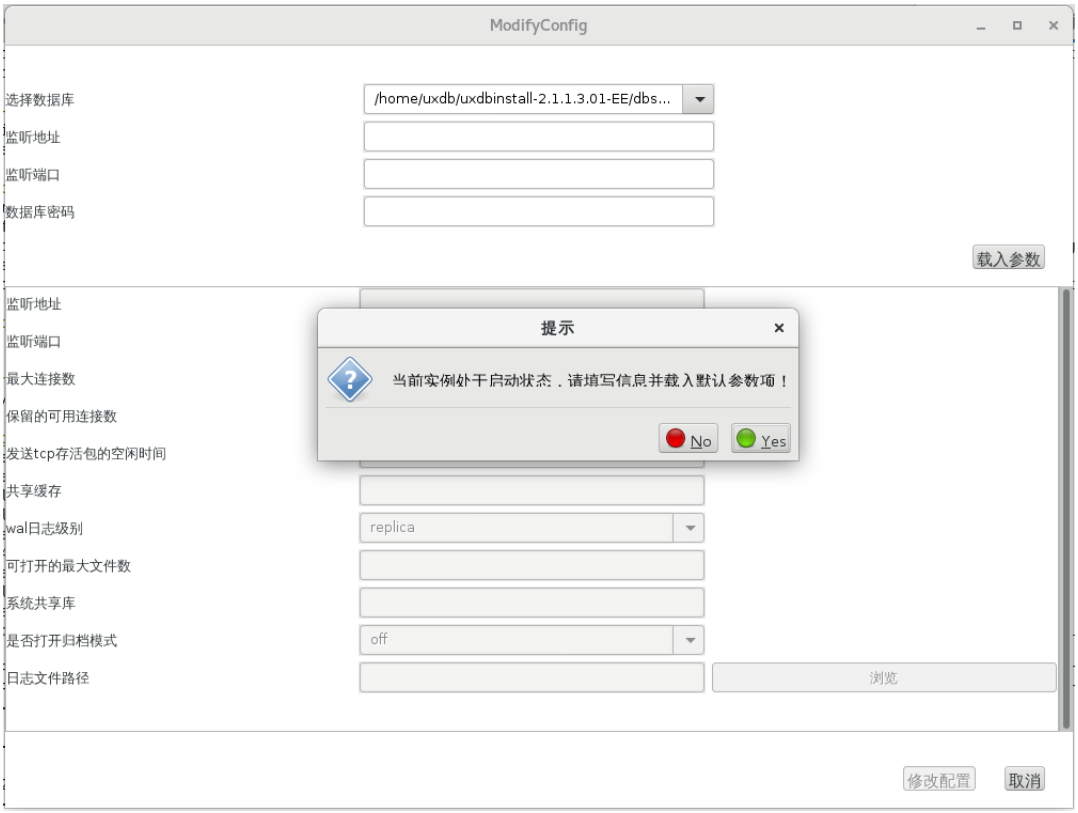
2. 配置数据库

使用脚本ModifyConf.bat或ModifyConf.sh启动后，界面如下图所示。页面分为三个部分，从上到下依次为：数据库信息选择框，当前可配置参数框和确认框。

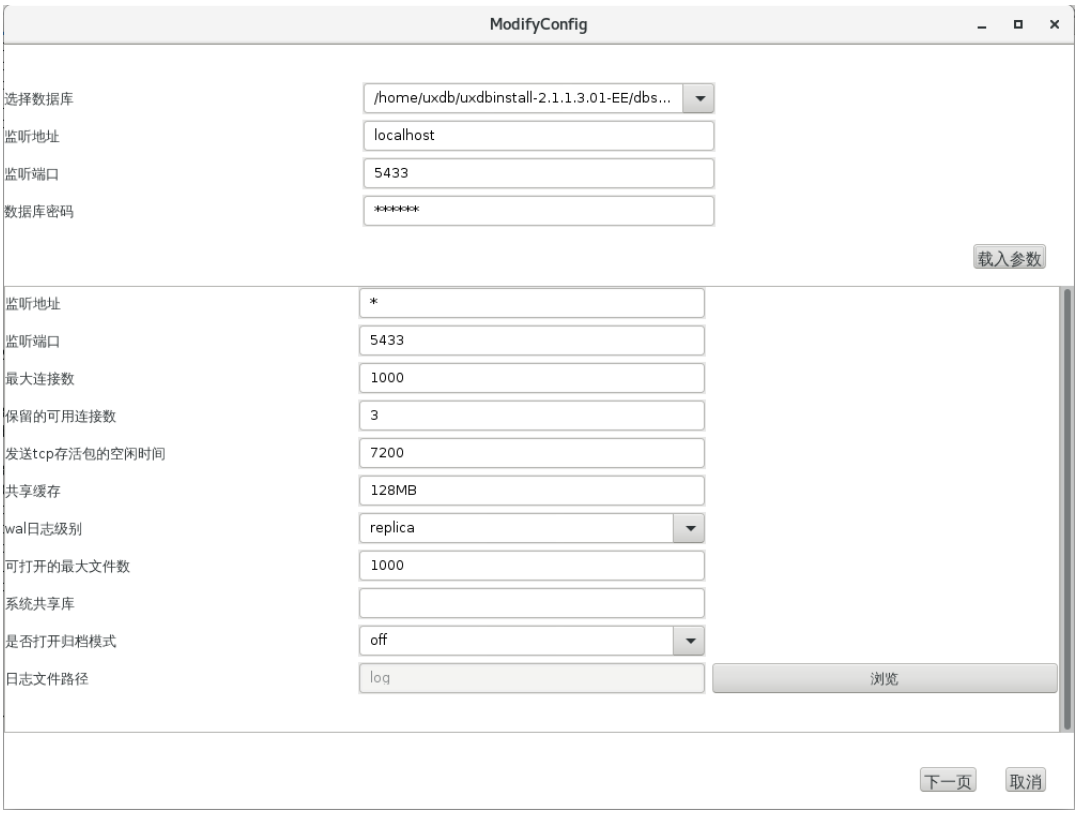


首先，选择一个数据库实例，如果选中的是未启动实例，则参数配置框变为可用状态，并显示当前配置文件中的对应配置值，修改参数值后，单击修改配置按钮。

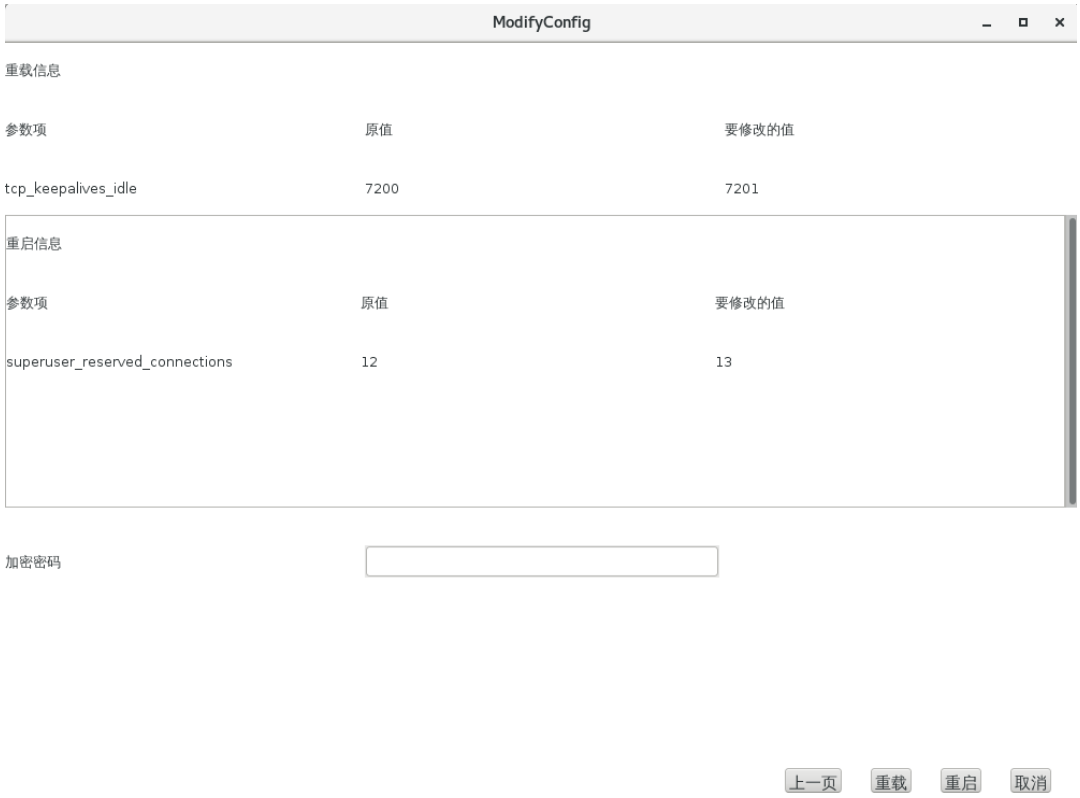
如果选中的是启动中的实例，则启动中实例连接所需信息输入框变为可用状态，如下图所示：



填写监听地址、监听端口、数据库密码后，单击载入参数，当前实例正在使用的相关参数值会载入参数配置框，同时，修改配置按钮变为下一页按钮，如下图所示：



修改参数值后，单击下一页按钮，显示参数配置页面修改的所有参数项，按照重载参数项和重启参数项分类显示，如下图所示：



重要

参数生效的方式有两种：重载或重启数据库实例。

- 重载操作：单击重载按钮，重载信息框的所有信息修改，并重载当前数据库实例使其生效。
- 重启操作：如果是加密实例，需要填写加密密码，然后单击重启按钮，重启信息框的所有信息修改，并重启当前数据库实例使其生效。

注意

当前云实例仅支持未启动且配置文件本地化的情况。

数据库配置支持启动中的云实例。

数据库配置支持配置文件未本地化的云实例。

2.3. removedb

removedb — 移除UXDB数据库集群

2.3.1. 用法

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

2.3.2. 选项

-D *directory*
--uxdata=*directory*

指定数据库集群的数据目录

-F
--force

强制删除给定的数据库集群，谨慎使用这个选项。

-V
--version

打印removedb版本并退出。

-?
--help

显示removedb命令行参数的帮助并退出。

2.3.3. 环境变量

UXDATA

指定数据库集群应该被存放的目录，可以使用-D选项覆盖。

2.3.4. 示例

$ **removedb -D /home/uxdb/uxdbinstall/dbsql/bin/test/**

2.4. ux_archivecleanup

ux_archivecleanup — 清理UXDBWAL归档文件

2.4.1. 用法

ux_archivecleanup [*option*...] *archivelocation oldestkeptwalfile*

2.4.2. 描述

ux_archivecleanup被设计用作archive_cleanup_command在作为备用服务器运行时来清理WAL文件归档。ux_archivecleanup也可以被用作一个单独的程序来清理WAL文件归档。

要配置一个备用服务器以使用ux_archivecleanup，把下面 的内容放在recovery.conf配置文件中：

archive_cleanup_command = 'ux_archivecleanup *archivelocation* %r'

其中*archivelocation*是要从中移除WAL段文件的目录。

当被用在archive-cleanup-command中时，所有逻辑上在 %r参数的值之前的WAL文件都将被从*archivelocation*移除。这能最小化需要被保留的文件数量， 同时能保留崩溃后重启的能力。如果对于这台特定的备用服务器， *archivelocation*是一个短暂需要的区域，使用这个参数就是 合适的，但是当*archivelocation*要用作一个长期的WAL归档 区域或者当多个备用服务器正在从这个归档位置恢复时，使用这个参数就 不合适。

当被用作一个单独的程序时，所有逻辑上在*oldestkeptwalfile* 之前的WAL文件将被从*archivelocation*中移除。在这种模式 中，如果指定了.partial或者.backup文件名，则 只有该文件前缀将被用作*oldestkeptwalfile*。这种对 .backup文件名的处理允许你移除所有在一个特定基础备份之前归 档的WAL文件而不出错。例如，下面的例子将移除所有比WAL文件名000000010000003700000010老的文件：

ux_archivecleanup -d archive 000000010000003700000010.00000020.backup

```
ux_archivecleanup:  keep WAL file "archive/000000010000003700000010" and later
ux_archivecleanup:  removing file "archive/00000001000000370000000F"
ux_archivecleanup:  removing file "archive/00000001000000370000000E"
```

ux_archivecleanup假定*archivelocation*是一个可读的目录并且对于服务器拥有者是可写的。

2.4.3. 选项

ux_archivecleanup接受下列命令行参数：

-d

在stderr上打印很多调试日志输出。

-n

在stdout上打印将被移除的文件的名字（执行一次演习）。

-V
--version

打印ux_archivecleanup版本并退出。

-x *extension*

提供一个扩展名，在决定所有的文件 是否应该被删除之前，将从文件名中剥离这个扩展名。这通常有助于清理已经 存储期间被压缩过并且被压缩程序增加了一个扩展名的归档。例如： -x .gz。

-?
--help

显示ux_archivecleanup命令行参数的帮助并退出。

2.4.4. 注解

ux_archivecleanup以 C 写成并且具有很容易修改的 源代码，其中有特别指定的区域用于修改以符合你的需要。

2.4.5. 示例

在 Linux 或者 Unix 系统上，你可能会用：

archive_cleanup_command = 'ux_archivecleanup -d /mnt/standby/archive %r 2>>cleanup.log'

其中归档目录位于备用服务器上，这样archive_command通过 NFS 来访问它，但是文件对于备用服务器来说是本地的。这将会：

- 在cleanup.log中产生调试输出；
- 从归档目录中移除不再需要的文件。

2.5. ux_checksums

ux_checksums - 在UXDB数据库集群中启用、禁用或检查数据校验和

2.5.1. 用法

ux_checksums [*option...*] [[-D | --uxdata] *datadir*]

2.5.2. 描述

ux_checksums在UXDB集群中检查、启用或禁用数据校验和。运行ux_checksums之前，必须彻底关闭服务器。验证校验和时，如果没有校验和错误，则退出状态为零，如果检测到至少一个校验和失败，则退出状态为非零。启用或禁用校验和时，如果操作失败，则退出状态为非零。

验证校验和时，集群中的每个文件都要被扫描。启用校验和时，集群中的每个文件都会被重写。禁用校验和时，仅更新ux_control文件。

2.5.3. 选项

ux_checksums接受下列命令行参数：

-D *directory*
--uxdata=*directory*

指定存储数据库集群的目录。

-c
--check

检查校验和。如果未指定其它任何内容，这是缺省模式。

-d
--disable

禁用校验和。

-e
--enable

启用校验和。

-f *filenode*
--filenode=*filenode*

仅验证文件节点为*filenode*的关系中的校验和。

-N
--no-sync

缺省地，ux_checksums会等待所有文件安全地写到磁盘上。该选项使得ux_checksums不等待就返回，这样更快，但意味着后续如果操作系统崩溃会让更新的数据目录损坏。一般地，该选项对测试有用，但不应用在生产安装上。当使用--check时，该选项无效。

-P
--progress

启用进度报告。在检查或启用校验和时，打开该选项，会提供进度报告。

-v
--verbose

启用详细输出。列出所有检查的文件 。

-V
--version

打印ux_checksums版本并退出。

-?
--help

显示关于ux_checksums命令行参数的帮助并退出。

2.5.4. 环境变量

UXDATA

指定数据库集群存储的目录；可以用-D选项覆盖。

UX_COLOR

指定是否在诊断消息中使用颜色。可能的值为always，auto，never。

2.5.5. 注意

在大型集群中启用校验和的时间可能很长。在此操作期间，写到数据目录的集群或其它程序必须是未启动的，否则可能出现数据丢失。

当复制设置与执行关系文件块的直接拷贝的工具（例如第 2.10 节 “ux_rewind”一起使用时，启用和禁用校验和会导致以不正确校验和形式出现的页面损坏，如果未在所有节点上执行一

致的操作的话。故在复制设置中启用或禁用校验和时，推荐一致地切换所有集簇之前停止所有集群。此外销毁所有备用数据库，在主数据库上执行操作，最后从头开始重建备用服务器，也是安全的。

如果在启用或禁用校验和时异常终止或杀掉ux_checksums，那么集群的数据校验和配置保持不变，ux_checksums可以重新运行以执行相同操作。

2.6. ux_controldata

ux_controldata — 显示一个UXDB数据库集群的控制信息

2.6.1. 用法

ux_controldata［*option*］［[-D］ *datadir*］

2.6.2. 描述

ux_controldata打印在initdb期间初始化的信息，例如目录版本。它也显示关于预写式日志和检查点处理的信息。这种信息是集群范围的，并且不针对任何一个数据库。

这个工具只能由初始化集群的用户运行，因为它要求对数据目录的读访问。你可以在命令行中指定数据目录，或者使用环境变量UXDATA。这个工具支持选项-V和--version，它们打印ux_controldata版本并退出。它也支持选项-?和--help，它们输出支持的参数。

2.6.3. 选项

ux_controldata接受下列命令行参数：

-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。

-V
--version

打印ux_controldata版本并退出。

-?
--help

显示ux_controldata命令行参数的帮助并退出。

2.6.4. 环境变量

UXDATA

默认的数据目录位置。

UX_COLOR

规定在诊断消息中是否使用颜色。可能的值为always、auto、never。

2.7. ux_ctl

ux_ctl — 初始化、启动、停止或控制一个UXDB服务器

2.7.1. 用法

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

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

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

ux_ctl restart [-D *datadir*] [-m s[mart] | f[ast] | i[mmediate]] [-W] [-t *seconds*] [-s] [-o *options*] [-c]

ux_ctl reload [-D *datadir*] [-s]

ux_ctl status [-D *datadir*]

ux_ctl promote [-D *datadir*] [-W] [-t *seconds*] [-s]

ux_ctl logrotate [-D *datadir*] [-s]

ux_ctl kill *signal_name process_id*

在Microsoft Windows上，还有：

ux_ctl register [-D *datadir*] [-N *servicename*] [-U *username*] [-P *password*] [-S a[uto] | d[emand]] [-e *source*] [-W] [-t *seconds*] [-s] [-o *options*]

ux_ctl unregister [-N *servicename*]

2.7.2. 描述

ux_ctl是一个用于初始化UXDB数据库集群，启动、停止或重启UXDB数据库服务器（uxdb），或者显示一个正在运行服务器的状态的工具。尽管服务器可以被手工启动，ux_ctl包装了重定向日志输出以及正确地从终端和进程组脱离等任务。它也提供了方便的选项用来控制关闭。

init或initdb模式会创建一个新的UXDB数据库集群，也就是将由一个单一服务器实例管理的数据库集合。这个模式调用initdb命令。详见initdb。

start模式启动一个新的服务器。该服务器被启动在后台，并且它的标准输出被附加到/dev/null（或Windows上的nul）。在 Unix 类系统上，默认情况下服务器的标准输出和标准错误被发送到ux_ctl的标准输出（不是标准错误）。ux_ctl的标准输出应该接着被重定向到一个文件或用管道导向另一个进程（例如日志轮转程序rotatelogs）。否则uxdb将把它的输出写到控制终端（从后台）并且将不会离开 shell 的进程组。在 Windows 上，默认情况下服务器的标准输出和标准错误被发送到终端。这些默认行为可以使用-l追加服务器的输出到一个日志文件来改变。我们推荐使用-l或输出重定向。

stop模式关闭运行在指定数据目录中的服务器。对-m选项可以选择三种不同的关闭方法。“Smart”模式等待所有客户端断开连接以及任何在线备份结束。如果该服务器是热备，一旦所有的客户端已经断开连接，恢复和流复制将被终止。“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。“Immediate”模式将立刻中止所有服务器进程，而不是做一次干净的关闭。这中选择将导致下一次重启时进行一次崩溃恢复。

restart模式实际上会先执行一个停止操作然后紧接着执行一个启动操作。这使得我们能够更改uxdb的命令行选项，或者更改不通过重启服务器无法更改的配置文件选项。如果在服务器启动期间在命令行上使用了相对路径，则restart可能会失败，除非ux_ctl被运行在与上次启动服务器相同的目录中。

reload模式简单地向uxdb服务器进程发送一个SIGHUP信号，导致它重新读取它的配置文件（uxsinodb.conf、ux_hba.conf等）。这允许改变配置文件选项而无需一次完整的服务器重启来让改变生效。

status模式检查一个服务器是否运行在指定的数据目录中。如果有一个服务器正在运行，其PID和用来调用它的命令行选项将被显示。如果服务器没有在运行，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。

promote模式命令运行在指定数据目录中的备用服务器结束备用模式并且开始读写操作。

logrotate模式轮换服务器日志文件。

kill模式向一个指定进程发送一个消息。这主要用于没有kill命令的Microsoft Windows。使用--help来查看受支持的信号名称列表。

register模式把UXDB服务器注册为Microsoft Windows上的一个系统服务。-S选项允许选择服务启动类型，可以是“auto”（随系统自动启动）或“demand”（按需启动）。

unregister模式在Microsoft Windows上移除一个系统服务的注册。这会撤销register命令的效果。

2.7.3. 选项

-c
--core-files

在可行的平台上尝试允许服务器崩溃产生核心文件，方法是提升在核心文件上的任何软性资源限制。这通过允许从一个失败的服务器进程中获得一个栈跟踪而有助于调试或诊断问题。

-D *datadir*
--uxdata=*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*
--log=*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。

-m *mode*
--mode=*mode*

指定关闭模式。*mode*可以是smart、fast或immediate，或者这三者之一的第一个字母。如果这个选项被忽略，则fast是默认值。

smart：所有客户端断开连接后退出；fast：直接退出并关机；immediate：不完全的关闭退出，重启后恢复。

-o *options*
--options=*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过去。

这些选项应该通常被单引号或双引号包围来确保它们被作为一个组传递。

-o *initdb-options*
--options=*initdb-options*

指定要直接传递给initdb命令的选项。-o可以被指定多次，所有给定的选项都会被传过去。

这些选项应该通常被单引号或双引号包围来确保它们被作为一个组传递。

-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。

-s
--silent

只打印错误，不打印信息性的消息。

-t *seconds*
--timeout=*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。

-V
--version

打印ux_ctl版本并退出。

-w
--wait

等待操作完成。模式start、stop、restart、promote以及register支持这个选项，并且对那些模式是默认的。

在等待时，ux_ctl会一遍又一遍地检查服务器的PID文件，在两次检查之间会休眠一小段时间。当PID文件指示该服务器已经做好准备接受连接时，启动操作被认为完成。当服务器移除PID文件时，关闭操作被认为完成。ux_ctl会基于启动或关闭的成功与否返回一个退出代码。

如果操作在超时时间（见选项-t）内未能完成，则ux_ctl会以一个非零退出状态退出。但是注意该操作可能会在后台继续进行并且最终取得成功。

-W
--no-wait

不等待操作完成。这是选项-w的对立面。

如果禁用等待，所请求的动作会被触发，但是不会有关于其成功与否的反馈。在这种情况下，可能必须用服务器日志文件或外部监控系统来检查该操作的进度以及成功与否。

-M

启用全数据库级加密，提示用户输入加密秘钥，程序对秘钥采用SHA256算法进行加密，保存至进程中，未在磁盘文件中进行存储。进程退出，此秘钥释放并失效。

-Y
--encrypted-pwfile=*filename*

从文件中读取密码，启动加密的数据库。

-?
--help

显示有关ux_ctl命令行参数的帮助并退出。

如果一个指定的选项有效，但与选中的操作模式无关，则ux_ctl会忽略它。

2.7.3.1. 用于 Windows 的选项

-e *source*

作为一个 Windows 服务运行时，ux_ctl用来 在事件日志中记录日志的事件源的名称。默认是UXDB。 注意这只控制由ux_ctl本身发送的消息，一旦开始， 服务器将使用event-source参数中指定的事件源。如果服务器在启动时很早（在该参数被设置前）就失败，那么在该参数设置之前，它可能也会使用默认的事件源名称UXDB来记录。 UXDB来记录。

-N *servicename*

要注册的系统服务的名称。这个名称将被用于服务名和显示名。默认是UXDB。

-P *password*

用于运行该服务的用户的密码。

-S *start-type*

要注册的系统服务的启动类型。启动类型可以是auto、demand或者两者之一的第一个字母。如果这个选项被忽略，则auto是默认值。

-U *username*

用于运行该服务的用户的用户名。对于域用户，使用格式DOMAIN\username。

2.7.4. 环境变量

UXCTLTIMEOUT

等待启动或者关闭完成时要等待的默认秒数限制。如果没有设置， 默认值是60秒。

UXDATA

默认的数据目录位置。

大部分的ux_ctl模式都要求知道数据目录的位置，因此-D选项是必需的，除非UXDATA被设置。

和大部分其他UXDB工具相似，ux_ctl也使用libuxsql支持的环境变量。

2.7.5. 文件

uxmaster.pid

ux_ctl在数据目录中检查这个文件来判断服务器当前是否正在运行。

uxmaster.opts

如果这个文件存在于数据目录中，ux_ctl（处于restart模式中）将把该文件的内容作为选项传递给uxdb，除非通过-o选项进行了覆盖。这个文件的内容也会被显示在status模式中。

2.7.6. 示例

2.7.6.1. 启动服务器

要启动服务器并且等到服务器接受连接：

$ **ux_ctl start**

要使用端口5433启动服务器并且运行时不使用fsync：

$ **ux_ctl -o "-F -p 5433" start**

2.7.6.2. 停止服务器

要停止服务器，使用：

$ **ux_ctl stop**

-m选项允许控制服务器如何关闭：

$ **ux_ctl stop -m smart**

2.7.6.3. 重启服务器

重启服务器几乎等价于停止服务器并且再次启动它，不过ux_ctl默认会保存并重用被传递给之前的运行实例的命令行选项。要以和之前相同的选项重启服务器，使用：

$ **ux_ctl restart**

但是如果指定了-o，则会替换任何之前的选项。要使用端口5433重启并在重启时禁用fsync：

$ **ux_ctl -o "-F -p 5433" restart**

2.7.6.4. 显示服务器状态

这里是ux_ctl状态输出的例子：

$ **ux_ctl status**

ux_ctl: server is running (PID: 13718)
/home/uxdb/uxdbinstall/dbsql/bin/uxdb "-D" " /home/uxdb/uxdbinstall/dbsql/data" "-p" "5433" "-B" "128"

第二行是在重启模式可能被调用的命令行。

2.8. ux_diagnose

ux_diagnose — 分析和重置UXDB数据库集群的预写式日志

2.8.1. 用法

ux_diagnose［*option*］[［-D］ *datadir*］

2.8.2. 选项

-D *directory*

指定数据库集群的数据目录

-m *mxid*,*mxid*

手工设置下一个和最老的多事务ID。

确定下一个多事务ID（第一部分）的安全值的方法：在数据目录下的ux_multixact/offsets目录中查找最大的数字文件名，然后在它的基础上加一并且乘以 65536 (0x10000)。确定最老的多事务ID（第二部分）的方法：在同一个目录中查找最小的数字文件名并且乘以65536。文件名是十六进制的数字，因此实现上述方法最简单的方式是以十六进制指定选项值并且追加四个零。

-O *mxoff*

手工设置下一个多事务偏移量。

查找数据目录下ux_multixact/members目录中最大的数字文件名，然后在它的基础上加一并且乘以52352 (0xCC80)。文件名是十六进制数字。没有像其他选项那样追加零的简单方法。

-x *xid*

手工设置下一个事务 ID。

在数据目录下的ux_xact目录中查找最大的数字文件名， 然后在它的基础上加一并且乘以1048576 (0x100000)。注意文件名是十六进制的数字。通常以十六进制的形式指定该选项值

也是最容易的。例如，如果0011是ux_xact中的最大项，-x 0x1200000就可以 （五个尾部的零就表示了前面说的乘数）。

-p *port*
--port=*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*
--username=*username*

要作为哪个用户连接。

-V
--version

打印版本信息并退出。

-?
--help

显示帮助并退出。

2.8.3. 示例

$ **./ux_diagnose -D /home/uxdb/uxdbinstall/dbsql/bin/test**

2.9. ux_resetwal

ux_resetwal — 重置一个UXDB数据库集群的预写式日志以及其他控制信息

2.9.1. 用法

ux_resetwal [--force | -f] [--dry-run | -n] [*option...*] [--uxdata | -D] *datadir*

2.9.2. 描述

ux_resetwal会清除预写式日志（WAL）并且有选择地重置存储在ux_control文件中的一些其他控制信息。如果这些文件已经被损坏，某些时候就需要这个功能。当服务器由于这样的损坏而无法启动时，这只应该被用作最后的手段。

在运行这个命令之后，就可能可以启动服务器，但是记住数据库可能包含由于部分提交事务产生的不一致数据。你应当立刻转储你的数据、运行initdb并且重新载入。重新载入后，检查不一致并且根据需要修复。

这个工具只能被安装服务器的用户运行，因为它要求对数据目录的读写访问。出于安全原因，你必须在命令行中指定数据目录。ux_resetwal不使用环境变量UXDATA。

如果ux_resetwal因为无法为ux_control确定有效数据暂停，可以通过指定-f（强制）选项强制它继续。在这种情况下，丢失的数据将被替换为看似合理的值。可以期望大部分域是匹配的，但是

下一个 OID、下一个事务 ID 和纪元、下一个多事务 ID 和偏移以及WAL开始位置域可能还是需要人工协助。这些域可以使用下面讨论的选项设置。如果你不能为所有这些域决定正确的值，-f还是可以被使用，但是恢复的数据库还是值得怀疑：一次立即的转储和重新载入是势在必行的。在你转储之前不要在该数据库中执行任何数据修改操作，因为任何这样的动作都可能使破坏更严重。

2.9.3. 选项

-f
--force

即使ux_resetwal无法从ux_control中确定有效的数据（如前面所解释的），也强迫ux_resetwal继续运行。

-n
--dry-run

-n/--dry-run选项指示ux_resetwal打印从ux_control重构出来的值以及要被改变的值，然后不修改任何东西退出。这主要是一个调试工具，但是可以用来在允许ux_resetwal真正执行下去之前进行完整性检查。

-D *directory*

指定数据库集群的数据目录

-f

即使ux_resetwal无法从ux_control中确定有效的数据（如前面所解释的），也强迫ux_resetwal继续运行。

-n

-n（无操作）选项指示ux_resetwal打印从ux_control重构出来的值以及要被改变的值，然后不修改任何东西退出。这主要是一个调试工具，但是可以用来在允许ux_resetwal真正执行下去之前进行完整性检查。

-V
--version

显示版本信息然后退出。

-?
--help

显示帮助然后退出。

只有当ux_resetwal无法通过读取ux_control确定合适的值时，才需要下列选项。安全值可以按下文所述来确定。对于接收数字参数的值，可以使用前缀0x指定16进制值。

-c *xid*,*xid*
--commit-timestamp-ids=*xid*,*xid*

手工设置提交时间可以检索到的最老的和最新的事务ID。

能检索到提交时间的最老事务ID的安全值（第一部分）可以通过在数据目录下ux_commit_ts目录中数字上最小的文件名来决定。反过来，能检索到提交时间的最新事务ID的安全值（第二部分）可以通过同一个目录中数字上最大的文件名来决定。文件名都是十六进制的。

-e *xid_epoch*
--epoch=*xid_epoch*

手工设置下一个事务ID的epoch。

事务ID的epoch实际上并没有存储在数据库中的任何地方，除了被ux_resetwal设置在这个域中，所以只要关心的是数据库本身，任何值都可以用。你可能需要调整这个值来确保诸如Slony-I和Skytools之类的复制系统正确地工作 — 如果确实需要调整，应该可以从下游的复制数据库的状态中获得一个合适的值。

-l *walfile*
--next-wal-file=*walfile*

手工设置WAL开始地址。

WAL起始地址应该比当前存在于数据目录下ux_wal目录中的任意 WAL 段文件名更大。这些名称也是十六进制的并且有三个部分。第一部分是“时间线ID”并且通常应该被保持相同。例如，如果000000010000000320000004A是ux_wal中最大的项，则使用-l 000000010000000320000004B或更高的值。

注意在使用非默认WAL段尺寸时，WAL文件名中的数字与系统函数和系统视图报告的LSN不同。这个选项要的是WAL文件名而不是LSN。

> **注意**
>
> ux_resetwal通常会查看ux_wal中的文件并选择一个超出最新现存文件名的默认值-l进行设置。因此，只有当你知道WAL段文件当前不在ux_wal中时，或者当ux_wal的内容完全丢失时，才需要对-l进行手工调整，例如一个离线归档中的项。

-m *mxid*,*mxid*
--multixact-ids=*mxid*,*mxid*

手工设置下一个和最老的多事务ID。

确定下一个多事务ID（第一部分）的安全值的方法：在数据目录下的ux_multixact/offsets目录中查找最大的数字文件名，然后在它的基础上加一并且乘以65536(0x10000)。反过来，确定最老的多事务ID（-m的第二部分）的方法：在同一个目录中查找最小的数字文件名并且乘以65536。文件名是十六进制的数字，因此实现上述方法最简单的方式是以十六进制指定选项值并且追加四个零。

-o *oid*
--next-oid=*oid*

手工设置下一个OID。

决定超过数据库中最大OID的下一个OID没有相对容易的方法。但幸运的是正确地得到下一个OID设置并不是决定性的。

-O *mxoff*
--multixact-offset=*mxoff*

手工设置下一个多事务偏移量。

查找数据目录下ux_multixact/members目录中最大的数字文件名，然后在它的基础上加一并且乘以52352(0xCC80)。文件名是十六进制数字。没有像其他选项那样追加零的简单方法。

--wal-segsize=*wal_segment_size*

设置新的WAL段尺寸，以兆字节为单位。这个值必须被设为2的1次幂和1024次幂（兆字节）之间。更多信息请参考第 2.1 节 “initdb”的相同选项。

> **注意**
>
> 虽然ux_resetwal将把WAL起始地址设置成超过最新的现有WAL段文件，但一些段尺寸的改变可能导致之前的WAL文件名被重用。如果WAL文件名重叠会导致归档策略出现问题，推荐把-l和这个选项一起使用来手动设置WAL起始地址。

-x *xid*
--next-transaction-id=*xid*

手工设置下一个事务ID。

在数据目录下的ux_xact目录中查找最大的数字文件名，然后在它的基础上加一并且乘以1048576(0x100000)。注意文件名是十六进制的数字。通常以十六进制的形式指定该选项值也是最容易的。例如，如果0011是ux_xact中的最大项，-x 0x1200000就可以（五个尾部的零就表示了前面说的乘数）。

2.9.4. 环境变量

UX_COLOR

规定在诊断消息中是否使用颜色。可能的值为always、auto、never。

2.9.5. 注解

这个命令不能在服务器正在运行时被使用。如果在数据目录中发现一个服务器锁文件，ux_resetwal将拒绝启动。如果服务器崩溃那么一个锁文件可能会被留下，在那种情况下你能移除该锁文件来让ux_resetwal运行。但是在你那样做之前，再次确认没有服务器进程仍然存活。

ux_resetwal仅能在具有相同主版本的服务器上使用。

2.10. ux_rewind

ux_rewind — 把一个UXDB数据目录与另一个从它复制出来的数据目录同步

2.10.1. 用法

ux_rewind [*option*...] { -D | --target-uxdata } *directory* { --source-uxdata=*directory* | --source-server=*connstr* }

2.10.2. 描述

ux_rewind是用于在集群的时间线分叉以后，同步一个UXDB集群和同一集群的另一份拷贝的工具。一种典型的场景是在失效后让一个旧的主服务器重新上线，同时有一个备用机跟随着新的主机。

其结果等效于把目标数据目录替换成源数据目录。数据文件中只有更改过的块才会被拷贝，所有其他的文件会被整个拷贝，包括配置文件。ux_rewind比起做一个新的基础备份或者rsync等工具的优势在于，ux_rewind不要求通读集群中未更改的块。这使得它在数据库很大并且在集群间只有小部分块不同时速度很快。

ux_rewind检查源集群和目标集群的时间线历史来判断它们在哪一点分叉，并且期望在目标集群的ux_wal目录中找到WAL来返回到分叉点。分叉点可能会在目标时间线、源时间线或者它们的共同祖先上找到。在典型的失效场景中，目标集群在分叉后很快就被关闭，这不是问题，但是如果目标集群在分叉后已经运行了很长时间，旧的WAL文件可能已经不存在了。在这样的情况下，它们可以被手工从WAL归档复制到ux_wal目录，或者通过配置recovery.conf在启动时取得。ux_rewind的使用并不限于失效的场景，例如一个备用服务器可能被提升、运行一些写事务，然后被倒回再次成为一个备用。

当目标服务器在运行了ux_rewind之后第一次启动时，它将进入到恢复模式并且重放源服务器在分叉点之后产生的所有WAL。当ux_rewind被运行时有某些WAL在源服务器上不可用，并且因此无法被ux_rewind会话所复制，则在目标服务器被启动时必须让这些 WAL 可用。这可以通过在目标数据目录中创建一个recovery.conf文件并且在其中使用一个适当的restore_command来实现。

ux_rewind要求目标服务器在uxsinodb.conf中启用了wal_log_hints选项，或者在用initdb初始化集群时启用了数据校验。目前默认情况下这两者都没有被打开。full-page-writes也必须被设置为on，这是默认的。

> **警告**
>
> 如果在处理时ux_rewind失败，则目标的数据目录很可能不在可恢复的状态。在这种情况下，推荐创建一个新的备份。
>
> 如果ux_rewind发现它无法直接写入的文件，它将立刻失败。例如当源服务器和目标服务器为只读的SSL密钥及证书使用相同的文件映射，就会发生这种情况。如果在目标服务器上存在这样的文件，推荐在运行ux_rewind之前移除它们。在做了rewind之后，一些那样的文件可能已经被从源服务器拷贝，这样就有必要移除已经拷贝的数据并且恢复到rewind之前使用的链接集合。

2.10.3. 选项

ux_rewind接受下列命令行参数：

-D *directory*
--target-uxdata=*directory*

这个选项指定要与源数据目录同步的目标数据目录。在运行ux_rewind之前目标服务器必须被干净地关闭。

--source-uxdata=*directory*

指定要和目标服务器同步的源服务器的数据目录的文件系统路径。这个选项要求源服务器必须被干净地关闭。

--source-server=*connstr*

指定一个libuxsql连接串用于连接要与目标服务器同步的源UXDB服务器。该连接必须是一个具有超级用户访问权限的普通（非复制）连接。这个选项要求源服务器正在运行且不处于恢复模式。

-n
--dry-run

做除了实际修改目标目录之外的其他所有事情。

-N
--no-sync

默认情况下，ux_rewind将等待所有文件安全地写入磁盘。 此选项会导致ux_rewind不等待即可返回，这更快，但意味着后续操作系统崩溃会使同步数据目录损坏。通常情况，此选项可用于测试，但在创建生产安装时不应使用。

-P
--progress

启用进度报告。在从源集群拷贝数据时，打开这个选项将会发送一个近似的进度报告。

--debug

打印详细的调试输出，这主要对于调试ux_rewind的开发者有用。

-V
--version

显示版本信息然后退出。

-?
--help

显示帮助然后退出。

2.10.4. 环境变量

UX_COLOR

规定在诊断消息中是否使用颜色。可能的值为always、auto、never。

在使用--source-server选项时，ux_rewind也使用libuxsql支持的环境变量。

2.10.5. 注解

当使用在线集群作为源执行ux_rewind时，具有充足权限来执行ux_rewind在源集群上使用的函数的角色可以用来代替超级用户。 这里介绍如何创建这样的角色，在这里命名rewind_user：

```
CREATE USER rewind_user LOGIN;
GRANT EXECUTE ON function ux_catalog.ux_ls_dir(text, boolean, boolean) TO rewind_user;
GRANT EXECUTE ON function ux_catalog.ux_stat_file(text, boolean) TO rewind_user;
GRANT EXECUTE ON function ux_catalog.ux_read_binary_file(text) TO rewind_user;
```

GRANT EXECUTE ON function ux_catalog.ux_read_binary_file(text, bigint, bigint, boolean) TO rewind_user;

当使用近期升级的在线集群作为源执行ux_rewind时，必须在升级后执行CHECKPOINT以便其控制文件反映最新的时间线信息， ux_rewind使用这些信息检查目标集群是否可以使用指定的源集群倒回。

2.10.5.1. 如何工作

其基本思想是从源集群拷贝所有文件系统级别的改变到目标集群：

1. 以源集群的时间线历史从目标集群分叉出来的点之前的最后一个检查点为起点，扫描目标集群的WAL日志。对于每一个WAL记录，读取每一个被动过的数据块。这会得到在目标集群中从源集群被分支出去以后所有被更改过的数据块列表。

2. 使用直接的文件系统访问（--source-uxdata）或者SQL（--source-server），把所有那些更改过的块从源集群拷贝到目标集群。

3. 把其他文件（除了关系文件之外的所有文件，例如ux_xact和配置文件）从源集群拷贝到目标集群。与基础备份类似，在从源集簇拷贝的数据中，目录ux_dynshmem/、ux_notify/、ux_replslot/、ux_serial/、ux_snapshots/、ux_stat_tmp/以及ux_subtrans/的内容会被忽略。任何以uxsql_tmp开始的文件或目录都会被忽略，backup_label、tablespace_map、ux_internal.init、uxmaster.opts以及uxmaster.pid也是这样。

4. 从源集群应用WAL，从失效处创建的检查点开始（严格来说，ux_rewind并不应用WAL，它只是创建一个备份标签文件，该文件让UXDB从那个检查点开始向前重放所有WAL）。

2.11. ux_standby

ux_standby — 对创建一个UXDB热备服务器提供支持

2.11.1. 用法

ux_standby [*option...*] *archivelocation nextwalfile walfilepath* [*restartwalfile*]

2.11.2. 描述

ux_standby用于支持创建一个“热备”数据库服务器。它被设计为一个可以随时投入生产的程序，以及一个可定制的模板供你进行特定的修改。

ux_standby被设计成一个等待着的restore_command，它被用来把一次标准的归档恢复变成一次热备操作。还需要一些其他的配置，所有这些配置都在主服务器手册中有相应的描述。

要配置一台备用服务器去使用ux_standby，可以把下面的内容放在recovery.conf配置文件中：

restore_command = 'ux_standby *archiveDir* %f %p %r'

其中*archiveDir*是 WAL 段文件应该被存储的目录。

如果指定了*restartwalfile*（通常用%r宏指定），那么所有在逻辑上位于这个文件之前的 WAL 文件都将被从*archivelocation*中移除。这使需要被保留的文件数最小化，与此同时能够保持崩溃重启

的能力。如果*archivelocation*对于这个特定备用服务器是一个临时暂存区域，使用这个参数是合适的，但当*archivelocation*是一个长期 WAL 归档区域时则不是合适的。

ux_standby假定*archivelocation*是一个拥有服务器的用户可读的目录。如果指定了*restartwalfile*（或者-k），*archivelocation*目录必须也是可写的。

当主服务器失效时，有两种方式转移到一个“热备”数据库服务器：

智能故障转移

在智能故障转移中，服务器在应用归档中可用的所有 WAL 文件之后被提升。即便备用服务器落后于主服务器，这也会导致零数据丢失，但是如果有很多未应用的 WAL，在备用服务器准备好之前就需要比较长的时间。要触发一次智能故障转移，创建一个包含单词smart的触发文件，或者只创建一个空文件。

快速故障转移

在快速故障转移中，服务器被立即提升。归档中任何还未被应用的 WAL 文件将被忽略，并且这些文件中的所有事务都会丢失。要触发一次快速故障转移，创建一个触发文件并且把单词fast写在其中。ux_standby也能被配置为在一段定义好的区间内没有新 WAL 文件出现时自动执行一次快速故障转移。

2.11.3. 选项

ux_standby接受下列命令行参数：

-c

使用cp或者copy命令来存储来自归档的 WAL 文件。这是唯一支持的行为，因此这个选项是无用的。

-d

在stderr上打印大量调试日志输出。

-k

从*archivelocation*移除文件，这样当前 WAL 文件之前不超过这么多个 WAL 文件会被保留在归档中。零（默认值）意味着不从*archivelocation*移除任何文件。如果指定了*restartwalfile*，这个参数将被安静地忽略，因为那种说明方法对于决定正确的归档切断点更为精确。这个参数的使用已经被废弃，指定一个*restartwalfile*参数更加安全和有效。一个太小的值可能导致备用服务器的重启仍需要已经被移除的文件，而一个太大的值则浪费归档空间。

-r *maxretries*

设定拷贝命令失败时重试的最大次数（默认为 3）。在每一次失败后，我们等待*sleeptime* * *num_retries*，这样等待时间会逐步增加。因此默认情况下，我们将等待 5 秒、10 秒、15秒，然后向备用服务器报告失败。这将被解释为恢复的结束并且该备用服务器将会完全被提升。

-s *sleeptime*

设置在检查要被恢复的 WAL 文件在归档中是否可用的测试之间休眠的秒数（最高 60，默认为 5）。

-t *triggerfile*

指定一个触发文件，它的出现将会导致故障转移。我们推荐使用一个有结构的文件名，这样可以避免在同一个系统上有多个服务器存在时无法确定是要触发哪个服务器。例如可以用/tmp/uxsql.trigger.5432。

-V
--version

打印ux_standby版本并退出。

-w *maxwaittime*

设置等待下一个 WAL 文件的最大秒数，之后将会执行一次快速故障转移。设置为 0 （默认值）表示永远等待。

-?
--help

显示有关ux_standby的命令行参数，然后退出。

2.11.4. 示例

在 Linux 或 Unix 系统上，你可能会使用：

```
archive_command = 'cp %p .../archive/%f'

restore_command = 'ux_standby -d -s 2 -t /tmp/uxsql.trigger.5442 .../archive %f %p %r 2>>standby.log'

recovery_end_command = 'rm -f /tmp/uxsql.trigger.5442'
```

因此存档目录物理上位于备用服务器上，因此archive_command通过NFS访问它，但文件是备用服务器的本地文件（支持使用ln）（启用ln）。这将：

- 在standby.log中产生调试输出
- 在检查下一个 WAL 文件的可用性之间睡眠 2 秒
- 只有当一个名为/tmp/uxsql.trigger.5442的触发文件出现时停止等待并且根据其内容执行故障转移
- 在恢复结束时移除触发文件
- 从归档目录中移除不再需要的文件

在 Windows 上，你可能会用：

```
archive_command = 'copy %p ...\\archive\\%f'

restore_command = 'ux_standby -d -s 5 -t C:\uxsql.trigger.5442 ...\archive %f %p %r 2>>standby.log'

recovery_end_command = 'del C:\uxsql.trigger.5442'
```

注意archive_command中的反斜线需要被双写，但是在restore_command或者recovery_end_command中不需要。这将：

- 使用copy命令恢复来自归档的 WAL 文件
- 在standby.log中产生调试输出
- 在检查下一个 WAL 文件的可用性之间睡眠 5 秒
- 只有当一个名为C:\uxsql.trigger.5442的触发文件出现时停止等待并且根据其内容执行故障转移
- 在恢复结束时移除触发文件
- 从归档目录中移除不再需要的文件

Windows 上的copy命令在文件被完全拷贝之前就会设置最终的文件尺寸，这通常会迷惑ux_standby。因此一旦看到正确的文件尺寸，ux_standby会等待*sleeptime*秒。GNUWin32 的cp只会在文件拷贝完成后设置文件尺寸。

由于 Windows 的例子在两端都使用copy，一个或者两个服务器可能通过网络访问归档目录。

2.12. ux_test_fsync

ux_test_fsync — 为UXDB判断最快的wal_sync_method

2.12.1. 用法

ux_test_fsync［*option*...］

2.12.2. 描述

ux_test_fsync是想告诉你在特定的系统上，哪一种 wal_sync_method最快，还可以在发生认定的 I/O 问题时提供诊断信息。不过，ux_test_fsync 显示的区别可能不会在真实的数据库吞吐量上产生显著的区别，特别是由于 很多数据库服务器被它们的预写日志限制了速度。ux_test_fsync为 wal_sync_method报告以微秒计的平均文件同步操作时间， 也能被用来提示用于优化commit_delay值的方法。

2.12.3. 选项

ux_test_fsync接受下列命令行选项：

-f
--filename

指定要写入测试数据到其中的文件名。这个文件必须位于和 ux_wal目录所在或者将被放置的同一个文件系统中（ ux_wal包含WAL文件）。默认是当前 目录中的ux_test_fsync.out。

-s
--secs-per-test

指定每次测试的秒数。每个测试的时间越长，测试的精度就越高，但是 它需要更多时间来运行。默认是5秒，这允许程序在2分钟以内完成。

-V
--version

打印ux_test_fsync版本并且退出。

-?
--help

显示有关ux_test_fsync命令行参数的帮助并且退出。

2.13. ux_test_timing

ux_test_timing — 度量计时开销

2.13.1. 用法

ux_test_timing [*option*...]

2.13.2. 描述

ux_test_timing是一种度量在你的系统上计时开销以及确认系统时间绝不会回退的工具。收集计时数据很慢的系统会给出不太准确的EXPLAIN ANALYZE结果。

2.13.3. 选项

ux_test_timing接受下列命令行选项：

-d *duration*
--duration=*duration*

指定测试的持续时间，以秒计。更长的持续时间会给出更好一些的精确度，并且更可能发现系统时钟回退的问题。默认的测试持续时间是3秒。

-V
--version

打印ux_test_timing版本并退出。

-?
--help

显示有关ux_test_timing的命令行参数，然后退出。

2.13.4. 功能

2.13.4.1. 结果解读

好的结果将显示大部分（>90%）的单个计时调用用时都小于1微秒。每次循环的平均开销将会更低，低于100纳秒。下面的例子来自于一台使用了一份TSC时钟源码的Intel i7-860系统，它展示了非常好的性能：

```
Testing timing overhead for 3 seconds.
Per loop time including overhead: 35.96 ns
Histogram of timing durations:
  < us   % of total      count
```

```
 1    96.40465   80435604
 2     3.59518    2999652
 4     0.00015        126
 8     0.00002         13
16     0.00000          2
```

注意每次循环时间和柱状图用的单位是不同的。循环的解析度可以在几个纳秒（ns），而单个计时调用只能解析到一个微秒（us）。

2.13.4.2. 度量执行器计时开销

当查询执行器使用**EXPLAIN ANALYZE**运行一个语句时，单个操作会被计时，总结也会被显示。你的系统的负荷可以通过使用uxsql程序计数行来检查：

```
CREATE TABLE t AS SELECT * FROM generate_series(1,100000);
\timing
SELECT COUNT(*) FROM t;
EXPLAIN ANALYZE SELECT COUNT(*) FROM t;
```

i7-860系统测到运行该计数查询用了9.8 ms而**EXPLAIN ANALYZE**版本则需要16.6ms，每次处理都在100,000行上进行。6.8 ms的差别意味着在每行上的计时负荷是68 ns，大概是 ux_test_timing 估计的两倍。即使这样相对少量的负荷也造成了带有计时的计数语句耗时多出了70%。在更大量的查询上，计时开销带来的问题不会有这么明显。

2.13.4.3. 改变时间来源

在一些较新的 Linux 系统上，可以在任何时候更改用来收集计时数据的时钟来源。第二个例子显示了在上述快速结果的相同系统上切换到较慢的acpi_pm时间源可能带来的降速：

```
# cat /sys/devices/system/clocksource/clocksource0/available_clocksource
tsc hpet acpi_pm
# echo acpi_pm > /sys/devices/system/clocksource/clocksource0/current_clocksource
# ux_test_timing
Per loop time including overhead: 722.92 ns
Histogram of timing durations:
  < us   % of total      count
     1     27.84870    1155682
     2     72.05956    2990371
     4      0.07810       3241
     8      0.01357        563
    16      0.00007          3
```

在这种配置中，上面的例子**EXPLAIN ANALYZE**用了115.9 ms。其中有1061 ns的计时开销，还是用这个工具直接度量结果的一个小倍数。这么多的计时开销意味着实际的查询本身只占了时间的一个很小的分数，大部分的时间都耗在了计时所需的管理开销上。在这种配置中，任何涉及到很多计时操作的**EXPLAIN ANALYZE**都会受到计时开销的显著影响。

FreeBSD也允许即时更改时间源，并且它会记录在启动期间有关计时器选择的信息：

```
# dmesg | grep "Timecounter"
Timecounter "ACPI-fast" frequency 3579545 Hz quality 900
Timecounter "i8254" frequency 1193182 Hz quality 0
```

```
Timecounters tick every 10.000 msec
Timecounter "TSC" frequency 2531787134 Hz quality 800
# sysctl kern.timecounter.hardware=TSC
kern.timecounter.hardware: ACPI-fast -> TSC
```

在旧的 Linux 系统上，“clock”内核设置是做这类更改的唯一方法。并且即使在一些更近的系统上，对于一个时钟源你将只能看到唯一的选项 ″jiffies″。Jiffies 是老的 Linux 软件时钟实现，当有足够快的计时硬件支持时，它能够具有很好的解析度，就像在这个例子中：

```
$ cat /sys/devices/system/clocksource/clocksource0/available_clocksource
jiffies
$ dmesg | grep time.c
time.c: Using 3.579545 MHz WALL PM GTOD PIT/TSC timer.
time.c: Detected 2400.153 MHz processor.
$ ux_test_timing
Testing timing overhead for 3 seconds.
Per timing duration including loop overhead: 97.75 ns
Histogram of timing durations:
  < us   % of total      count
     1     90.23734   27694571
     2      9.75277    2993204
     4      0.00981       3010
     8      0.00007         22
    16      0.00000          1
    32      0.00000          1
```

2.13.4.4. 时钟硬件和计时准确性

在计算机上通常是使用具有不同精度的时钟硬件收集准确的计时信息。操作系统基本能够通过使用一些硬件直接把系统时钟时间传递给程序。系统时钟也可以产生于一块简单的芯片，这块芯片提供计时中断、在某个已知时间间隔内周期性地发出滴答声。在两种情况中，操作系统内核提供一个隐藏这些细节的时钟源。但是时钟源的精确度以及能多快返回结果会根据底层硬件而变化。

不精确的计时导致系统不稳定，对任何时钟源的更改都要仔细地测试。操作系统默认是有时会更倾向于可靠性而不是精确性。如果正在使用一个虚拟机器，应查看与之兼容的推荐时间源。在模拟计时器时，虚拟硬件面临着额外的困难，供应商通常会建议对每个操作系统进行设置。

时间戳计数器（TSC）时钟源是当前一代CPU上最精确的一种。当操作系统支持TSC并且TSC可靠时，它是跟踪系统时间更好的方式。有多种方式会使TSC无法提供准确的计时源，这会让它不可靠。旧的系统能有一种基于CPU温度变化的TSC时钟，这让它不能用于计时。尝试在一些就的多核CPU上使用TSC可能在多个核心之间给出不一致的时间报告。这可能导致时间倒退，这个程序会检查这种问题。并且即使最新的系统，在非常激进的节能配置下也可能无法提供准确的TSC计时。

更新的操作系统可能检查已知的TSC问题并且当它们被发现时切换到一种更慢、更稳定的时钟源。如果你的系统支持TSC时间但是并不默认使用它，很可能是由于某种充分的理由才禁用它。某些操作系统可能无法正确地检测所有可能的问题，或者即便在知道TSC不精确的情况下也允许使用TSC。

如果系统上有高精度事件计时器（HPET）并且TSC不准确，该系统将会更喜欢HPET计时器。计时器芯片本身是可编程的，最高允许100纳米的解析度，但是在你的系统时钟中可能见不到那么高的准确度。

高级配置和电源接口（ACPI）提供了一种电源管理（PM）计时器，Linux 把它称之为acpi_pm。得自于acpi_pm 的时钟最好时将能提供300纳秒的解析度。

在旧的PC硬件上使用的计时器包括8254可编程区间计时器（PIT）、实时时钟（RTC）、高级可编程中断控制器（APIC）计时器以及Cyclone计时器。这些计时器是以毫秒解析度为目标的。

2.14. ux_upgrade

ux_upgrade — 升级UXDB服务器实例

2.14.1. 用法

ux_upgrade -b -B *newbindir* -d *oldconfigdir* -D *newconfigdir* [*option...*]

2.14.2. 描述

ux_upgrade（之前被称为ux_migrator） 允许存储在UXDB数据文件中的数据被升级到一个更高的UXDB主版本而无需进行主版本升级通常所需的数据转储/重载。

主UXDB发行通常会加入新的特性，这些新特性常常会更改系统表的布局，但是内部数据存储格式很少会改变。ux_upgrade使用这一事实来通过创建新系统表并且重用旧的用户数据文件来执行快速升级。如果一个未来的主发行没有把数据存储格式改得让旧数据格式不可读取，这类升级就用不上ux_upgrade（将尝试避免这类情况）。

ux_upgrade会尽力（例如通过检查兼容的编译时设置）确保新旧集群在二进制上也是兼容的，包括32/64位二进制。保持外部模块也是二进制兼容的也很重要，不过ux_upgrade无法检查这一点。

2.14.3. 选项

ux_upgrade接受下列命令行参数：

-b *bindir*
--old-bindir=*bindir*

旧的UXDB可执行文件目录； 环境变量UXBINOLD。

-B *bindir*
--new-bindir=*bindir*

新的UXDB可执行文件目录； 环境变量UXBINNEW。

-c
--check

只检查集群，不更改任何数据。

-d *configdir*
--old-datadir=*configdir*

旧的集群数据目录；环境变量UXDATAOLD。

-D *configdir*
--new-datadir=*configdir*

新的集群数据目录；环境变量UXDATANEW。

-j
--jobs

要同时使用的进程或线程数。

-k
--link

使用硬链接来代替将文件拷贝到新集群。

-o *options*
--old-options *options*

直接传送给旧 uxdb命令的选项，多个选项可以追加在后面。

-O *options*
--new-options *options*

直接传送给新 uxdb命令的选项，多个选项可以追加在后面。

-p *port*
--old-port=*port*

旧的集群端口号；环境变量 UXPORTOLD。

-P *port*
--new-port=*port*

新的集群端口号；环境变量 UXPORTNEW。

-r
--retain

即使在成功完成后也保留SQL和日志文件。

-s *dir*
--socketdir=*dir*

用于升级期间uxmaster套接字的目录;默认是当前目录; 环境变量 UXSOCKETDIR

-U *username*
--username=*username*

集群的安装用户名；环境变量 UXUSER。

-v
--verbose

启用详细的内部日志。

-V
--version

显示版本信息，然后退出

--clone

使用有效的文件克隆（在一些系统上也被称为“reflinks”），而不是将文件拷贝到新集群。 这可以导致数据文件接近瞬时的复制，从而获得-k/--link的速度优势，同时保留旧集群不受影响。

文件克隆仅在某些操作系统和文件系统上得到支持。如果选中但不被支持，则 ux_upgrade运行将会出错。目前，它支持在Linux（内核4.5或更高版本）上的Btrfs和XFS（在文件系统创建reflink支持），以及macOS上的APFS。

-?
--help

显示帮助，然后退出。

2.14.4. 使用

下面是用ux_upgrade执行一次升级的步骤：

1. 移动旧集群（可选）

   移动当前的UXDB安装目录，以免它干扰新的 UXDB安装。一旦当前的 UXDB服务器被关闭，就可以安全地重命名 UXDB安装目录。假设旧目录是 /home/uxdb/uxdbinstall，你可以这样：

   ```
   mv /home/uxdb/uxdbinstall /home/uxdb/uxdbinstall.old
   ```

   来重命名该目录。

2. 安装新的版本

   安装新服务器的二进制文件和支持文件。ux_upgrade 会被包含在默认的安装中。

3. 初始化新的UXDB集群

   使用initdb初始化新集群。这里也要使用与 旧集群相兼容的initdb标志。许多预编译的 安装程序会自动做这个步骤。这里没有必要启动新集群。

4. 安装自定义的共享对象文件

   把旧集群使用的所有自定义共享对象文件（或者DLL）安装到新集亲中， 例如uxcrypto.so，不管它们是来自于 contrib还是某些其他源码。不要安装模式定义 （例如CREATE EXTENSION uxcrypto），因为这些将会从旧集群升级得到。 还有，任何自定义的全文搜索文件（词典、同义词、辞典、停用词）也必须 被复制到新集群中。

5. 调整认证

   ux_upgrade将会多次连接到旧服务器和新服务器，因此 你可能想要在ux_hba.conf中把认证设置成 peer或者使用一个~/.uxpass文件。

6. 停止两个服务器

   确认两个数据库服务器都被停止使用，例如在 Unix 上可以：

   ```
   ux_ctl -D /opt/UXDB/test stop
   ```

ux_ctl -D /opt/UXDB/test1 stop

流复制和日志传送备用服务器可以一直运行到后面的步骤。

7. 为升级备用服务器做准备

如果正在使用小节步骤9中给出的方法升级备用服务器，请通过对旧的主集群和备用集群运行ux_controldata来验证旧的备用服务器是否被占用。验证“最新检查点位置”的值在所有集群中是否匹配。（如果在旧主服务器之前关闭旧备用服务器，将会出现不匹配）。此外，在新的主集群上的uxsinodb.conf文件中把wal_level改为replica。

8. 运行 ux_upgrade

总是应该运行新服务器而不是旧服务器的ux_upgrade二进制文件。 ux_upgrade要求制定新旧集群的数据和可执行文件（ bin）目录。也可以指定用户和端口值，以及是否要链接数据文件而不是默认的复制行为。

如果使用链接模式，升级将会快很多（不需要文件拷贝）并且将使用更少的磁盘空间，但是在升级后一旦启动新集群，旧集群就无法被访问。链接模式也要求新旧集群数据目录位于同一个文件系统中（表空间和ux_wal可以在不同的文件系统中）。完整的选项列表可见ux_upgrade --help。克隆模式提供了相同的速度以及磁盘空间优势，但不会导致新集群启动后旧集群不可用。 克隆模式还需要新旧数据目录位于同一文件系统中。 此模式仅在某些操作系统和文件系统上可用。

--jobs选项允许多个CPU核心被用来复制/链接文件以及 并行地转储和重载数据库模式。这个选项一个比较好的值是CPU核心数 和表空间数的最大值。这个选项可以显著地减少升级运行在一台多处理 器机器上的多数据库服务器的时间。

对于 Windows 用户，必须以一个超级账号登录，并且以 uxdb用户启动一个shell并且设置正确的路径：

```
RUNAS /USER:uxdb "CMD.EXE"
SET PATH=%PATH%;C:\Program Files\UXDB\dbsql\bin;
```

并且用带引号的目录运行ux_upgrade，例如：

```
ux_upgrade.exe
    --old-datadir "C:/Program Files/UXDB/test/data"
    --new-datadir "C:/Program Files/UXDB/test1/data"
    --old-bindir "C:/Program Files/UXDB/test/bin"
    --new-bindir "C:/Program Files/UXDB/test1/bin"
```

一旦启动，ux_upgrade将验证两个集群是否兼容并且 执行升级。你可以使用ux_upgrade --check来只执行检查， 这种模式即使在旧服务器还在运行时也能使用。 ux_upgrade --check也将列出任何在更新后需要做的手工 调整。如果你将要使用链接模式，你应该使用--link选项和 --check一起来启用链接模式相关的检查。 ux_upgrade要求在当前目录中的写权限。

显然，没有人可以在升级期间访问这些集群。ux_upgrade 默认会在端口5432上运行服务器来避免意外的客户端连接。在做升级时， 可以对两个集群使用相同的端口号，因为新旧集群不会在同时被运行。不过， 在检查一个旧的运行中服务器时，新旧端口号必须不同。

如果在恢复数据库模式时发生错误，ux_upgrade将会退出 并且你必须按照下文步骤15中所说的恢复 旧集群。要再次尝试ux_upgrade，你将需要修改 旧集群，这样ux_upgrade模式会

成功恢复。如果问题是一个 contrib模块， 你可能需要从旧集群中卸载该模块并且在升级后重新把它安装在新集群中，不过 这样做的前提是该模块没有被用来存储用户数据。

9. 升级流复制和日志传送备用服务器

   如果使用链接模式并且有流复制或者日志 传送备用服务器，你可以遵照下面的 步骤对它们进行快速的升级。你将不用在这些备用服务器上运行 ux_upgrade，而是在主服务器上运行rsync。 到这里还不要启动任何服务器。

   如果你没有使用链接模式、没有或不想使用rsync或者想用一种更容易的解决方案，请跳过这一节中的过程并且在ux_upgrade完成并且新的主集群开始运行后重建备用服务器。

   a. 在备用服务器上安装新的UXDB二进制文件

      确保新的二进制和支持文件被安装在所有备用服务器上。

   b. 确保不存在新的备用机数据目录

      确保新的备用机数据目录不存在或者为空。如果 运行过initdb，请删除备用服务器的新数据目录。

   c. 安装自定义共享对象文件

      在新的备用机上安装和新的主集群中相同的自定义共享对象文件。

   d. 停止备用服务器

      如果备用服务器仍在运行，现在使用上述的指令停止它们。

   e. 保存配置文件

      从旧备用机的配置目录保存任何需要保留的配置文件，例如 uxsinodb.conf、recovery.conf， 因为这些文件在下一步中会被重写或者移除。

   f. 运行rsync

      在使用链接模式时，备用服务器可以使用rsync快速升级。为了实现这一点，在主服务器上一个高于新旧数据库集群目录的目录中为每个备用服务器运行这个命令：

      ```
      rsync --archive --delete --hard-links --size-only --no-inc-recursive old_cluster new_cluster
       remote_dir
      ```

      其中old_cluster和new_cluster是相对于主服务器上的当前目录的，而remote_dir是备用服务器上高于新旧集群目录的一个目录。在主服务器和备用服务器上指定目录之下的目录结构必须匹配。指定远程目录的详细情况请参考rsync的手册，例如：

      ```
      rsync --archive --delete --hard-links --size-only --no-inc-recursive /opt/UXDB/test/data \
      /opt/UXDB/test1/data standby.example.com:/opt/UXDB
      ```

      可以使用rsync的--dry-run选项验证该命令将做的事情。虽然在主服务器上必须为至少一台备用运行rsync，可以在一台已经升级过的备用服务器上运行rsync来升级其他的备用服务器，只要已升级的备用服务器还没有被启动。

      这个命令所做的事情是记录由ux_upgrade的链接模式创建的链接，它们连接主服务器上新旧集群中的文件。该命令接下来在备用服务器的旧集群中寻找匹配的文件并且为它们

在该备用的新集群中创建链接。主服务器上没有被链接的文件会被从主服务器拷贝到备用服务器（通常都很小）。这提供了快速的备用服务器升级。不幸地是，rsync会不必要地拷贝与临时表和不做日志表相关的文件，因为通常在备用服务器上不存在这些文件。

如果有表空间，你将需要为每个表空间目录运行一个类似的rsync命令，例如：

```
rsync --archive --delete --hard-links --size-only --no-inc-recursive /vol1/ux_tblsp/
ux_test_201710051 \
 /vol1/ux_tblsp/ux_test1_201808131 standby.example.com:/vol1/ux_tblsp
```

如果你已经把ux_wal放在数据目录外面，也必须在那些目录上运行rsync。

g. 配置流复制和日志传送备用服务器

为日志传送配置服务器（不需要运行ux_start_backup() 以及ux_stop_backup()或者做文件系统备份，因为从属机 仍在与主机同步）。

10. 恢复 **ux_hba.conf**

如果你修改了ux_hba.conf，则要将其恢复到原始的设置。 也可能需要调整新集群中的其他配置文件（例如 uxsinodb.conf）来匹配旧集群。

11. 启动新服务器

现在可以安全地启动新的服务器，并且可以接着启动任何rsync备用服务器。

12. 升级后处理

如果需要做任何升级后处理，ux_upgrade 将在完成后发出警告。它也将 生成必须由管理员运行的脚本文件。这些脚本文件将连接到每一个需要做 升级后处理的数据库。每一个脚本应该这样运行：

```
uxql --username=uxdb --file=script.sql uxdb
```

这些脚本可以以任何顺序运行并且在运行之后立即删除。

> **注意**
>
> 通常在重建脚本运行完成之前访问重建脚本中引用的表是不安全的，这样做 可能会得到不正确的结果或者很差的性能。没有在重建脚本中引用的表可以 随时被访问。

13. 统计信息

由于ux_upgrade并未传输优化器统计信息，在升级的尾声 你将被指示运行一个命令来生成这些信息。你可能需要设置连接参数来匹配你 的新集群。

14. 删除旧集群

一旦你对升级表示满意，你就可以通过运行 ux_upgrade完成时提到的脚本来删除旧集群的数据目录（如果在旧数据目录中有用户定义的表空间就不可能实现自动删除）。 你也可以删除旧安装目录（例如bin、share）。

15. 恢复到旧集群

在运行ux_upgrade之后，如果你希望恢复到旧集群，有几个选项：

- 如果使用了 --check 选项，则旧集群没有被修改；它可以被重新启动。
- 如果 --link选项没有被使用，旧集群没有被修改；它可以被重新启动。
- 如果使用了--link 选项，数据文件可能在新旧集群之间共享:
  - 如果ux_upgrade在链接启动之前中止，旧集群没有被修改，它可以重新启动。
  - 如果你没有启动新集群，旧集群没有被修改，当链接启动时，一个.old后缀会附加到$UXDATA/global/ux_control。 如果要重用旧集群，从$UXDATA/global/ux_control移除.old后缀；你就可以重启旧集群。
  - 如果你已经启动新集群，它已经写入了共享文件，并且使用旧集群会不安全。这种情况下，需要从备份中还原旧集群。

2.14.5. 注解

ux_upgrade创建不同的工作文件，如模式转储，在当前工作目录中。为了安全，请确保该目录不可被任何其他用户读取或者写入。

ux_upgrade在新旧数据目录中启动短期的uxmaster。临时 Unix 套接字文件用于与这些uxmaster通信，默认情况下，在当前工作目录中进行。 在某些情况下，当前目录的路径名称可能太长，无法成为有效的套接字名称。这种情况下你可以使用-s选项将套接字文件放在某些具有较短路径名称的目录中。 为了安全原因，请确保该目录不可被任何其他用户读取或者写入。（这与 Windows 无关。）

如果失败、重建和重索引会影响你的安装，ux_upgrade将会报告这些情况。用来重建表和索引的升级后脚本将会自动被建立。如果你正在尝试自动升级很多集群，你应该发现具有相同数据库模式的集群对所有集群升级都要求同样的升级后步骤，这是因为升级后步骤是基于数据库模式而不是用户数据。

对于部署测试，创建一个只有模式的旧集群副本，在其中插入假数据并且升级。

ux_upgrade不支持对某些数据库的升级，此类 数据库包含以下reg*开头的OID引用的系统数据类型： regproc、regprocedure、 regoper、regoperator、 regconfig以及regdictionary（ regtype可以被升级）。

如果你想要使用链接模式并且你不想让你的旧集群在新集群启动时被修改， 可以复制一份旧集群并且在副本上以链接模式进行升级。要创建旧集群的一 份合法拷贝，可以在服务器运行时使用rsync创建旧集群的 一份脏拷贝，然后关闭旧服务器并且再次运行rsync --checksum 把更改更新到该拷贝以让其一致（--checksum是必要的，因为 rsync在判断文件修改时间的更改时的精度只能到秒级）。可能想要排除 一些文件，例如uxmaster.pid。如果你的文件系统支持文 件系统快照或者copy-on-write文件副本，你可以使用它们来创建旧集群和 表空间的一个备份，不过快照和副本必须被同时创建或者在数据库服务器关闭 期间被创建。

2.15. ux_waldump

ux_waldump — 以可读的形式显示一个UXDB数据库集群的预写式日志

2.15.1. 用法

ux_waldump［option...］［startseg［endseg］］

2.15.2. 描述

ux_waldump显示预写式日志（WAL），它主要用于调试。

这个工具只能由安装该服务器的用户运行，因为它要求对数据目录的只读访问。

2.15.3. 选项

下列命令行选项控制输出的位置和格式：

startseg

从指定的日志段文件开始读取。这也隐含地决定了要搜索文件的路径以及 要使用的时间线。

endseg

在读取指定的日志段文件后停止。

-b
--bkp-details

输出有关备份块的细节。

-e *end*
--end=*end*

在指定的WAL位置停止读取，而不是一直读取到日志流的末尾。

-f
--follow

在到达可用WAL的末尾之后，保持每秒轮询一次是否有新的WAL出现。

-n *limit*
--limit=*limit*

显示指定数量的记录，然后停止。

-p *path*
--path=*path*

指定搜索日志段文件的目录或包含这些文件的包含ux_wal子目录的目录。 缺省值是在当前目录中搜索，当前目录的ux_wal子目录和 UXDATA的ux_wal子目录。

-r *rmgr*
--rmgr=*rmgr*

只显示由指定资源管理器生成的记录。如果把list作为资源管理器名称 传递给这个选项，则打印出可用资源管理器名称的列表然后退出。

-s *start*
--start=*start*

开始读取的WAL位置。默认是从找到的最早的文件的第一个可用日志记录开始。

-t *timeline*
--timeline=*timeline*

读取日志记录的时间线。默认是使用*startseg*（如果指定） 中的值，否则默认为1。

-V
--version

打印ux_waldump版本并且退出。

-x *xid*
--xid=*xid*

只显示用给定事务ID标记的记录。

-z
--stats[=record]

显示概括统计信息（记录的数量和尺寸以及全页镜像）而不是显示 每个记录。可以选择针对每个记录生成统计信息，而不是针对每个 资源管理器生成。

-?
--help

显示有关ux_waldump命令行参数的帮助并且退出。

2.15.4. 环境变量

UXDATA

数据目录，另请参见-p 选项。

UX_COLOR

规定在诊断消息中是否使用颜色。可能的值为always、 auto、never。

2.15.5. 注解

当服务器正在运行时可能会给出错误的结果。

只有指定的时间线会被显示（如果没有指定，则显示默认时间线）。 其他时间线上的记录会被忽略。

ux_waldump不能读取具有后缀.partial 的WAL文件。如果需要读取那些文件，需要从文件名中移除.partial后缀。

2.16. uxdb

uxdb — UXDB数据库服务器

2.16.1. 用法

uxdb [option...]

2.16.2. 描述

uxdb是UXDB数据库服务器。一个客户端应用为了能访问一个数据库，它会（通过一个网络或者本地）连接到一个运行着的uxdb实例。该uxdb实例接着会开始一个独立的服务器进程来处理该连接。

一个uxdb实例总是管理正好一个数据库集群的数据。一个数据库集群是一个数据库的集合，它们被存储在一个共同的文件系统位置（“数据区”）上。 一个系统上可以同时运行多个uxdb实例，只要它们使用不同的数据区和不同的通信端口。uxdb启动时需要知道数据区的位置，该位置必须通过-D选项或UXDATA环境变量指定，对此是没有默认值的。通常，-D或UXDATA会直接指向由initdb创建的数据区目录。

默认情况下，uxdb会在前台启动并将日志消息打印到标准错误流。但在实际应用中，uxdb应当作为一个后台进程启动，而且多数是在系统启动时自动启动。

uxdb还能在单用户模式中被调用。这种模式的主要用途是在启动过程中由initdb使用。有时候它也被用于调试或者灾难性恢复。注意，运行一个单用户模式服务器并不真地适合调试服务器，因为不会发生实际的进程间通信和锁定。当从shell中调用单用户模式时，用户可以输入查询并且结果会被以一种更适合开发者阅读（不适合普通用户）的形式打印在屏幕上。在单用户模式中，会话用户将被设置为 ID 为1的用户，并且这个用户会被隐式地赋予超级用户权限。该用户不必实际存在，因此单用户模式运行可以被用来对某些意外损坏的系统目录进行手工恢复。

2.16.3. 选项

uxdb接受下列命令行参数。你可以通过设置一个配置文件来减少输入大部分这些选项。有些（安全）选项还可以从连接的客户端以一种与应用相关只应用于会话的方法设置。例如，如果设置了UXOPTIONS环境变量，那么基于libuxsql的客户端将都把那个字符串传递给服务器，它将被服务器解释成uxdb命令行选项。

2.16.3.1. 通用选项

-B *nbuffers*

设置被服务器进程使用的共享内存缓冲区数量。这个参数的默认值是initdb自动选择的。指定这个选项等效于设置shared_buffers配置参数。

-c *name=value*

设置指定的运行时参数。大多数其它命令行选项实际上都是这种参数赋值的短形式。-c可以出现多次用于设置多个参数。

-C *name*

打印指定的运行时参数，并且退出。这可以被用在一个运行服务器上，并且从uxsinodb.conf中返回值，这些值可能被在这次调用中的任何参数修改过。它并不反映集群启动时提供的参数。

这个选项用于与一个服务器实例交互的其他程序来查询配置参数值，例如ux_ctl。面向用户的应用应该使用SHOW或者ux_settings视图。

-d *debug-level*

设置调试级别。数值设置得越高，写到服务器日志的调试输出就越多。取值范围是从1到5。还可以针对某个特定会话使用-d 0来阻止父uxdb进程的服务器日志级别被传播到这个会话。

-D *datadir*

指定数据库配置文件的文件系统位置。

-e

把默认日期风格设置为“European”，也就是输入日期域的顺序是DMY。这也导致在一些日期输出格式中把日打印在月之前。

-F

禁用fsync调用以提高性能，但是要冒系统崩溃时数据损坏的风险。指定这个选项等效于禁用fsync配置参数。

-h *hostname*

指定uxdb监听来自客户端应用 TCP/IP 连接的 IP 主机名或地址。该值也可以是一个用逗号分隔的地址列表，或者*表示监听所有可用的地址。一个空值表示不监听任何IP地址，在这种情况下可以使用 Unix 域套接字连接到服务器。缺省只监听localhost。声明这个选项等效于设置listen_addresses配置参数。默认只监听localhost。指定这个选项等效于设置listen_addresses配置参数。

-i

允许远程客户端使用 TCP/IP （互联网域）连接。没有这个选项，将只接受本地连接。这个选项等效于在uxdbsinodb.conf中或者通过-h选项将listen_addresses设为*。

这个选项已经被废弃，因为它不允许访问listen_addresses的完整功能。所以最好直接设置listen_addresses。

-k *directory*

指定uxdb用来监听来自客户端应用连接的 Unix 域套接字的目录。这个值也可以是一个逗号分隔的目录列表。一个空值指定不监听任何 Unix 域套接字，在这种情况下只能用 TCP/IP 套接字来连接到服务器。默认值通常是/tmp，但是可以在编译的时候修改。指定这个选项等效于设置unix_socket_directories配置参数。

-l

启用使用SSL的安全连接。要使这个选项可用，编译UXDB时必须打开SSL支持。

-I

初始化数据库级审计。

-M

启用全数据库级加密。注意，数据库必须以全数据库加密模式初始化。

-Y
--encrypted-pwfile=*filename*

从文件中读取密码，启动加密的数据库。

-N *max-connections*

设置该服务器将接受的最大客户端连接数。该参数的默认值由initdb自动选择。指定这个选项等效于设置max_connections配置参数。

-o *extra-options*

在*extra-options*中指定的命令行风格的参数会被传递给所有由这个uxdb进程派生的服务进程。

*extra-options*中的空格被视作 参数分隔符，除非用反斜线（\）转义。要表示一个字面意 义上的反斜线，可以写成\\。通过多次使用-o 也可以指定多个参数。

这个选项的使用已经被废弃。用于服务器进程的所有命令行选项可以在uxdb命令行上直接指定。

-p *port*

指定uxdb用于监听客户端应用连接的 TCP/IP 端口或本地 Unix 域套接字文件扩展。默认为UXPORT环境变量的值。如果UXPORT没有设置，那么默认值是编译期间设立的值（通常是5432）。如果你指定了一个非默认端口，那么所有客户端应用都必须用命令行选项或者UXPORT指定同一个端口。

-s

在每条命令结束时打印时间信息和其它统计信息。这个选项对测试基准和调节缓冲区数量有用处。

-S *work-mem*

指定内部排序和散列在使用临时磁盘文件之前能使用的内存数量。

-V
--version

打印uxdb版本并退出。

--name=value

设置一个命名的运行时参数；其缩写形式是-c。

--describe-config

这个选项会用制表符分隔的COPY格式导出服务器的内部配置变量、描述以及默认值。设计它的目的是用于管理工具。

-?
--help

显示有关uxdb的命令行参数，并且退出。

2.16.3.2. 调试选项

这里描述的选项主要被用于调试目的，并且在某些情况下可以协助恢复严重受损的数据库。在生产数据库环境中应该不会去使用它们。在这里列举它们只是为了让UXDB系统开发者使用。

-f { s | i | o | b | t | n | m | h }

禁止某种扫描和连接方法的使用：s和i分别禁用顺序和索引扫描， o、b和t分别禁用只用索引扫描、位图索引扫描以及TID扫描，而n、m和h则分别禁用嵌套循环、归并和哈希连接。

顺序扫描和嵌套循环连接都不可能完全被禁用。-fs和-fn选项仅仅是在有其他选择时不鼓励优化器使用这些计划类型。

-n

该选项主要用于调试导致服务器进程异常崩溃的问题。对付这种情况的一般策略是通知所有其它服务器进程，让它们终止并且接着重新初始化共享内存和信号量。这是因为一个错误的服务器进程可能在终止之前就已经对共享状态造成了破坏。该选项指定uxdb将不会重新初始化共享数据结构。一个有经验的系统程序员这时就可以使用调试器检查共享内存和信号量状态。

-O

允许修改系统表的结构。这个选项用于initdb。

-P

读取系统表时忽略系统索引（但在更改系统表时仍然更新索引）。这在从损坏的系统索引中恢复时有用。

-t pa[rser] | pl[anner] | e[xecutor]

打印与每个主要系统模块相关的查询的时间统计。这个选项不能和-s选项一起使用。

-T

该选项主要用于调试导致服务器进程异常崩溃的问题。对付这种情况的一般策略是通知所有其它服务器进程，让它们终止并且接着重新初始化共享内存和信号量。这是因为一个错误的服务器进程可能在终止之前就已经对共享状态造成了破坏。该选项指定uxdb将通过发送SIGSTOP信号停止其他所有服务器进程，但是并不让它们终止。这样就允许系统程序员手动从所有服务器进程收集内核转储。

-v *protocol*

声明这次会话使用的前/后服务器协议的版本数。该选项仅在内部使用。

-W *seconds*

在一个新服务器进程被启动时，它实施认证过程之后会延迟这个选项所设置的秒数。这就留出了机会来用一个调试器附着在服务器进程上。

2.16.3.3. 用于单用户模式的选项

下面的选项仅适用于单用户模式。

--single

选择单用户模式。这必须是命令行中的第一个选项。

database

指定要访问的数据库的名称。这必须是命令行中的最后一个参数。如果省略它，则默认为用户名。

-d *debug-level*

设置调试级别。数值设置得越高，写到服务器日志的调试输出就越多。取值范围是从1到5。还可以针对某个特定会话使用-d 0来阻止父uxdb进程的服务器日志级别被传播到这个会话。

-E

在执行命令之前回显所有命令到标准输出。

-j

使用跟着两个新行的分号而不是仅用新行作为命令终止符。

-r *filename*

将所有服务器日志输出发送到*filename*中。 只有在作为一个命令行选项提供时，这个选项才会兑现。

2.16.3.4. 用于引导模式的选项

下面的选项仅适用于引导模式。

--boot

选择引导模式。这必须是命令行中的第一个选项。

database

指定要访问的数据库的名称。在引导模式中是必选参数。

-r *filename*

将所有服务器日志输出发送到*filename*中。

-x *num*

内部使用。

2.16.4. 环境变量

UXCLIENTENCODING

客户端使用的默认字符编码（客户端可以独立地覆盖它）。这个值也可以在配置文件中设置。

UXDATA

默认的数据目录位置。

UXDATESTYLE

DateStyle运行时参数的默认值（这个环境变量的使用已被废弃）。

UXPORT

默认端口号（在配置文件中设置更好）。

2.16.5. 诊断

一个提到了semget或shmget的错误消息可能意味着你需要配置内核来提供足够的共享内存和信号量。你也可以通过降低shared_buffers值减少UXDB的共享内存消耗， 或者降低max_connections值减少信号量消耗，这样可以推迟对内核的重新配置。

如果一个消息说另外一个服务器已经在运行，应该仔细地检查，例如根据你的系统可以用命令：

```
$ ps ax | grep uxdb
```

或

```
$ ps -ef | grep uxdb
```

如果你确信没有冲突的服务器正在运行，那么你可以删除消息中提到的锁文件然后再次尝试。

如果一个失败消息指示它无法绑定到一个端口，可能意味着该端口已经被某些非UXDB进程所使用。如果你终止uxdb并且立即使用相同的端口重启它，你也可能会得到这种错误。在这种情况系，你必须等待几秒直到操作系统关闭该端口，然后再重试。最后，如果你指定了一个操作系统认为需要保留的端口号，你可能也会得到这个错误。例如，很多版本的 Unix 认为低于1024的端口号是“可信的”并且只允许 Unix 超级用户访问它们。

2.16.6. 注解

实用命令ux_ctl可以用来安全方便地启动和关闭uxdb服务器。

只要有可能，就不要使用SIGKILL杀死主uxdb服务器。这样会阻止uxdb在终止前释放它持有的系统资源（例如共享内存和信号量）。这样可能会导致启动新的uxdb进程时出现问题。

要正常地终止uxdb服务器，可以使用SIGTERM、SIGINT或者SIGQUIT信号。第一个在退出前将等待所有客户端终止，第二个将强行断开所有客户端的连接，第三个会不做正确的关闭立即退出并且会导致重启时的恢复。

SIGHUP信号会重新加载服务器配置文件。也可以向一个单独的服务器进程发送SIGHUP信号，但是这样做通常没什么意义。

要取消一个正在运行的查询，可以向运行该查询的进程发送SIGINT信号。要干净地终止一个后端进程，可向它发送SIGTERM。在SQL中可调用的与这两种动作等效的命令为ux_cancel_backend和ux_terminate_backend。

uxdb服务器使用SIGQUIT来告诉子服务器进程终止但不做正常的清理。该信号不应该被用户使用。向一个服务器进程发送SIGKILL也是不明智的 — 主uxdb进程将把这解释为一次崩溃，并且作为其标准崩溃恢复过程的一部分，它将强制所有的后台进程退出。

2.16.7. 单用户模式

要启动一个单用户模式的服务器，使用这样的命令：

```
uxdb --single -D /home/uxdb/uxdbinstall/dbsql/data other-options my_database
```

用-D给服务器提供正确的数据库目录的路径，或者确保环境变量UXDATA被设置。同时还要指定你想在其中工作的特定数据库的名字。

通常，单用户模式的服务器会把换行符当做命令输入的终止符。要想把一个命令分成多行，必须在最后一个换行符以外的每个换行符前面敲一个反斜线。这个反斜线和旁边的新行都会被从输入命令中去掉。注意即使在字符串或者注释中也会这样做。

但是如果使用了-j命令行选项，那么单个新行将不会终止命令输入。相反，分号-新行-新行的序列才会终止命令输入。也就是说，输入一个紧跟着空行的分号。在这种模式下，反斜线-新行不会被特殊对待。此外，在字符串或者注释内的这类序列也不会被特殊对待。

不管在哪一种输入模式中，如果输入的一个分号不是正好在命令终止符之前或者不是命令终止符的一部分，它会被认为是一个命令分隔符。当真正输入一个命令终止符时，已经输入的多个语句将被作为一个单个事务执行。

要退出会话，输入EOF（通常是Control+D）。如果从上一个命令终止符以来已经输入了任何文本，那么EOF将被当作命令终止符，并且如果要退出则需要另一个EOF。

请注意单用户模式的服务器不会提供复杂的行编辑特性（例如没有命令历史）。但用户模式也不会做任何后台处理，例如自动检查点或者复制。

2.16.8. 示例

要用默认值在后台启动uxdb：

```
$ nohup uxdb >logfile 2>&1 </dev/null &
```

要用指定端口启动uxdb，例如 1234：

```
$ uxdb -p 1234
```

要使用uxsql连接到这个服务器，用-p选项指定这个端口：

```
$ uxsql -p 1234
```

或者设置环境变量UXPORT：

```
$ export UXPORT=1234
$ uxsql
```

命名运行时参数可以用这些形式之一设置：

```
$ uxdb -c work_mem=1234
$ uxdb --work-mem=1234
```

两种形式都覆盖uxsinodb.conf中可能存在的work_mem设置。请注意在参数名中的下划线在命令行可以写成下划线或连字符。除了用于短期的使用外，更好的习惯是编辑uxsinodb.conf中的设置， 而不是倚赖命令行开关来设置参数。

2.17. uxmaster

uxmaster — UXDB数据库服务器

2.17.1. 用法

uxmaster [*option*...]

2.17.2. 描述

uxmaster是uxdb的一个废弃的别名。