# 优炫数据库2.1 备份与还原手册

# 优炫数据库2.1 备份与还原手册



## 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

本文档介绍数据库的备份与还原，为相关技术人员和用户提供了必要的参考。

## 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

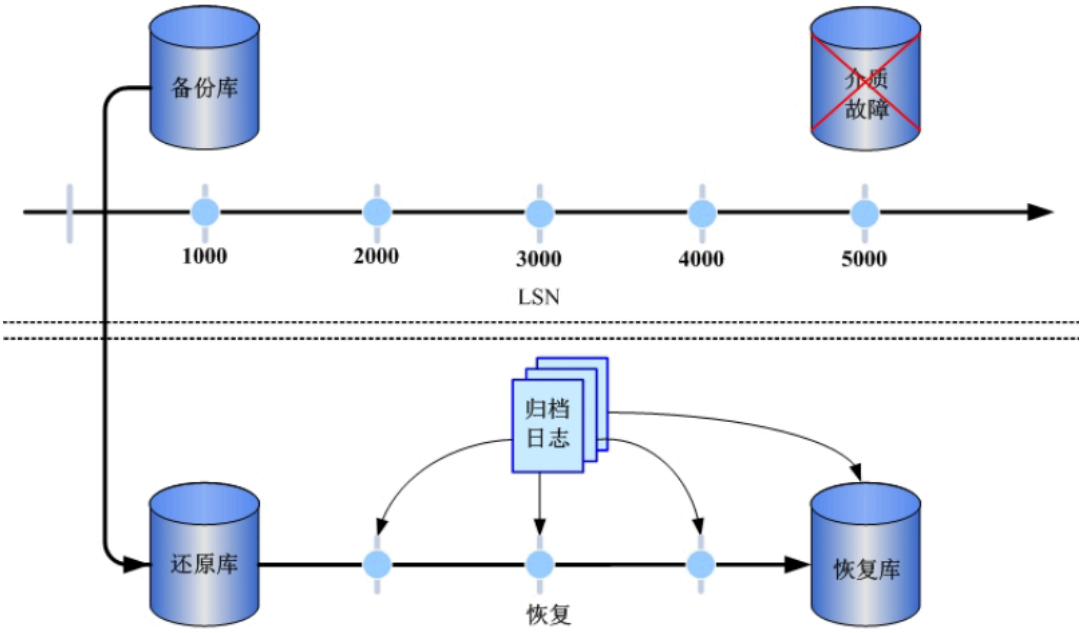
| 工具版本 | 发布日期 | 修改说明 |
| --- | --- | --- |
| 2.1.1.3 | 2021-01-06 | 第一次正式发布。 |

# 第 1 章 概述

UXDB数据库中的数据存储在数据库的物理数据文件中，数据文件按照页、簇和段的方式进行管理，数据页是最小的数据存储单元。任何一个对UXDB数据库的操作，归根结底都是对某个数据文件页的读写操作。

因此，UXDB备份的本质就是从数据库文件中拷贝有效的数据页保存到备份集中，这里的有效数据页包括数据文件的描述页和被分配使用的数据页。而在备份的过程中，如果数据库系统还在继续运行，这期间的数据库操作并不是都会立即体现到数据文件中，而是首先以日志的形式写到归档日志中，因此，为了保证用户可以通过备份集将数据恢复到备份结束时间点的状态，就需要将备份过程中产生的归档日志也保存到备份集中。

还原与恢复是备份的反过程。还原是将备份集中的有效数据页重新写入目标数据文件的过程。恢复则是指通过重做归档日志，将数据库状态恢复到备份结束时的状态；也可以恢复到指定时间点和指定LSN（日志序列号）。恢复结束以后，数据库中可能存在处于未提交状态的活动事务，这些活动事务在恢复结束后的第一次数据库系统启动时，会由UXDB数据库服务器自动进行回滚。备份、还原与恢复的关系如下图所示：




由于包含有价值的数据，UXDB数据库应当被定期地备份。有以下不同的基本方法来备份UXDB数据库：

SQL转储、文件系统级别备份、连续归档、UXFS备份。

每种方法都有其优缺点，在下面的小节中将分别讨论。

# 第 2 章 SQL转储

本章主要介绍SQL转储的备份工具（ux_dump、ux_dumpall），恢复工具（uxsql、ux_restore），UXDB的基本备份与恢复操作以及大型数据库的转储与恢复操作。

> **注意**
>
> 由于UXDB社区版本不支持ux_dump、ux_dumpall工具，所以UXDB社区版本不支持SQL转储。

## 2.1. 使用ux_dump

SQL转储方法的思想是创建一个由SQL命令组成的文件，当把这个文件反馈给服务器时，服务器将利用其中的SQL命令重建与转储时状态一样的数据库。UXDB为此提供了工具ux_dump。这个工具的基本用法是：

ux_dump *dbname* > *dumpfile*

ux_dump把结果输出到标准输出。上述命令会创建一个文本文件，ux_dump可以用其他格式创建文件以支持并行和细粒度的对象恢复控制。

ux_dump是一个普通的UXDB客户端应用。这就意味着你可以在任何可以访问该数据库的远程主机上进行备份工作。但是ux_dump不会以任何特殊权限运行，具体说来，就是它必须要有需要备份的表的读权限，因此为了备份整个数据库必须以数据库超级用户运行（如果没有足够的权限备份整个数据库，可以使用-n *schema*或-t *table*等选项来备份数据库中能够访问的部分）。

要声明ux_dump连接哪个数据库服务器，使用命令行选项-h *host*和 -p *port*。默认主机是本地主机或UXHOST环境变量指定的主机。同理，默认端口是5432或环境变量UXPORT指定的端口号。

ux_dump默认使用与当前操作系统用户名同名的数据库用户名称进行连接。要使用其他用户名称，可以声明–U选项或设置环境变量UXUSER。请注意ux_dump的连接要通过客户认证机制。

ux_dump对于其他备份方法的一个重要优势是，ux_dump的输出可以很容易地在新版本的UXDB中载入，而文件系统级别备份和连续归档都是因服务器版本而异的。ux_dump也是唯一可以将数据库传送到不同机器架构上的方法，例如从一个32位服务器到一个64位服务器。

由ux_dump创建的备份在内部是一致的，这意味着转储的是ux_dump开始运行时的数据库快照，ux_dump运行过程中发生的更新将不会被转储。ux_dump工作的时候并不阻塞其他的对数据库的操作（但是会阻塞需要排它锁的操作，例如ALTER TABLE）。

## 2.2. 转储恢复

ux_dump生成的文本文件可以由uxsql程序读取。从转储中恢复的常用命令是：

uxsql *dbname* < *dumpfile*

其中*dumpfile*就是ux_dump命令的输出文件。这条命令不会创建数据库*dbname*，用户必须在执行uxsql前以数据库template0为模板创建（例如，用命令createdb -T template0 *dbname*）。uxsql支持类似ux_dump的选项用以指定要连接的数据库服务器和要使用的用户名。非文本文件转储可以使用ux_restore工具来恢复。

在开始恢复之前，目标库中必须已经存在转储库中对象的拥有者以及在对象上被授予权限的用户。如果它们不存在，那么恢复过程将无法将对象创建成具有原来的所属关系以及权限的状态。

默认情况下，uxsql脚本在遇到一个SQL错误后会继续执行。如果希望在遇到一个SQL错误后让uxsql退出，那么可以设置ON_ERROR_STOP变量来运行uxsql，这将使uxsql在遇到SQL错误后退出并返回状态3：

uxsql --set ON_ERROR_STOP=on dbname < infile

不管怎样，最终将只能得到一个部分恢复的数据库。作为另一种选择，可以指定让整个恢复作为一个单独的事务运行，这样恢复要么完全完成要么完全回滚。这种模式可以通过向uxsql传递-1（数字1）或--single-transaction命令行选项来指定。在使用这种模式时，即使是很小的一个错误也会导致运行了数小时的恢复被回滚。

由于ux_dump和uxsql拥有读写管道的能力，可以直接从一个服务器转储一个数据库到另一个服务器，例如：

ux_dump -h *host1 dbname* | uxsql -h *host2 dbname*

> **重要**
>
> ux_dump产生的转储是相对于template0。这意味着在template1中添加的任何语言、过程等都会被ux_dump转储。如果在恢复时使用的是一个自定义的template1，那么必须从template0创建一个空的数据库，正如上面的例子所示。
>
> 当恢复完成，建议在每个数据库上运行ANALYZE，这样优化器就有可用的统计数据。

# 2.3. 使用ux_dumpall

ux_dump每次只转储一个数据库，而且它不会转储关于角色或表空间（因为它们属于集群范畴）的信息。为了支持便捷地转储一个数据库集群的全部内容，提供了ux_dumpall程序。ux_dumpall备份给定集群中的每个数据库，并且也保留了集群内的所有数据，如角色和表空间定义。该命令的基本用法是：

ux_dumpall > *dumpfile*

转储的结果可以使用uxsql恢复：

uxsql -f *dumpfile* dbname

实际上，可以指定从任何现有的数据库开始恢复，但是如果将转储载入到一个空集群中则通常要用uxdb数据库。在恢复一个ux_dumpall转储时需要具有数据库超级用户访问权限，因为需要恢复角色和表空间信息。如果恢复包含表空间，请确保转储中的表空间路径适合于目标集群。

ux_dumpall运行时会重新创建角色、表空间和空数据库，然后为每一个数据库调用ux_dump。这意味着每个数据库内部是一致的，但是不同数据库的快照并不同步。

集群范围的数据可以使用ux_dumpall的--globals-only选项单独转储。如果在单个数据库上运行ux_dump命令，则有必要对集群进行完全备份。

# 2.4. 处理大型数据库

一些操作系统有最大文件大小限制，这在创建大型ux_dump输出文件时可能会出现问题。幸运的是，ux_dump可以写入标准输出，因此可以使用标准Linux工具解决这个潜在问题。有如下几种方法：

### 2.4.1. 压缩转储

使用压缩程序，例如gzip：

ux_dump *dbname* | gzip > *filename*.gz

恢复：

gunzip -c *filename*.gz | uxsql *dbname*

或者：

cat *filename*.gz | gunzip | uxsql *dbname*

### 2.4.2. 分割转储

split命令允许将输出分割成较小的文件以便能够适应底层文件系统的尺寸要求。例如让每一块的大小为1兆字节：

ux_dump *dbname* | split -b 1m - *filename*

恢复：

cat *filename** | uxsql *dbname*

### 2.4.3. 自定义转储

如果UXDB所在的系统上安装了zlib压缩库，自定义转储格式将在写出数据到输出文件时对其压缩。这种方式的优势是输出文件中的表可以被有选择地恢复。下面的命令使用自定义转储格式来转储一个数据库：

ux_dump -Fc *dbname* > *filename*

自定义格式的转储输出不是uxsql的脚本，只能通过ux_restore恢复：

ux_restore -d *dbname filename*

对于非常大型的数据库，需要将split配合压缩转储或自定义转储两种方法之一进行使用。

### 2.4.4. 并行转储

为了加快转储大型数据库的速度，可以使用ux_dump的并行模式。它将同时转储多个表。-j参数控制并行度。并行转储只支持“目录”归档格式。

ux_dump -j *num* -F d -f *out.dir dbname*

ux_restore -j以并行方式恢复转储。它只能适合于“自定义”归档或者“目录”归档，不论归档是否由ux_dump -j创建。

# 第 3 章 文件系统级别备份

不同于SQL转储的另外一种备份策略是文件系统级别备份。

## 3.1. 文件系统级别冷备份

直接复制UXDB用于存储数据库中数据的文件：

- 本地集群

  tar -jcv -f backup.tar.bz2 `clusterdir`（本地集群所在路径）

  恢复：

  tar -jxv -f backup.tar.bz2 -C /home/uxdb/uxdbinstall/dbsql/bin

- 分布式集群

  mkdir /mnt/volume
  mount.uxfs `dirhost`：port/`volume`（分布式集群所在volume名称） /mnt/volume
  tar -jcv -f backup01.tar.bz2 /mnt/volume/`uxdbuxfs`（分布式集群挂载路径）

  恢复：

  tar -jxv -f backup01.tar.bz2 -C /mnt/volume/

  这种方法有两个限制：

  1. 为了得到一个可用的备份，数据库服务器必须被关闭。禁止所有连接等不彻底的措施是无用的（不仅因为tar和类似的工具在备份时并不对文件系统的状态做原子快照，而且服务器内部存在缓冲数据）。同样，恢复数据之前也需要关闭服务器。

  2. 文件级别冷备份的策略不适用于希望通过表和数据库相应的文件或目录来备份或恢复特定的表和数据库。因为包含在这些文件中的信息只有配合提交日志文件（`ux_xact/*`）才有用，提交日志文件包含了所有事务的提交状态。表文件只有和这些信息一起才有用。当然也不可能只恢复一个表及相关的ux_xact数据，因为这会导致数据库集群中所有其他表变得无用。因此文件级别冷备份只适合于完整地备份或恢复整个数据库集群。

## 3.2. rsync备份

还有一种选择是使用rsync来执行文件系统级别备份。其做法是先在数据库服务器运行时执行rsync（`rsync -r clusterdir backupdir`），然后关闭数据库服务器足够长时间接着执行rsync（`rsync --checksum -r clusterdir backupdir`）（--checksum是必需的，因为rsync的文件修改时间粒度只能精确到秒）。第二次执行rsync会比第一次快，因为这时只需要传送相对较少的数据，由于服务器是停止的，所以最终结果将是一致的。这种方法允许在最小停机时间内进行一次文件级别备份。

一个文件级别备份通常会比一个SQL转储体积更大（例如ux_dump不需要转储索引的内容，而是转储用于重建索引的命令）。但是，做一次文件级别备份可能更快。

# 第 4 章 连续归档和时间点恢复（PITR）

在任何时间，UXDB在数据集群目录的ux_wal/子目录下都有一个预写式日志（WAL）。这个日志存在的目的是为了保证系统崩溃后的安全：如果系统崩溃，可以“重放”从最后一次检查点以来的日志项来恢复数据库的一致性。该日志的存在使第三种备份数据库的策略变得可能：将文件系统级别的备份和WAL文件的备份结合起来。当需要恢复时，首先恢复文件系统备份，然后从备份的WAL文件中重放来把系统带到一个当前状态。这种方法比之前的方法管理起来要更复杂，但是有其显著的优点：

1. 不需要一个完全一致的文件系统备份作为开始点。备份中的任何内部不一致性将通过日志重放（这和崩溃恢复期间发生的并无显著不同）来修正。因此不需要文件系统快照功能，只需要tar或一个类似的归档工具。

2. 由于可以结合一个无穷长的WAL文件序列用于重放，可以通过简单地归档WAL文件来达到连续备份。这对于大型数据库特别有用，因为在其中不方便频繁地进行完全备份。

3. 并不需要一直重放WAL一直到最后。WAL重放可以在任何点停止重放，得到数据库在当时的一致快照。这样，该技术支持时间点恢复：在得到你的基础备份以后，可以将数据库恢复到它在其后任何时间的状态。

4. 如果连续地将一系列WAL文件输送给另一台已经载入了相同基础备份文件的机器，就得到了一个热备份系统：在任何时间点都能提出第二台机器，它是数据库的当前副本。

> **注意**
>
> ux_dump和ux_dumpall不会产生文件系统级别的备份，并且不能用于连续归档。这类转储是逻辑的并且不包含足够的信息用于WAL重放。

就简单的文件系统备份技术来说，这种方法只能支持整个数据库集群的恢复，却无法支持其中一个子集的恢复。另外，它需要大量的归档存储：一个基础备份的体积可能很庞大，并且一个繁忙的系统将会产生大量需要被归档的WAL流量。尽管如此，在很多需要高可靠性的情况下，它是首选的备份技术。

要使用连续归档成功地恢复，需要从基础备份时间开始的连续的归档WAL文件序列。为了开始，在建立第一个基础备份之前，应该建立并测试用于归档WAL文件的过程。如下首先讨论归档WAL文件的机制。

在编写此文档时，连续归档技术存在一些限制。这可能会在未来的发布中被修复：

- 如果在基础备份时执行CREATE DATABASE命令，然后在基础备份进行时CREATE DATABASE所复制的模板数据库被修改，恢复中可能会导致这些修改也被传播到已创建的数据库中。为了避免这种风险，最好不要在创建基础备份时修改任何模板数据库。

- 创建表空间的命令以绝对路径的形式存储在日志中，因此将作为具有相同绝对路径的表空间创建来重播。当日志在一台不同的机器上被重放时，这可能也不是用户希望的。即使在同一台机器上重播日志到一个新的数据目录中也可能是危险的，重播仍然会覆盖原始表空间的内容。为了避免这种潜在的危险，最佳做法是在创建或删除表空间后创建新的基础备份。

> **注意**
>
> 还需要注意的是，因为包括很多磁盘页快照，默认的WAL格式相当庞大。这些页快照用于支持崩溃恢复，修复断裂的磁盘页。依靠系统硬件和软件，页断裂的风险小到可以忽略，在此种情况下可以通过使用full_page_writes参数关闭

> 页快照来显著降低归档日志的总容量。关闭页快照并不会阻止使用日志进行PITR操作。同时，管理员可能希望通过尽可能增大检查点间隔参数来减少WAL中包含的页快照数量。

# 4.1. 建立WAL归档

一个运行中的UXDB系统产生一个无穷长的WAL记录序列。系统从物理上将这个序列划分成WAL 段文件，通常是每个16MB（段尺寸在配置UXDB时可修改）。段文件会被分配一个数字名称以便反映它在整个抽象WAL序列中的位置。在没有使用WAL归档时，系统通常只创建少量段文件，并且通过重命名不再使用的段文件为更高的段编号来“回收”它们。系统假设位于最后一个检查点之前的段文件的内容是无用的且可以被回收。

在归档WAL数据时，我们需要在每一个WAL段被写满时获取其内容，并且在段文件被回收重用之前保存该数据。依靠应用和可用的硬件，有很多不同的方法来保存数据：可以将段文件拷贝到另一台机器上的文件系统目录，或者将它们写出到一个磁盘驱动器（确保标识每个文件的原始文件名），或者将它们批量写入到CD上，或者其他方法。为了向数据库管理员提供灵活性，UXDB不对如何归档做任何假设。取而代之的是，UXDB需要数据库管理员声明一个shell命令将完整的段文件拷贝到需要的路径。该命令可以是简单的cp命令，也可以复杂到调用一个shell脚本。

要启用WAL归档，需设置wal_level配置参数为replica或更高，设置archive_mode为on，并且使用archive_command配置参数指定一个shell命令。实际上，这些设置总是被放置在uxsinodb.conf文件中。在archive_command中，%p会被将要归档的文件路径所替代，而%f会被文件名所替代（路径名是相对于当前工作目录而言的，即集群的数据目录）。如果需要在命令中嵌入一个真正的%字符，可以使用%%。最简单的命令类似于：

```
archive_command = 'test ! -f /home/uxdb/uxdbinstall/dbsql/bin/archivedir/%f && cp %p /home/uxdb/uxdbinstall/dbsql/bin/archivedir/%f'  # Linux
archive_command = 'copy "%p" "E:\\Program Files (x86)\\uxdb\\dbsql\\bin\\archivedir\\%f"'  # Windows
```

它将把 WAL 段拷贝到目录/home/uxdb/uxdbinstall/dbsql/bin/archivedir/（Linux）或者E:\\Program Files (x86)\\uxdb\\dbsql\\bin\\archivedir\\（Windows）（这个只是一个例子，并非建议的方法，可能不能在所有系统上都正确运行）。在%p和%f参数被替换之后，实际被执行的命令示例如下：

```
test ! -f /home/uxdb/uxdbinstall/dbsql/bin/archivedir/00000001000000A900000065 && cp ux_wal/00000001000000A900000065 /home/uxdb/uxdbinstall/dbsql/bin/archivedir/00000001000000A900000065
```

对每一个将要被归档的新文件都会生成一个类似的命令。

如果担心归档存储的尺寸过大，可以使用gzip来压缩归档文件：

```
archive_command = 'gzip < %p > /home/uxdb/uxdbinstall/dbsql/bin/archive/%f'
```

那么在恢复时将需要使用gunzip：

```
restore_command = 'gunzip < /home/uxdb/uxdbinstall/dbsql/bin/archivedir/%f > %p'
```

归档命令将在运行UXDB服务器的同一个用户的权限下执行。因为被归档的一系列WAL 文件包含数据库的所有东西，所以应该确保该用户的归档数据不会被别人窥探。 比如，归档到一个没有组或者全局读权限的目录里。

很多人选择使用脚本来定义archive_command，这样在uxsinodb.conf中该项非常简单：

```
archive_command = 'local_backup_script.sh "%p" "%f"'
```

任何时候如果希望在归档处理中使用多个命令，最好的方法是使用独立的脚本文件。允许使用流行的脚本语言来编写，例如bash或perl。

需要在脚本内解决的需求包括：

- 将数据拷贝到安全的集群外数据存储。
- 批处理WAL文件，例如每三小时被传输一次，而不是一次一个。
- 与其他备份和恢复软件交互。
- 与监控软件交互以报告错误。

> **提示**
>
> 在使用一个archive_command脚本时，最好启用logging_collector。任何从该脚本被写到stderr的消息将出现在数据库服务器日志中，这允许在复杂配置失败后能更容易诊断错误。

> **重要**
>
> 当且仅当归档命令成功时，才返回零退出。在得到零值结果之后，UXDB将假设该文件已经成功归档，因此它稍后将被删除或者被新的数据覆盖；得到非零值表示该文件没有被归档，因此它会周期性地重试直到成功。

归档命令通常应该拒绝覆盖已经存在的归档文件。这是一个非常重要的安全特性，可以在管理员操作失误（比如把两个不同的服务器的输出发送到同一个归档目录）的时候保持归档的完整性。

建议首先测试使用到归档命令，以保证实际上不会覆盖现有的文件，并且在这种情况下它返回非零状态。以上Linux中的命令例子通过包含一个独立的test步骤来保证这一点。在Linux平台上，cp具有诸如-n、-i的选项，可更简洁地完成测试，但是在没有验证返回的退出状态正确之前不能依赖它们。

在进行归档设置时，请考虑归档命令不停失败的情况：有些情况要求操作者的干涉，有些是归档空间不够了。例如，如果往磁带机上写，但没有自动换带机，那么就有可能发生这种情况；如果磁带满了，除非换磁带，否则归档无法成功。应该确保任何错误情况或者任何要求操作员干涉的情况都会被正确报告，这样才能迅速解决这些问题。否则ux_wal/目录会不停地被WAL段文件填充，直到问题解决（如果包含ux_wal/的文件系统被填满，UXDB将会关闭服务器。不会有未提交事务被丢失，但是数据库将会保持离线直到释放一部分空间）。

归档命令的速度并不强制要求，只要不低于服务器生成WAL数据的平均速度即可。即使归档进程稍微落后，正常的操作也会继续进行。如果归档进程慢很多，就会增加灾难发生时丢失的数据量。这同时也意味着ux_wal/目录包含大量未归档的段文件，并且可能最后超出了可用磁盘空间。建议监控归档进程，确保归档过程按照期望运转。

被归档的文件名最长为64个字符并且可以包含 ASCII 字母、数字以及标点符号的任意组合。不需要保持原始的相对路径（%p），但是有必要保持文件名（%f）。

> **注意**
>
> 请注意尽管WAL归档允许恢复任何对UXDB数据库中数据所做的修改，但它不会恢复对配置文件的修改（即uxsinodb.conf、ux_hba.conf和

> ux_ident.conf），因为这些文件都是手动编辑的，而不是通过SQL操作来编辑的。所以需要把配置文件放在一个日常文件系统备份过程可处理的位置。

归档命令只会被完成的WAL段调用。因此如果服务器产生了一点点WAL流量（或者在产生时有宽松的周期），从一个事务完成到它被安全地记录在归档存储中之间将会有较长的延迟。要为未归档数据设置一个时间限制，设置archive_timeout来强制要求服务器按照其设定的频度切换到一个新的WAL段。注意由于强制切换而被归档的文件还是具有和完全归档的文件相同的长度。因此设置一个很短的archive_timeout是很不明智的—将会占用归档存储空间。将archive_timeout设置为1分钟左右通常是合理的。

同样，如果希望确保一个刚刚完成的事务能被尽快归档，可以使用ux_switch_xlog进行一次手动段切换。其他与WAL管理相关的使用函数如下表所示：

表 **4.1.** 备份控制函数

| 名称 | 返回类型 | 描述 |
|---|---|---|
| ux_create_restore_point(name text) | ux_lsn | 为执行恢复创建一个命名点（只限于超级用户）。 |
| ux_current_wal_flush_lsn() | ux_lsn | 得到当前的预写式日志刷写位置。 |
| ux_current_wal_insert_lsn() | ux_lsn | 获得当前预写式日志插入位置。 |
| ux_current_wal_lsn() | ux_lsn | 获得当前预写式日志写入位置。 |
| ux_start_backup(label text [, fast boolean [, exclusive boolean ]]) | ux_lsn | 准备执行在线备份（只限于超级用户或者复制角色）。 |
| ux_stop_backup() | ux_lsn | 完成执行排他的在线备份（默认只限于超级用户或者复制角色， 但是可以授予其他用户 EXECUTE 特权来执行该函数）。 |
| ux_stop_backup(exclusive boolean [, wait_for_archive boolean ]) | setof record | 结束执行排他或者非排他的在线备份 （默认只限于超级用户， 但是可以授予其他用户 EXECUTE 特权来执行该函数）。 |
| ux_is_in_backup() | bool | 如果一个在线排他备份仍在进行中则为真。 |
| ux_backup_start_time() | timestamp with time zone | 获得一个进行中的在线排他备份的开始时间。 |
| ux_switch_wal() | ux_lsn | 强制切换到一个新的预写式日志文件（只限于超级用户）。 |
| ux_walfile_name(lsn ux_lsn) | text | 转换预写式日志位置为文件名。 |
| ux_walfile_name_offset(lsn ux_lsn) | text, integer | 转换预写式日志位置为文件名以及文件内的十进制字节偏移。 |
| ux_wal_lsn_diff(lsn ux_lsn, lsn ux_lsn) | numeric | 计算两个预写式日志位置间的差别。 |

当wal_level为minimal时，一些SQL命令被优化为避免记录WAL日志。在这些语句的其中之一的执行过程中如果打开了归档或流复制，WAL中将不会包含足够的信息用于归档恢复（崩

溃恢复不受影响）。出于这个原因，wal_level只能在服务器启动时修改。但是，archive_command可以通过重载配置文件来修改。如果你希望暂时停止归档，一种方式是将archive_command设置为空串（''）。这将导致WAL文件积累在ux_wal/中，直到一个可用的archive_command被重新建立。

# 4.2. 使用ux_basebackup制作基础备份

创建基础备份最简单的方法是使用ux_basebackup工具。它将会以普通文件或tar归档的方式创建一个基础备份。如果需要比ux_basebackup更高的灵活性，也可以使用低级API制作基础备份。

通常没有必要关心创建一个基础备份所需的时间。但是，如果在禁用full_page_write的情况下运行服务器，那么备份运行时的性能会下降，因为在备份模式中，full_page_writes会被强制执行。

要使用备份，需要保留所有在文件系统备份期间及之后生成的WAL段文件。为了为用户提供方便，基础备份过程会创建一个备份历史文件，它将被立刻存储到WAL归档区域。该文件以文件系统备份中需要的第一个WAL段文件命名。例如，如果开始的WAL文件是000000010000012340000055CD，则备份历史文件将被命名为000000010000012340000055CD.007C9330.backup.（文件名的第二部分表明WAL文件中的一个准确位置，一般可以被忽略）。一旦安全地归档了文件系统备份和在备份过程中被使用的WAL段文件（如备份历史文件中所指定的），所有名称在数字上低于备份历史文件中记录值的已归档WAL段对于恢复文件系统备份就不再需要了，可以被删除。但是应该考虑保持多个备份集以绝对保证能够恢复数据。

备份历史文件是一个很小的文本文件。它包含指定给ux_basebackup的标签字符串、备份的起止时间以及起止WAL段。如果使用该标签来标识相关转储文件，则已归档的备份历史文件足以说明需要哪个转储文件进行恢复。

由于不得不保存最后一次基础备份之后的所有归档WAL文件，基础备份之间的间隔通常应该根据希望在归档WAL文件上花费的存储空间来设定。也应该考虑准备花多长时间来进行恢复，如果需要恢复—系统将不得不重放所有那些WAL段，如果这些WAL段覆盖了最后一次基础备份以后的很长时间，重放过程将会花费较长时间。

> **注意**
>
> 由于UXDB社区版本不支持ux_basebackup工具，所以UXDB社区版本不支持使用ux_basebackup制作基础备份。

# 4.3. 使用低级API制作基础备份

使用低级API制作一个基础备份的过程比ux_basebackup方法要包含更多的步骤，但操作相对简单。这些步骤要按照顺序执行，并且在执行下一步之前要验证上一步是否成功。

可以用非排他或者排他的方法来制作低级基础备份。推荐非排他方法，而排他的方法已经被废弃并且最终将被移除。

## 4.3.1. 制作一个非排他低级备份

非排他低级备份允许其他并发备份运行（既包括那些使用同样的备份API开始的备份，也包括那些使用ux_basebackup开始的备份）。

1. 确保WAL归档被启用且正常工作。

2. 作为一个具有运行ux_start_backup权限的用户（超级用户，或者被授予权限执行该函数的用户）连接到服务器（任何数据库均可）并且执行命令：

SELECT ux_start_backup('label', false, false);

其中label是用来唯一标识这次备份操作的任意字符串。调用ux_start_backup的连接必须被保持到备份结束，否则备份将被自动中止。

默认情况下，ux_start_backup可能需要较长的时间完成。这是因为它会执行一个检查点，并且该检查点所需要的I/O将会花费一段时间，默认情况下是检查点间隔（由配置参数checkpoint_completion_target设置）的一半。默认情况下对查询处理的影响最小。如果想要尽可能快地开始备份，请把第二个参数改成true，这将使用尽可能多的I/O发出即时检查点。

第三个参数为false会告诉ux_start_backup开始一次非排他基础备份。

3. 使用文件系统备份工具（例如tar或者cpio，而不是ux_dump或者ux_dumpall）执行备份。此时，不需要停止正常的数据库操作。在这类备份期间需要注意的事项请参见备份数据目录。

4. 在同一个连接中，执行命令：

   SELECT * FROM ux_stop_backup(false);

   该函数终止备份模式。在主服务器上，将自动切换到下一个WAL段。在备服务器上，无法自动切换WAL段，可以在主服务器上运行ux_switch_wal以执行手动切换。切换的原因是让备份间隔期间写入的最后一个WAL段文件准备归档。

   ux_stop_backup将返回一个具有三个值的行。第二个返回值应该被写入到该备份根目录中名为backup_label的文件。第三个返回值应该被写入到一个名为tablespace_map的文件，除非该域为空。这些文件对该备份正常工作来说是至关重要的，不能被随意修改。

5. 一旦备份期间活动的WAL段文件被归档，备份就完成了。由ux_stop_backup的第一个返回值标识的文件是构成一个完整备份文件集合所需的最后一个段。在主服务器上，如果archive_mode被启用，并且wait_for_archive参数为true，则最后一个段被归档后ux_stop_backup的结果才会被返回。在备用服务器上，为了使ux_stop_backup等待，archive_mode必须是always。由于已经配置了archive_command，WAL段文件的归档会自动发生。在大部分情况下，这些归档会很快发生，但是建议监控归档系统确保没有延迟。如果归档进程由于归档命令的失败而落后，它将会持续重试直到归档成功并且备份完成。如果希望对ux_stop_backup的执行给出一个时间限制，可以设置一个合适的statement_timeout值，但要注意如果ux_stop_backup因此而中止可能会致使备份失效。

   如果对备份进程监控并确保备份所需的所有WAL段文件已成功归档，则可以将wait_for_archive参数（默认为true）设置为false以使停止备份记录写入WAL，ux_stop_backup立即返回。默认情况下，ux_stop_backup将一直等到所有WAL归档完毕，这可能需要等待一些时间。必须谨慎使用此选项：如果未正确监控WAL归档，则备份可能未包括所有WAL文件，因此将不完整且无法恢复。

## 4.3.2. 制作一个排他低级备份

一个排他备份的处理绝大部分都和非排他备份相同，但是在一些关键步骤上不同。这种类型的备份只能在主服务器上进行，并且不允许并发备份。

1. 确保WAL归档被启用且正常工作。

2. 作为一个具有运行ux_start_backup权限的用户（超级用户，或者被授予权限执行该函数的用户）连接到服务器（任何数据库均可）并且执行命令：

   SELECT ux_start_backup('label');

   其中label是用来唯一标识这次备份操作的任意字符串。该函数会在集群目录中创建一个关于备份信息的备份标签文件（也被称为backup_label，其中包括了开始时间和标签字符

串）和名为tablespace_map的表空间映射文件（文件包含在ux_tblspc/中一个或者多个表空间符号链接的信息）。如果需要从备份中恢复，这两个文件对于备份的完整性都至关重要。

默认情况下，ux_start_backup可能需要较长的时间完成。这是因为它会执行一个检查点，并且该检查点所需要的I/O将会花费一段时间，默认情况下是检查点间隔（由配置参数checkpoint_completion_target设置）的一半。默认情况下对查询处理的影响最小。如果你要尽快开始备份，可使用如下命令使检查点尽可能快地被完成：

SELECT ux_start_backup('label', true);

3. 使用文件系统备份工具（例如tar或者cpio，而不是ux_dump或者ux_dumpall）执行备份。此时，不需要停止正常的数据库操作。在这类备份期间需要注意的事项请参见备份数据目录。

> **注意**
>
> 如果服务器在备份期间崩溃，从UXDATA目录手动删除backup_label文件后，才可能成功重新启动服务器。

4. 在同一个连接中，执行命令：

   SELECT ux_stop_backup();

   该函数终止备份模式，并且自动切换到下一个WAL段。进行切换的原因是使在备份期间生成的最新WAL段可归档。

5. 一旦备份期间活动的WAL段文件被归档，备份就完成了。由ux_stop_backup的返回值标识的文件是构成一个完整备份文件集合所需的最后一个段。如果archive_mode被启用，则最后一个段被归档后ux_stop_backup的结果才会被返回。在备用服务器上，为了使ux_stop_backup等待，archive_mode必须是always。由于已经配置了archive_command，WAL段文件的归档就会自动发生。在大部分情况下，这些归档会很快发生，但是建议监控归档系统确保没有延迟。如果归档进程由于归档命令的失败而落后，它将会持续重试直到归档成功并且备份完成。如果希望对ux_stop_backup的执行给出一个时间限制，可以设置一个合适的statement_timeout值，但要注意如果ux_stop_backup因此而中止可能会致使备份失效。

## 4.3.3. 备份数据目录

如果被拷贝的文件在拷贝过程中发生变化，某些文件系统备份工具会发出警告或错误。在建立一个活跃数据库的基础备份时，这种情况是正常的，并非一个错误。此时，需要确保能够区分它们和真正的错误。例如，某些版本的rsync为“消失的源文件”返回一个独立的退出码，可以编写一个驱动脚本将该退出码作为一种非错误情况接受。同样，如果一个文件在被tar复制的过程中被截断，某些版本的GNU tar会返回一个与致命错误无法区分的错误代码。幸运的是，如果一个文件在备份期间被改变，版本为1.16及其后的GNU tar将会退出并返回1，而对于其他错误返回2。在版本1.23及其后的GNU tar中，可以使用警告选项--warning=no-file-changed --warning=no-file-removed来隐藏相关的警告消息。

确认备份包含数据库集群目录（例如/home/uxdb/uxdbinstall/dbsql/bin/uxdblocal）下的所有文件。如果使用了不在此目录下的表空间，注意也把它们包括在内（并且确保备份将符号链接归档为链接，否则恢复过程将破坏表空间）。

从基础备份中移除集群的ux_wal/子目录中的文件，这种微小的调整是必要的，因为它降低了恢复时的错误风险。也可以移除uxmaster.pid和uxmaster.opts，它们记录了关于uxmaster运行的信息，但与最终使用这个备份的uxmaster无关（这些文件可能会使ux_ctl搞混淆）。

从基础备份中移除集群的ux_replslot/子目录中的文件也是个好主意，这样主控机上存在的复制槽不会成为备份的一部分。否则，后续用该备份创建一个后备机可能会导致该后备机上的

WAL文件被无限期保留；并且在启用了热后备反馈的情况下可能导致主控机膨胀，因为使用那些复制槽的客户端将继续连接到主控机（而不是后备机）并且继续更新复制槽。即使该备份是要被用来创建一个新的主控机，拷贝复制槽也不是特别有用，因为这些槽的内容在新主控机上线时很可能已经过时。

ux_dynshmem/、ux_notify/、ux_serial/、ux_snapshots/、ux_stat_tmp/和ux_subtrans/目录的内容（但不是目录本身）可以从备份中省略，因为它们将在uxmaster启动时初始化。如果设置了stats_temp_directory并且是在数据目录下，那么也可以省略该目录的内容。

任何以uxsql_tmp开头的文件或目录都可以从备份中省略。这些文件在uxmaster启动时被删除，并且根据需要重新创建目录。

备份标签文件包含指定给ux_start_backup的标签字符串，以及ux_start_backup被运行的时刻和起始WAL文件的名称。在发生混乱的情况下就可以在备份文件中查看并准确地决定该转储文件来自于哪个备份会话。表空间映射文件包括存在于目录ux_tblspc/中的符号链接名称以及每一个符号链接的完整路径。这些文件不仅是为了提供参考，它们的存在和内容对于系统恢复过程的正确操作至关重要。

在服务器停止时也可以创建一个备份。在这种情况下，显然不能使用ux_start_backup或ux_stop_backup，因此只能用户自身跟踪备份，以及相关WAL文件应该追溯到多久以前。通常最好遵循上面的连续归档过程。

# 4.4. 连续归档备份恢复

从连续归档备份进行恢复。过程如下：

1. 如果服务器仍在运行，首先停止服务器。
2. 如果空间足够，将整个集群数据目录和表空间复制到一个临时位置，稍后可能用到它们。注意这种预防措施要求系统有足够的空闲空间来保留现有数据库的两份拷贝。如果没有足够的空间，至少要拷贝集群的ux_wal子目录的内容，因为它可能包含系统停止之前还未被归档的日志。
3. 移除所有位于集群数据目录和正在使用的表空间根目录下的文件和子目录。
4. 从文件系统备份中恢复数据库文件。注意使用正确的系统用户恢复（数据库系统用户，而不是root）并且使用正确的权限。如果使用表空间，应该验证ux_tblspc/中的符号链接被正确地恢复。
5. 移除ux_wal/中的所有文件，这些是来自于文件系统备份而不是当前日志，因此可以被忽略。如果根本没有归档ux_wal/，那么以正确的权限重建它。注意如果以前它是一个符号链接，请确保以同样的方式进行重建。
6. 如果有第2步中保存的未归档WAL段文件，把它们拷贝到ux_wal/（最好是拷贝而不是移动，这样如果在开始恢复后出现问题仍然有未修改的文件）。
7. 临时修改ux_hba.conf，阻止普通用户在成功恢复之前连接。
8. 启动服务器。服务器将会进入到恢复模式进而根据需要读取归档WAL文件。恢复可能因为一个外部错误而被终止，可以重新启动服务器，这样它将继续恢复。
9. 检查数据库的内容来确保已经恢复到了期望的状态。如果没有，返回到第1步。如果一切正常，通过恢复ux_hba.conf允许普通用户连接。

所有这些的关键部分是设置恢复配置命令，它描述将如何恢复以及恢复要运行到什么程度，在uxsinodb.conf中指定restore_command，告诉UXDB如何获取归档WAL文件段。与archive_command相似，这也是一个shell命令字符串。它可以包含%f（被日志文件名替换）和%p（将被日志文件被拷贝的目标路径名替换）。（路径名是相对于当前工作目录的，即集群的数据目录）。如果需要在命令中嵌入一个真正的%字符，可以写成%%。最简单的命令如下：

restore_command = 'cp /home/uxdb/uxdbinstall/dbsql/bin/archivedir/%f %p'

它将从目录/home/uxdb/uxdbinstall/dbsql/bin/archivedir中拷贝之前归档的WAL段。当然，可以使用更复杂的shell脚本，例如一个要求操作者装载合适磁带的shell脚本。

命令失败将返回非零退出状态。命令被调用请求存档中不存在的文件而返回非零的情况不属于错误情况。恢复失败的一种例外是该命令被一个信号（被用作数据库服务器关闭动作的一部分，除了SIGTERM）终止或者被shell的错误（例如命令未找到）终止，那样恢复将中止并且服务器将不会启动。

并非所有被请求的文件都是WAL段文件，也许还会请求一些.backup或.history后缀的文件。还要注意的是，%p路径的基本名字和%f不同，不可以互换。

归档中找不到的WAL段可以在ux_wal/中找到，这使得可以使用最近未归档的段。但是，在归档中可用的段优先于ux_wal/中的文件被使用。

通常，恢复将会处理完所有可用的WAL段，从而将数据库恢复到当前时间点（或者尽可能接近给定的可用WAL段）。因此，一个正常的恢复将会以一个“文件未找到”消息结束，错误消息的准确文本取决于用户设置的restore_command。也可能在恢复的开始看到一个针对名称类似于00000001.history文件的错误消息，这也是正常的，对此情况的讨论见时间线。

如果希望恢复到之前的某个时间点（例如，恢复到DBA删除主事务表之前），只需要在uxsinodb.conf中指定要求的停止点。可以使用日期/时间、特定恢复点或指定事务ID完成时间来定义停止点（也被称为“恢复目标”）。在这种写法中，只有日期/时间和特定恢复点选项非常有用，因为没有工具可以准确地确定要用哪个事务ID。

> **注意**
>
> 停止点必须位于基础备份的完成时间之后，即ux_stop_backup的完成时间。不能使用基本备份恢复到备份正在进行时的状态（要恢复到这样的时间，必须返回到以前的基本备份点并从那里向前恢复）。

如果恢复找到被破坏的WAL数据，恢复将会停止于该点并且服务器不会启动。在这种情况下，恢复进程需要从开头重新开始运行，并指定一个在损坏点之前的“恢复目标”以便恢复能够正常完成。如果恢复由于一个外部原因失败，例如一个系统崩溃或者WAL归档变为不可访问，可以重启服务器并且该次恢复将会从上次失败的地方继续。恢复重启的工作方式与正常操作中的检查点非常相似：服务器周期性地强制把它的所有状态写到磁盘中，然后更新ux_control文件以表明已经处理的WAL数据无需再次扫描。

# 4.5. 时间线

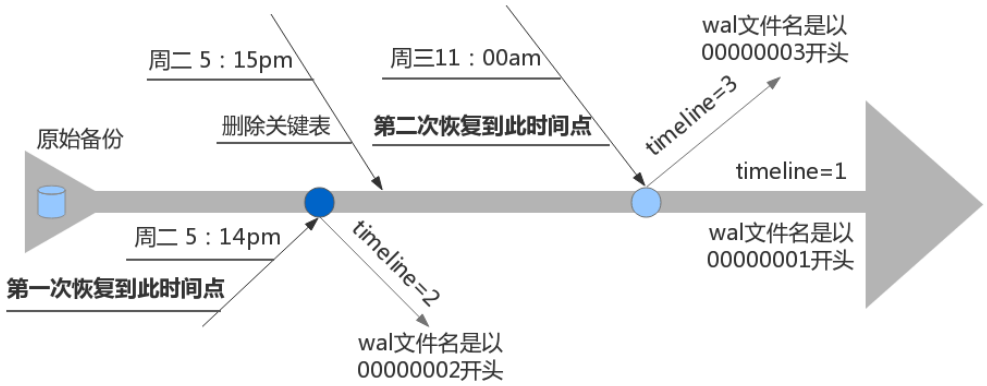
将数据库恢复到一个之前的时间点的能力带来了一些复杂性，这和有关时间旅行和平行宇宙的科幻小说有些相似。例如，在数据库的最初历史中，假设周二晚上5：15时删除了一个关键表，但是一直到周三中午才意识到错误。这时，可以备份恢复到周二晚上5：14的时间点，并上线运行。在数据库宇宙的这个历史中，从没有丢弃该表。但是假设后来意识到这并非一个好主意，并且想回到最初历史中周三早上11：00。此时已经没法这样做，在数据库在线运行期间，重写了某些WAL段文件，而这些文件本来可以将数据库返回到用户希望回到的时间。因此，为了避免出现这种状况，用户需要将完成时间点恢复后生成的WAL记录序列与初始数据库历史中产生的WAL记录序列区分开来。

要解决这个问题，UXDB有一个时间线概念。无论何时当一次归档恢复完成，一个新的时间线被创建来标识恢复之后生成的WAL记录序列。时间线ID号是WAL段文件名的一部分，因此一个新的时间线不会重写由之前的时间线生成的WAL数据。实际上可以归档很多不同的时间线。虽然这可能看起来是一个无用的特性，但是它常常扮演救命稻草的角色。考虑到用户不太确定需要恢复到哪个时间点的情况，可能不得不做多次时间点恢复尝试，直到最终找到从旧历史中分支出去的最佳位置。如果没有时间线，该处理将会很快成为一堆不可管理的混乱。而有了时间线，可以恢复到任何之前的状态，包括早先被放弃的时间线分支中的状态。

每次当一个新的时间线被创建，UXDB会创建一个“时间线历史”文件，它显示了新时间线是什么时候从哪个时间线分支出来的。系统在从一个包含多个时间线的归档中恢复时，这些历史文件对于允许系统选取正确的WAL段文件非常必要。因此，和WAL段文件相似，它们也要被归档到WAL归档区域。历史文件是很小的文本文件，因此将它们无限期地保存起来的代价很小，而且也是很合适的（而段文件都很大）。为了清晰明了，用户可以在历史文件中增加注释来记录如何和为什么要创建该时间线。当由于试验的结果拥有了一大堆错综复杂的不同时间线时，这种注释将会特别有价值。

恢复的默认行为是沿着相同的时间线进行恢复，该时间线是基础备份创建时的当前时间线。如果希望恢复到某个子时间线（即，回到在一次恢复尝试后产生的某个状态），需要在uxsinodb.conf中指定目标时间线ID。不能恢复到早于该基础备份之前分支出的时间线。

上述时间线具体如下如所示：




- 第一次恢复：

  1. 解压基础备份：

     tar -xvf backup.tar

  2. 配置uxsinodb.conf文件：

     vi backup/uxsinodb.conf

     restore_command = 'cp /mnt/server/archivedir/%f %p' //从归档目录恢复日志。
     recovery_target_time = '2018-12-11 17:14:00 ' //指定归档时间点，若未指定则恢复到故障前的最后一个完成的事务。
     recovery_target_timeline = 'latest' //指定归档时间线，‘latest’代表最新的时间线分支，如没指定恢复到故障前的ux_control中的时间线。

  3. 创建并配置标志文件standby.signal：

     vi standby.signal

     standby_mode = 'off' //打开后将会以备库身份启动，而不是即时恢复。

  4. 启动数据库：

     uxsql

     此时将会产生新的timeline，而且会生成一个新的history文件。

- 第二次恢复：

  1. 停止数据库，删除数据文件夹。

2. 解压基础备份：

   tar -xvf backup.tar

3. 配置uxsinodb.conf文件：

   vi backup/uxsinodb.conf

   restore_command = 'cp /mnt/server/archivedir/%f %p' //从归档目录恢复日志。
   recovery_target_time = '2018-12-12 11:00:00 ' //指定归档时间点，若未指定则恢复到故障前的最后一个完成的事务。
   recovery_target_timeline = '1' //指定归档时间线，如没指定恢复到故障前的ux_control中的时间线

4. 创建并配置标志文件standby.signal：

   vi standby.signal

   standby_mode = 'off' //打开后将会以备库身份启动，而不是即时恢复。

5. 启动数据库：

   uxsql

此时将再次产生新的timeline，而且会生成一个新的history文件。

# 4.6. 单机热备份

使用UXDB的备份功能可以进行单机热备份，但是该备份不能被用于时间点恢复。但是备份和恢复过程要比ux_dump转储快（由于文件尺寸一般也比ux_dump转储更大，所以在某些情况下速度优势可能会被否定）。

在基础备份的帮助下，单机热备份最简单的方式是使用ux_basebackup工具。如果在调用ux_basebackup时使用了-X参数，使用该备份所需的所有预写式日志将会被自动包含在该备份中，并且恢复该备份也不需要特殊的操作。

如果在复制备份文件时需要更多灵活性，也可以使用一个较底层的程序来创建单机热备份。底层单机热备份的准备工作：将wal_level设置为replica或更高，archive_mode设置为on，并且设置archive_command，该命令只在开关文件（/home/uxdb/uxdbinstall/dbsql/bin/backup_in_progress）存在时执行归档，否则会返回0值退出状态，命令如下：

archive_command = 'test ! -f /home/uxdb/uxdbinstall/dbsql/bin/backup_in_progress || (test ! -f /home/uxdb/uxdbinstall/dbsql/bin/archive/%f && cp %p /home/uxdb/uxdbinstall/dbsql/bin/archive/%f)'

通过上面的准备，可以使用如下所示的脚本来建立备份：

touch /home/uxdb/uxdbinstall/dbsql/bin/backup_in_progress
uxsql -c "select ux_start_backup('hot_backup');"
tar -cf /home/uxdb/uxdbinstall/dbsql/bin/backup.tar clusterdir（备份集群所在路径）
uxsql -c "select ux_stop_backup();"
rm /home/uxdb/uxdbinstall/dbsql/bin/backup_in_progress
tar -rf /home/uxdb/uxdbinstall/dbsql/bin/backup.tar -P /home/uxdb/uxdbinstall/dbsql/bin/archive/

开关文件/home/uxdb/uxdbinstall/dbsql/bin/backup_in_progress首先被创建，允许对已完成的WAL文件进行归档，备份完成之后开关文件被删除。归档的WAL文件则被加入到备份中，这样基础备份和所有需要的WAL文件都在同一个tar文件中。

# 第 5 章 UXFS备份

操作步骤示例环境为Linux操作系统，安装目录/home/uxdb/uxdbinstall。

> **注意**
>
> UXDB社区版、标准版不支持UXFS备份。

## 5.1. osd备份

多OSD是基于在线状态下扩容，即在不停机的情况下无限扩展可使用的磁盘的空间，理论上可以支持无限大的空间扩展。多个OSD的replication配置是指针对文件或者数据库等进行基本的复制备份的配置。

- 创建并挂载Volume：

  1. 进入安装目录下的uxfs/bin目录：

     cd /home/uxdb/uxdbinstall/uxfs/bin

  2. 创建volume：

     mkfs.uxfs localhost/demo

  3. 挂载：

     mkdir /mnt/volume
     mount.uxfs localhost/demo /mnt/volume

- 设置Volume备份属性：

  xtfsutil --set-drp --replication-policy WqRq --replication-factor 3 /mnt/volume

  --replication-factor在WqRq时至少为3，建议备份份数为服务器osd数量的一半向上取整。

## 5.2. mrc备份

示例三台机器IP分别如下：

192.168.0.122、192.168.0.123、192.168.0.124

1. 同步三台机器的系统时间。

2. 分别修改三台机器的mrcconfig.properties配置文件：

   cd /home/uxdb/uxdbinstall/uxfs/etc/xos/uxfs_conf
   vi mrcconfig.properties

   修改参数配置如下：

   babudb.sync = FDATASYNC
   babudb.plugin.0 = /home/uxdb/uxdbinstall/uxfs/etc/xos/uxfs_conf/server-repl-plugin/mrc.properties

   配置dir、mrc、osd的uuid使其各不相同。

3. 分别修改三台机器的mrc.properties配置文件：

   cd server-repl-plugin
   vi mrc.properties

   修改参数配置如下：

   babudb.repl.participant.0 = 192.168.0.122
   babudb.repl.participant.0.port = 35676
   babudb.repl.participant.1 = 192.168.0.123
   babudb.repl.participant.1.port = 35676
   babudb.repl.participant.2 = 192.168.0.124
   babudb.repl.participant.2.port = 35676
   babudb.repl.sync.n = 2
   plugin.jar = /home/uxdb/uxdbinstall/uxfs/java/lib/BabuDB_replication_plugin.jar
   babudb.repl.dependency.0 = /home/uxdb/uxdbinstall/uxfs/java/flease/dist/Flease.jar

4. 分别启动三台机器的uxfs：

   cd /home/uxdb/uxdbinstall/uxfs
   ./start-all.sh

# 5.3. dir备份

示例三台机器IP分别如下：

192.168.0.122、192.168.0.123、192.168.0.124

1. 同步三台机器的系统时间。

2. 分别修改三台机器的dirconfig.properties配置文件：

   cd /home/uxdb/uxdbinstall/uxfs/etc/xos/uxfs_conf
   vi dirconfig.properties

   修改参数配置如下：

   babudb.plugin.0 = /home/uxdb/uxdbinstall/uxfs/etc/xos/uxfs_conf/server-repl-plugin/dir.properties

   配置dir、mrc、osd的uuid使其各不相同。

3. 分别修改三台机器的dir.properties配置文件：

   cd server-repl-plugin
   vi dir.properties

   修改参数配置如下：

   babudb.repl.participant.0 = 192.168.0.122
   babudb.repl.participant.0.port = 35678
   babudb.repl.participant.1 = 192.168.0.123
   babudb.repl.participant.1.port = 35678
   babudb.repl.participant.2 = 192.168.0.124
   babudb.repl.participant.2.port = 35678
   babudb.repl.sync.n = 2
   plugin.jar = /home/uxdb/uxdbinstall/uxfs/java/lib/BabuDB_replication_plugin.jar
   babudb.repl.dependency.0 = /home/uxdb/uxdbinstall/uxfs/java/flease/dist/Flease.jar

4. 分别修改三台机器的mrcconfig.properties和osdconfig.properties配置文件：

cd /home/uxdb/uxdbinstall/uxfs/etc/xos/uxfs_conf
vi mrcconfig.properties
vi osdconfig.properties

修改参数配置如下：

dir_service.host = 192.168.0.122
dir_service.port = 32638
dir_service.1.host = 192.168.0.123
dir_service.1.port = 32638
dir_service.2.host = 192.168.0.124
dir_service.2.port = 32638

5. 分别启动三台机器的uxfs：

cd /home/uxdb/uxdbinstall/uxfs
./start-all.sh

# 5.4. dir、mrc和osd全备份

示例三台机器IP分别如下：

192.168.0.122、192.168.0.123、192.168.0.124

1. 同步三台机器的系统时间。
2. 配置dir replication：
   - 分别修改三台机器的dirconfig.properties配置文件：

     cd /home/uxdb/uxdbinstall/uxfs/etc/xos/uxfs_conf
     vi dirconfig.properties

     修改参数配置如下：

     babudb.plugin.0 = /home/uxdb/uxdbinstall/uxfs/etc/xos/uxfs_conf/server-repl-plugin/dir.properties

   - 分别修改三台机器的dir.properties配置文件：

     cd server-repl-plugin
     vi dir.properties

     修改参数配置如下：

     babudb.repl.participant.0 = 192.168.0.122
     babudb.repl.participant.0.port = 35678
     babudb.repl.participant.1 = 192.168.0.123
     babudb.repl.participant.1.port = 35678
     babudb.repl.participant.2 = 192.168.0.124
     babudb.repl.participant.2.port = 35678
     babudb.repl.sync.n = 2
     plugin.jar = /home/uxdb/uxdbinstall/uxfs/java/lib/BabuDB_replication_plugin.jar
     babudb.repl.dependency.0 = /home/uxdb/uxdbinstall/uxfs/java/flease/dist/Flease.jar

   - 分别修改三台机器的mrcconfig.properties和osdconfig.properties配置文件：

     cd /home/uxdb/uxdbinstall/uxfs/etc/xos/uxfs_conf
     vi mrcconfig.properties

vi osdconfig.properties

修改参数配置如下：

```
dir_service.host = 192.168.0.122
dir_service.port = 32638
dir_service.1.host = 192.168.0.123
dir_service.1.port = 32638
dir_service.2.host = 192.168.0.124
dir_service.2.port = 32638
```

3. 配置mrc replication：

   - 分别修改三台机器的mrcconfig.properties配置文件：

     ```
     cd /home/uxdb/uxdbinstall/uxfs/etc/xos/uxfs_conf
     vi mrcconfig.properties
     ```

     修改参数配置如下：

     ```
     babudb.sync = FDATASYNC
     babudb.plugin.0 = /home/uxdb/uxdbinstall/uxfs/etc/xos/uxfs_conf/server-repl-plugin/
     mrc.properties
     ```

     配置dir、mrc、osd的uuid使其各不相同。

   - 分别修改三台机器的mrc.properties配置文件：

     ```
     cd server-repl-plugin
     vi mrc.properties
     ```

     修改参数配置如下：

     ```
     babudb.repl.participant.0 = 192.168.0.122
     babudb.repl.participant.0.port = 35676
     babudb.repl.participant.1 = 192.168.0.123
     babudb.repl.participant.1.port = 35676
     babudb.repl.participant.2 = 192.168.0.124
     babudb.repl.participant.2.port = 35676
     babudb.repl.sync.n = 2
     plugin.jar = /home/uxdb/uxdbinstall/uxfs/java/lib/BabuDB_replication_plugin.jar
     babudb.repl.dependency.0 = /home/uxdb/uxdbinstall/uxfs/java/flease/dist/Flease.jar
     ```

4. 启动三台机器的uxfs：

   ```
   cd /home/uxdb/uxdbinstall/uxfs
   ./start-all.sh
   ```

5. 创建并挂载Volume，参考osd备份：

   - 进入安装目录下的uxfs/bin目录：

     ```
     cd /home/uxdb/uxdbinstall/uxfs/bin
     ```

   - 创建volume：

     ```
     mkfs.uxfs localhost/demo
     ```

   - 挂载：

     ```
     mkdir /mnt/volume
     mount.uxfs 192.168.0.122/demo /mnt/volume
     ```

6. 配置osd replication：

   xtfsutil --set-drp --replication-policy WqRq --replication-factor 5 /mnt/volume

# 第 6 章 Hot-Standby环境搭建

> 注意
>
> 由于UXDB社区版本不支持ux_basebackup工具，所以UXDB社区版本不支持同步流复制和异步流复制。

## 6.1. 同步流复制

表 **6.1.** 同步流复制不同机器配置

| 角色 | **IP** | 数据目录 |
| --- | --- | --- |
| 主机 | 192.168.0.122 | /home/uxdb/uxdbinstall/dbsql/bin/test |
| 备机01 | 192.168.0.123 | /home/uxdb/uxdbinstall/dbsql/bin/test122 |
| 备机02 | 192.168.0.124 | /home/uxdb/uxdbinstall/dbsql/bin/test122 |

1. 主机配置：

```
vi /home/uxdb/uxdbinstall/dbsql/bin/test/ux_hba.conf
 host  replication  uxdb  0.0.0.0/0  md5
vi /home/uxdb/uxdbinstall/dbsql/bin/test/uxsinodb.conf
 listen_addresses = '*'
max_wal_senders = 10
 wal_level = replica
 synchronous_standby_names = 'standby01,standby02'
ux_ctl reload -D /home/uxdb/uxdbinstall/dbsql/bin/test
```

2. 备机01配置：

   - 生成基础备份：

     ```
     ux_basebackup -h 192.168.0.122 -U uxdb -F p -P -R -D /home/uxdb/uxdbinstall/dbsql/bin/test122
     ```

   - 创建并配置标志文件standby.signal：

     ```
     vi /home/uxdb/uxdbinstall/dbsql/bin/test122/standby.signal
     standby_mode = 'on'
     ```

   - 配置uxsinodb.auto.conf：

     ```
     vi /home/uxdb/uxdbinstall/dbsql/bin/test122/uxsinodb.auto.conf
     primary_conninfo = 'application_name=standby01 user=uxdb password=123456
      host=192.168.0.122 port=5432 sslmode=prefer sslcompression=1 target_session_attrs=any'
     ```

   - 启动：

     ```
     vi /home/uxdb/uxdbinstall/dbsql/bin/test122/uxsinodb.conf
      hot_standby=on
     ux_ctl -D /home/uxdb/uxdbinstall/dbsql/bin/test122 start
     ```

3. 备机02配置：

- 生成基础备份：

  ux_basebackup -h 192.168.0.122 -U uxdb -F p -P -R -D /home/uxdb/uxdbinstall/dbsql/bin/test122

- 创建并配置标志文件standby.signal：

  vi /home/uxdb/uxdbinstall/dbsql/bin/test122/standby.signal
  standby_mode = 'on'

- 配置uxsinodb.auto.conf：

  vi /home/uxdb/uxdbinstall/dbsql/bin/test122/uxsinodb.auto.conf
  primary_conninfo = 'application_name=standby01 user=uxdb password=123456
  host=192.168.0.122 port=5432 sslmode=prefer sslcompression=1 target_session_attrs=any'

- 启动：

  vi /home/uxdb/uxdbinstall/dbsql/bin/test122/uxsinodb.conf
  hot_standby=on
  ux_ctl -D /home/uxdb/uxdbinstall/dbsql/bin/test122 start

4. 验证结果：

   关掉备机01，主机可以正常操作；

   关掉备机01和02，主机的非更新操作正常，更新操作夯住，启动任一备机后，夯住的操作可以继续。

   关掉主机，如果此时更新操作正在同步，正在等待的事务将在主机恢复时被记为完全同步。没有办法确认所有备机是否已经收到了在主机崩溃时所有还未处理的WAL 数据。即使在主机上显示为已提交，某些事务可能不会在后备服务器上被提交。

# 6.2. 异步流复制

表 **6.2.** 异步流复制不同机器配置

| 角色 | **IP** | 数据目录 |
| --- | --- | --- |
| 主机 | 192.168.0.122 | /home/uxdb/uxdbinstall/dbsql/bin/test |
| 备机 | 192.168.0.123 | /home/uxdb/uxdbinstall/dbsql/bin/test122 |

1. 主机配置：

   vi /home/uxdb/uxdbinstall/dbsql/bin/test/ux_hba.conf
   host replication uxdb 0.0.0.0/0 md5
   vi /home/uxdb/uxdbinstall/dbsql/bin/test/uxsinodb.conf
   listen_addresses = '*'
   max_wal_senders = 10
   wal_level = replica
   ux_ctl reload -D /home/uxdb/uxdbinstall/dbsql/bin/test

2. 在备机生成基础备份：

   ux_basebackup -h 192.168.0.122 -U uxdb -F p -P -R -D /home/uxdb/uxdbinstall/dbsql/bin/test122

3. 启动备机：

```
vi /home/uxdb/uxdbinstall/dbsql/bin/test122/uxsinodb.conf
 hot_standby=on
ux_ctl -D /home/uxdb/uxdbinstall/dbsql/bin/test122 start
```

4. 验证：

   在主机新建表并插入数据，在备机上查看，发现数据即时同步。

   hot standby是只读的，所以不能在备机上进行修改。

# 第 7 章 术语&缩略语

表 **7.1.** 缩略语

| 缩略语 | 英文全称 | 中文名称 |
| --- | --- | --- |
| API | Application Programming Interface | 应用程序编程接口 |
| ASCII | American Standard Code for Information Interchange | 美国标准信息交换码 |
| UXFS | Distributed File System | 分布式文件系统 |
| LSN | Log Sequence Number | 日志序列号 |
| PITR | Point-In-Time Recovery (Continuous Archiving) | 基于时间点恢复（连续存档） |
| UXDB | UXSINO Database | 优炫数据库 |
| WAL | Write-Ahead Log | 预写式日志 |