虚谷数据库 V12.7

PostgreSQL 数据迁移指南

文档版本 01

发布日期 2025-01-10

虚谷数据库
XUGU DB

声明

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

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

商标声明

虚谷数据库 XUGU DB

为成都虚谷伟业科技有限公司的注册商标。

本文档提及的其他商标或注册商标均非本公司所有。

注意事项

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

除非合同另有约定，本文档仅作为使用指导，所有内容均不构成任何声明或保证。

成都虚谷伟业科技有限公司

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

邮编：610023

网址：www.xugudb.com

前言

概述

本文档规定了虚谷数据库针对 PostgreSQL 应用迁移服务具备的服务能力，覆盖迁移评估、业务改造、迁移执行、运维优化。适用于 PostgreSQL 数据库各版本应用迁移服务的评价与指导。

读者对象

本文档主要适用于以下用户：

- 数据库开发人员
- 数据库维护人员
- 数据库管理员

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

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

修改记录

文档版本	发布日期	修改说明
01	2025-01-10	第一次正式发布

目录

1 迁移概述

PostgreSQL 是一种特性齐全的、开源的、以加州大学计算机系开发的 POSTGRES V4.2 为基础的对象关系型数据库管理系统（ORDBMS）。PostgreSQL 支持大部分的 SQL 标准并且提供了很多其他现代特性，如复杂查询、外键、触发器、视图、事务完整性、多版本并发控制等。同时，PostgreSQL 可以通过增加新的数据类型、函数、操作符、聚集函数、索引方法、过程语言等进行扩展。另外，因为许可证的灵活，任何人都可以以任何目的免费使用、修改和分发 PostgreSQL。

虚谷数据库是完全自有（非基于开源数据库源码改造）的国产数据库管理系统，在系统体系架构、部署应用方面存在差异，某些语法层需要进行转换，虚谷数据库针对 PostgreSQL 语法使用做了大量的兼容性研发，但并不能完全兼容 PostgreSQL。虚谷数据库提供了自研的数据迁移工具，能完成 PostgreSQL 数据库的表结构、数据、约束、索引等对象的自动化迁移，针对一些 PostgreSQL 非兼容项，需要 DBA 或应用开发进一步的人工干预。本迁移指导手册将针对 PostgreSQL 迁移至虚谷数据库做全面的讲解及流程梳理。

PostgreSQL 迁移到虚谷数据库主要工作及流程如下：

1. 待迁移系统评估：分析待迁移系统，确定需要迁移的对象，做好迁移内容的梳理及统计。
2. 确定迁移方案：迁移开展工作的人员、工具及计划安排。
3. 准备迁移环境：准备迁移环境，包括软硬件环境、配置参数等。
4. 开展正式迁移：按照迁移方案完成系统的整体迁移，并进行迁移工作确认。
5. 核对迁移结果：对照原有系统与迁移系统的结果一致性（对象、数量、数据、状态等）。
6. 系统验证及修订：核对迁移结果完成后，进行业务系统的对接及系统验证性测试，发现错误时及时反馈，并解决。
7. 迁移工作评估总结：调整配置、优化 SQL 等提升迁移系统能力的改进完善工作。

2 迁移系统评估

2.1 软硬件环境信息

- 数据库版本：源端采用的 PostgreSQL 数据库版本及 SQL 引擎类型。
- 中间件版本：业务系统中间件产品名称及版本，支持的数据库种类，是否支持通用数据库驱动 (如：JDBC) 配置。
- 业务系统版本：业务系统软件产品名称及版本。
- 操作系统：PostgreSQL 数据库、应用后台、中间件采用的操作系统类型及版本。

2.2 PostgreSQL 数据库统计信息

1. 查看数据库及数据库的字符集。

```
SQL> select datname,pg_encoding_to_char(encoding) from
  pg_database;
```

2. 统计数据库当前库中用户创建的对象及个数。

```
SQL> select 'database' as type,datname as db,count(*) from
  pg_catalog.pg_database group by datname
union  all
select 'schemas' as type,catalog_name as db,count(*) from
  information_schema.schemata where schema_name not in ('
  pg_catalog','pg_toast','information_schema') group by
  catalog_name
union  all
select 'tables' as type,table_catalog as db,count(*) from
  information_schema.tables where table_type='BASE TABLE'
  and table_schema not in ('pg_catalog','pg_toast','
  information_schema') group by table_catalog
union  all
select 'views' as type,table_catalog as db,count(*) from
  information_schema.views where table_schema not in ('
  pg_catalog','pg_toast','information_schema') group by
  table_catalog
union  all
select 'sequences' as type,sequence_catalog as db,count(*)
  from information_schema.sequences group by
  sequence_catalog
union  all
select 'procedures' as type,specific_catalog as db,count(*)
   from information_schema.routines where  routine_type='
```

```
      PROCEDURE' and specific_schema not in ('pg_catalog','
      pg_toast','information_schema') group by specific_catalog
   union  all
   select 'functions' as type,specific_catalog as db,count(*)
      from information_schema.routines where routine_type='
      FUNCTION' and specific_schema not in ('pg_catalog','
      pg_toast','information_schema') group by specific_catalog
   union  all
   select 'triggers' as type,trigger_catalog as db,count(*)
      from information_schema.triggers group by trigger_catalog
   union all
   select 'indexes' as type,s.catalog_name as db,count(*) from
       pg_catalog.pg_indexes pi left join information_schema.
      schemata s on pi.schemaname=s.schema_name where
      schemaname not in ('pg_catalog','pg_toast','
      information_schema') group by s.catalog_name
   union all
   select 'partition table' as type,current_database() as db,
      count(*) from (select distinct inhparent from pg_inherits
      ) tb
```

3. 统计表数据量（包括分区表中各分区的表数据量，父表数据量需要求和其对应分区表的数据量总和，可以使用 select count(*) from 分区表父表查询出分区表的总数据量）。

```
   SQL> select schemaname,relname, n_live_tup  from pg_catalog
  .pg_stat_user_tables psst order by relname;
```

4. 查看所有用户下的表对象使用空间大小。

```
   SQL> select schemaname,relname, pg_size_pretty(
      pg_relation_size(relid)) from pg_stat_user_tables  order
      by relname,pg_relation_size(relid) desc;
```

5. 查看数据库表对象使用的列名称（关键字排查）。

```
   SQL> select distinct column_name from information_schema.
      columns where table_schema not in ('pg_catalog','
      information','pg_toast') order by column_name;
```

6. 查看数据库表对象使用的数据类型，示例为 public 模式。

```
   SQL> select  data_type,count(*) from information_schema.
      columns where table_schema='public' group by data_type
      order by count(*) desc;
```

7. 列出所有的非系统数据所有表的主键信息和外键信息。

```
   SQL> select
   kcu.constraint_schema  as 约束拥有者,
   pc.conname as 约束名称,
   kcu.table_schema as 表拥有者,
   kcu.table_name as 表名,
   kcu.column_name as 列名,
   case when kcu.table_name=ccu.table_name then null else ccu.
      table_schema end as 外键表拥有者,
```

```
case when kcu.table_name=ccu.table_name then null else ccu.
   table_name end as 外键表名,
case when kcu.table_name=ccu.table_name then null else  ccu
   .column_name end as 外键表列名,
case pc.confupdtype
when 'a' then '无动作'
when 'r' then '限制'
when 'c' then '级联'
when 'n' then '设置为空'
when 'd' then '设置为缺省'
else null end as 约束更新规则,
case pc.confdeltype
when 'a' then '无动作'
when 'r' then '限制'
when 'c' then '级联'
when 'n' then '设置为空'
when 'd' then '设置为缺省'
else null end as 约束删除规则,
pc.contype as 约束类型,
pc.convalidated as 约束状态
from pg_catalog.pg_constraint pc
inner join information_schema.key_column_usage kcu on kcu.
   constraint_name=pc.conname
inner join information_schema.constraint_column_usage ccu
   on ccu.constraint_name=pc.conname
where kcu.constraint_schema not in ('pg_catalog','
   information_schema','pg_toast');
```

2.3 业务系统统计信息

2.3.1 业务框架信息

- 数据库驱动接口：类型及版本号（如：postgresql-42.5.0.jar）。
- 数据库方言版本：数据库方言及版本（如：org.hibernate.dialect.PostgreSQL95Dialect）。
- 事务特性及使用：是否使用事务 ACID 特性，应用事务隔离级别设置（read uncommitted/read committed/repeatable read/serializable）。
- 数据库操作框架：数据库操纵框架及版本（如：MyBatis VX.X.X/Hibernate VX.X.X/Spring JPA VX.X.X）。
- 数据库连接池：数据库连接池名称及版本（如：druid/dbcp/c3p0/hikaricp）。
- 大小写敏感及关键字：PostgreSQL 不区分大小写，默认使用小写，大写会转换为小写，需要大写时则使用双引号或者函数 upper/lower；业务中使用到的关键字。

2.3.2 业务运行信息

- 系统压力：业务系统最大、最小并发访问规模，低谷和峰值（如：最大 2000 用户/秒），出现时间及频率（如：早 9:00 11:00）。
- 数据访问热表：业务访问最频繁的数据表及范围。
- 磁盘 I/O 峰值：PostgreSQL 数据库最大磁盘 I/O 效率（单位：MB/秒）。
- 慢 SQL 统计：业务系统响应时长超过阈值的 SQL 命令。
- 数据库连接：连接活性最大保持时长。
- 事务变更（最大）：单个事务最大变更规模（行数）。
- 预处理参数：Preparestatement 对象最大参数个数。
- SQL 命令长度：业务系统 SQL 命令最大长度。
- 数据变更：单位时间（单位：秒）内，数据变更（INSERT、UPDATE、DELETE）规模（单位：条/秒），单日数据增长规模（单位：GB/天）。

3 迁移方案

迁移工作的开展需根据具体的工作要求制定相应的工作计划及迁移方案，迁移目的可能是基于测试验证的兼容性测试，也可能是替换 PostgreSQL 上线运行，迁移目的决定了迁移方案的具体规划。

迁移方案内容的选择如表3-1所示。

表 3-1 业务应用迁移方案内容选择

工作类型	兼容性测试	正式性迁移
硬件环境	基础硬件	业务硬件
软件环境	测试软件	业务软件
数据来源	部分数据或脱敏数据	正式数据
应用功能	部分功能	全部功能
应用性能	无或按需	核心业务性能
压力测试	无或按需	业务并发访问峰值
稳定性测试	无	7x24 小时
容灾性测试	无	容灾验证
数据同步	无	按需进行
业务割接	无	按需进行

兼容性测试：用户或者集成开发商对于迁移可能性和技术工作量的评估和确认工作，即尝试性迁移，迁移后可能并不会立刻进行产品级的应用功能、性能、稳定性测试。此类测试，以验证试验为主，对配置无特别要求，满足基本运行条件即可，在这种情况下，基础环境可以灵活选取，用虚拟机或物理机服务器均可。

正式性迁移：以正式上线为目的，需要对应用进行产品级全方位的功能点测试、性能测试、压力测试以及稳定性测试等集成测试，需考虑应用上线的各种可能性及保障措施。正式性迁移优

先采用物理服务器搭建数据库，并且对于物理服务器的相关硬件配置有硬性要求，根据系统数据量规模、性能要求、并发规模、可用性要求等基本情况向测试方提出建议。硬件支撑的合理性是保证迁移和测试效果的必要条件。

4 迁移准备

4.1 环境准备

硬件部署及检测

迁移环境所有的硬件全部准备就绪后，需要对硬件的状况进行一次整体性的监测，尤其是针对正式性迁移的硬件设备，需确保上线前的硬件设备能完全满足部署要求。

硬件监测：

1. CPU 超线程状态

   对于 CPU 并发运行要求较高的场景，应考虑关闭 CPU 超线程、减少 CPU 争用及锁等。

2. 服务器节能模式

   通常情况，服务器为了节约电能，默认情况处于节能模式，节能模式的服务器性能利用率一般在整机性能的 60% 到 80%，对有性能要求的业务应用或数据库服务器，应关闭服务器节能模式。

3. 网卡配置及 MTU 大小

   通常情况，服务器均配置有多网卡，用于扩大网络带宽路数或做网络容灾使用，应根据实际网络需求选取合适的网卡绑定策略。大部分网络设备的 MTU 都是 1500，应保持服务器本地 MTU 与网络 MTU 一致，否则在进行网络数据传输时，大的数据包就会被拆开来传送，这样会产生很多数据包碎片，增加丢包率，增加主机 CPU 计算量，消耗 CPU 资源，同时降低网络速度，本地 MTU 与网络 MTU 值建议设置为网络设备支持的 MTU 最大值（绝大多数 MTU 最大值为 9000）。

4. 操作系统配置参数

   检查操作系统文件句柄数、堆栈大小、网络内核参数，按照预估的访问压力情况进行参数配置。

5. 运行环境依赖包

   确保相关的依赖包已经完成安装。如：用于数据库监控的 SNMP 服务，用于分布式集群部署时钟一致性保障的 NTP 服务。

6. 防火墙规则设置

   确保用于数据库部署及用户访问相关的 TCP、UDP 端口、IP 地址在防火墙规则中已放

开。

软件部署及配置

根据迁移方案的设计选择合适的数据库产品系列（试用版/标准版/企业版/分布式版），版本选择时，优先选择产品系列中的最新数据库版本，避免已经修复的问题在迁移系统中出现，同时应充分考虑部署平台及操作系统类型，确保软硬件兼容性无问题。

数据库部署及配置参数调整：

1. 数据文件分组

   物理机为了充分利用磁盘 I/O 性能，可能制作有多 RAID 磁盘组，数据库数据文件可根据磁盘组的配置进行分组，最大化利用磁盘性能同时均衡磁盘数据库文件负载。数据库文件分组在“datafile.ini”配置文件中进行配置，配置时需指定数据库文件存放位置及文件数。

2. 数据库运行配置参数

   数据库运行参数应根据业务运行的实际情况及硬件资源进行调整与配置。数据库运行配置参数在“xugu.ini”配置文件中进行配置。

   - 数据缓存区内存“data_buff_mem”。数据缓存区大小用于定义数据库可缓存的数据大小，配置大小的选择有两种，一种是以物理硬件可分配给数据库使用的最大内存进行配置，一种是以统计获取的业务应用最大热数据规模进行配置。
   - 系统全局区内存“system_sga_mem”。系统全局区内存大小是数据库划分给 SQL 请求进行排序、统计等中间运算时的内存大小，可根据业务应用运算数据规模大小进行设置。
   - 最大 prepare 语句数“max_prepare_num”。单个数据库连接会话运行建立的最大“PrepareStatement”对象大小。
   - 单个事务最大允许变更行数“max_trans_modify”。单个数据库事务允许数据变更的最大行数，用于限制单个事务变更规模。
   - 最大闲置时间“max_idle_time”。数据库连接空闲，无执行动作的最大延时，超过最大闲置时间时，数据库自动清理空闲连接（中断退出）。
   - 系统最大连接数“max_conn_num”。数据库运行建立的最大数据库连接会话数。
   - 大小写敏感“cata_case_sensitive”。用于确定数据库对象及数据是否区分大小写，默认为“false”（不区分大小写）。

- 其他数据库运行参数“SYSDBA”。用户以系统管理员身份 **SYSDBA** 登录“SYSTEM”库，查看数据库配置参数表“SYS_VARS”，根据建议取值范围与业务应用实际需求，调整具体的数据库运行参数值。

> **⚠ 注意**
>
> - 部分数据库运行参数需重启数据库服务生效。
> - 严禁在生产环境中直接迁移。因为直接在生产库上进行数据迁移，会有很多风险存在，例如会影响生产库的效率、引发崩溃的可能等等，所以应该避免在 PostgreSQL 生产环境数据库中直接进行迁移，需要提前向应用开发商提出搭建一个测试环境的建议，准备 PostgreSQL 需要迁移的环境和数据。

3. 数据库安全信任配置

   对于高安全的应用场景，针对访问数据库的客户端及用户等进行安全信任规则检测，只允许符合安全规则的数据库访问连接进行数据库访问操作。数据库安全信任配置在“trust.ini”配置文件中进行配置。

创建数据库及用户

1. 创建数据库

   根据 PostgreSQL 源端数据库的字符集及时区，创建目标数据库端同名且字符集、时区一致的数据库实例。

2. 创建用户

   PostgreSQL 数据库中，用户在数据库实例的下层，PostgreSQL 共享用户给所有数据库，一个 PostgreSQL 用户可以登录所有数据库；虚谷数据库中，用户在数据库实例的下层，与 PostgreSQL 的用户策略类似（除了所有数据库共享用户），可采用以下方案：将 PostgreSQL 数据库实例对应虚谷数据库的数据库实例，每个虚谷数据库实例下创建同名称的用户以及同名称模式。

3. 用户赋权

   根据 PostgreSQL 数据库获取的用户权限信息，在目标数据库端赋予同等的用户权限。

4.2 工具准备

1. 管理工具准备

推荐使用 pgAdmin4 工具、Navicat 工具或 DBeaver 工具。管理工具用于连接到需要移植的 PostgreSQL 环境，以便进行移植数据的确认和初步的分析，同时便于获取 PostgreSQL 数据库对象的 DDL 脚本语句。也可以使用 PostgreSQL 其他的客户端工具，PostgreSQL 是一个开源数据库，可供选择的工具比较多，使用起来也比较方便。

2. 迁移工具准备

借助虚谷数据库的迁移工具，可以实现 PostgreSQL 数据库表结构及数据到虚谷数据库的快速迁移，迁移工具已内置了 PostgreSQL 到虚谷数据库的映射关系，极大节约了人工成本。也可以使用第三方的迁移工具或迁移同步工具，先手动完成数据库对象的创建，再将 PostgreSQL 数据库中的表数据迁移同步至虚谷数据库中。

5 迁移操作

5.1 兼容适配

PostgreSQL 包含了一些可能在其他 SQL 数据库中不支持的扩充。使用 PostgreSQL 中非常不规范的 SQL 语法，导致数据库迁移到其他数据库中时，发现原来写的某些视图或者 SQL 查询放到目标数据库中无法直接运行。

针对 PostgreSQL 数据库与虚谷数据库的兼容情况，虚谷梳理了《PostgreSQL-虚谷数据库兼容适配手册》（可联系虚谷技术支持人员获取），用于 PostgreSQL 迁移至虚谷数据库时的操作参考。

5.2 数据迁移

5.2.1 迁移步骤

借助虚谷数据库数据迁移工具可以实现 PostgreSQL 数据库表对象及数据迁移至虚谷数据库，迁移内容包括表、表约束、视图、序列、索引、数据，快速构建虚谷数据库基础环境。

使用虚谷数据库数据迁移工具迁移 PostgreSQL 至虚谷数据库的具体执行步骤：

1. 选择迁移方式（PostgreSQL 迁移到 Xugu）
2. 填写迁移数据源（源端 PostgreSQL 与目标端 Xugu）
3. 选择迁移的数据库
4. 选择迁移的数据库对象
5. 迁移任务生成及确认
6. 执行迁移任务
7. 获取迁移执行报告

> ⚠ 注意
>
> - 步骤 4 中的迁移对象顺序：序列值 -> 表 -> 视图。
> - 迁移表时，对于大表应当单独迁移。
> - Xugu 不支持物化视图（实体化视图）以及物化视图日志（实体化视图日志）。

详细操作步骤，请参见《迁移工具用户指南》。

5.2.2 兼容转换

使用虚谷数据库数据迁移工具的过程中，数据迁移工具针对一些需要兼容 PostgreSQL 的数据类型、函数或使用方式拟定了自动化的替换规则，用户可以在工具上根据实际情况调整单项替换规则，进行手动补充，如图 5-1 所示。

图 5-1 映射条件以及迁移选项设置

5.2.3 默认兼容规则

5.2.3.1 默认数据类型映射

PostgreSQL 与虚谷数据库的默认数据类型映射如表 5-1 所示。

表 5-1 PostgreSQL 与 XuGu 默认数据类型映射

PostgreSQL 数据类型	PostgreSQL 列类型	XuGu 数据类型	XuGu 列类型
SMALLINT	SMALLINT	SMALLINT	SMALLINT
INT INTEGER	INTEGER	INTEGER	INTEGER
		接下页	

PostgreSQL 数据类型	PostgreSQL 列类型	XuGu 数据类型	XuGu 列类型
BIGINT	BIGINT	BIGINT	BIGINT
DECIMAL[(M[,D])] NUMERIC[(M[,D])]	NUMERIC	NUMERIC	NUMERIC
REAL	REAL	FLOAT	FLOAT
DOUBLE PRESI- CION	DOUBLE PRESI- CION	DOUBLE	DOUBLE
SMALLSERIAL SERIAL BIGSERIAL	SAMLLINTRGER INTRGER BIGINTRGER	INTEGER iden- tity(1,1) INTEGER iden- tity(1,1) BIGINTEGER iden- tity(1,1)	INTEGER
CHARACTER(M) CHAR(M)	CHARACTER	CHARACTER(M) CHAR(M)	CHAR
CHARACTER VARY- ING(M) VARCHAR(M)	CHARACTER VARY- ING	CHARACTER VARY- ING(M) VARCHAR(M)	VARCHAR
TEXT	TEXT	CLOB	CLOB
BYTEA	BYTEA	BINARY	BINARY

接下页

PostgreSQL 数据类型	PostgreSQL 列类型	XuGu 数据类型	XuGu 列类型
TIMESTAMP[(M)] [WITH-OUT TIME ZONE]	TIMESTAMP WITH-OUT TIME ZONE	TIMESTAMP	TIMESTAMP
TIMESTAMP[(M)] WITH TIME ZONE	TIMESTAMP WITH TIME ZONE	TIMESTAMP[(M)] WITH TIME ZONE	DATETIME WITH TIME ZONE
DATE	DATE	DATE	DATE
TIME[(M)] [WITH-OUT TIME ZONE]	TIME WITH-OUT TIME ZONE	TIME	TIME
TIME[(M)] WITH TIME ZONE	TIME WITH TIME ZONE	TIME[(M)] WITH TIME ZONE	TIME WITH TIME ZONE
INTERVAL	INTERVAL	INTERVAL	INTERVAL
BOOLEAN	BOOLEAN	BOOLEAN	BOOLEAN
ENUM	ENUM	X	X
BIT[(M)]	BIT	X	X

接下页

PostgreSQL 数据类型	PostgreSQL 列类型	XuGu 数据类型	XuGu 列类型
BIT VARY-ING[(M)]	BIT VARY-ING	X	X
TSVECTOR	TSVECTOR	X	X
TSQUERY	TSQUERY	X	X
UUID	UUID	GUID	GUID
XML	XML	X	X
JSON	JSON	X	X

1. PostgreSQL 数据类型的特殊值：
    1. numeric 类型数据可以插入 NaN 字符串，带单引号插入，含义为“不是一个数”。
    2. real、double precision 类型数据可以插入 NaN、infinity、-infinity 字符串，带单引号插入，含义分别为“不是一个数”、“正无穷大”、“负无穷大”。
    3. 日期/时间类型插入特殊字符串。

输入字符串	适用类型	描述
epoch	date, timestamp	1970-01-01 00:00:00+00 (Unix 系统零时)
infinity	date, timestamp	比任何其它时间戳都晚
-infinity	date, timestamp	比任何其它时间戳都早
now	date, time, timestamp	当前事务的开始时间
today	date, timestamp	今日午夜
tomorrow	date, timestamp	明日午夜

接下页

输入字符串	适用类型	描述
yesterday	date, timestamp	昨日午夜
allballs	time	00:00:00.00 UTC

2. PostgreSQL 的序列号类型（smallserial、serial、bigintserial）不是真正的类型，是一个逻辑类型，实质上就是 INTEGER 类型加了默认自增值，INTEGER NOT NULL DEFAULT nextval（’tablename_colname_seq’），其中 small 和 big 分别对应 INTEGER 的 small 与 big。

5.2.3.2 TIMESTAMP 类型默认值

在 PostgreSQL 中时间类型“TIMESTAMP”默认 default 设置为字符串 epoch，其值为“1970-01-01 00:00:00”。

数据迁移工具将 PostgreSQL 时间值“0000-00-00”映射替换为“1970-01-01”。

数据迁移工具将 PostgreSQL 时间值“0000-00-00 00:00:00”映射替换为“1970-01-01 00:00:00”。

5.3 对象迁移

5.3.1 特定对象改写

对系统中的表和视图，可以使用虚谷数据库数据迁移工具进行迁移。如果还用到了自定义类型、存储过程、函数、触发器等，需要导出源端数据库上的定义，手动进行等价改写。

首先，可以先从 PostgreSQL 导出用户对象 SQL 脚本，PostgreSQL 导出存储过程及函数的执行命令：

```
[root@xugu]# pg_dump -s -h ipaddress -U username databasename> dum.
   sql
```

然后，将导出的 SQL 脚本，经人工修改之后，再导入到虚谷数据库中运行。

5.3.2 常见函数处理

5.3.2.1 extract() 函数

以 extract 函数为例，PostgreSQL 中为获取日期/时间的子集，也可以转换为 unixtimestamp（单位：秒），虚谷数据库有对应的函数获取日期/时间的各个子集和 unixtimestamp（单位：

毫秒），改写后在虚谷数据库中可以达到相同的效果，示例如下：

```
--PostgreSQL, 获取对应的子集
SQL> select extract('year' from current_timestamp);
SQL> select extract('month' from current_timestamp);
SQL> select extract('day' from current_timestamp);
SQL> select extract('hour' from current_timestamp);
SQL> select extract('minute' from current_timestamp);
SQL> select extract('second' from current_timestamp);

--XuGu, 获取对应的子集
SQL> select getyear(sysdate);
SQL> select getmonth(sysdate);
SQL> select getday(sysdate);
SQL> select gethour(sysdate);
SQL> select getminute(sysdate);
SQL> select getsecond(sysdate);
SQL> select extract_year(sysdate);
SQL> select extract_month(sysdate);
SQL> select extract_day(sysdate);
SQL> select extract_hour(sysdate);
SQL> select extract_minute(sysdate);
SQL> select extract_second(sysdate);

--PostgreSQL, 获得unixtimestamp
SQL> select extract('epoch' from current_timestamp);

--XuGu, 获得unixtimestamp
SQL> select unix_timestamp(sysdate);
```

5.3.2.2 系统时间函数

PostgreSQL 数据库中获取系统时间的函数如下，其中“precision”为秒的标度，“timeofday”为 text 格式的应用显示时间。

- CURRENT_DATE
- CURRENT_TIME
- CURRENT_TIMESTAMP
- CURRENT_TIME(precision)
- CURRENT_TIMESTAMP(precision)
- LOCALTIME
- LOCALTIMESTAMP
- LOCALTIME(precision)

- LOCALTIMESTAMP(precision)
- transaction_timestamp()
- statement_timestamp()
- clock_timestamp()
- timeofday()
- now()

虚谷数据库中对应的系统时间函数为“current_timestamp”、“sysdate”。

```
-- PostgreSQL系统时间函数（部分示例）
SQL> select current_timestamp as systemtime;
SQL> select localtime as systemtime;
SQL> select localtimestamp as systemtime;
systemtime          |
--------------------+
2022-07-28 03:14:53|

-- XuGu系统时间函数
SQL> select current_timestamp as systemtime from dual;
SQL> select sysdate() as systemtime from dual;
systemtime          |
--------------------+
2022-07-28 03:14:53|
```

5.3.2.3 其他函数

对于 PostgreSQL 独有的函数或基于 PostgreSQL 语法创建的自定义函数，在虚谷数据库中需要基于虚谷数据库的 SQL 语法进行改写，实现相同的功能。

5.3.2.4 自动增长

在 PostgreSQL 数据库中，设置自动增长的数据类型为“smallserial”、“serial”、“bigserial”，在插入数据时，自增字段自动生成自增值。与 PostgreSQL 数据库对应，在虚谷数据库中，使用“IDENTITY”设置自动增长。

自动增长使用规则区别如表 5-3 所示。

表 5-3 自动增长使用规则区别

属性	PostgreSQL	XuGu
关键字	smallserial、serial、bigse-rial	IDENTITY
数据类型	整型	整型
要求	无	无
使用场景	自动增长字段插入 DE-FAULT 或在插入时省略该字段	自动增长字段插入 DE-FAULT 或在插入时省略该字段
起始值	固定为 1	可设置范围【1, 数据类型最大值】
增长步长	固定为 1	可设置范围【1, 数据类型最大值】
插入值 ≤ 自动增长值	自动增长无影响	自动增长无影响
删除数据	自动增长无影响	自动增长无影响
删除自增	X	√
添加自增	X	√
自增循环	X	X

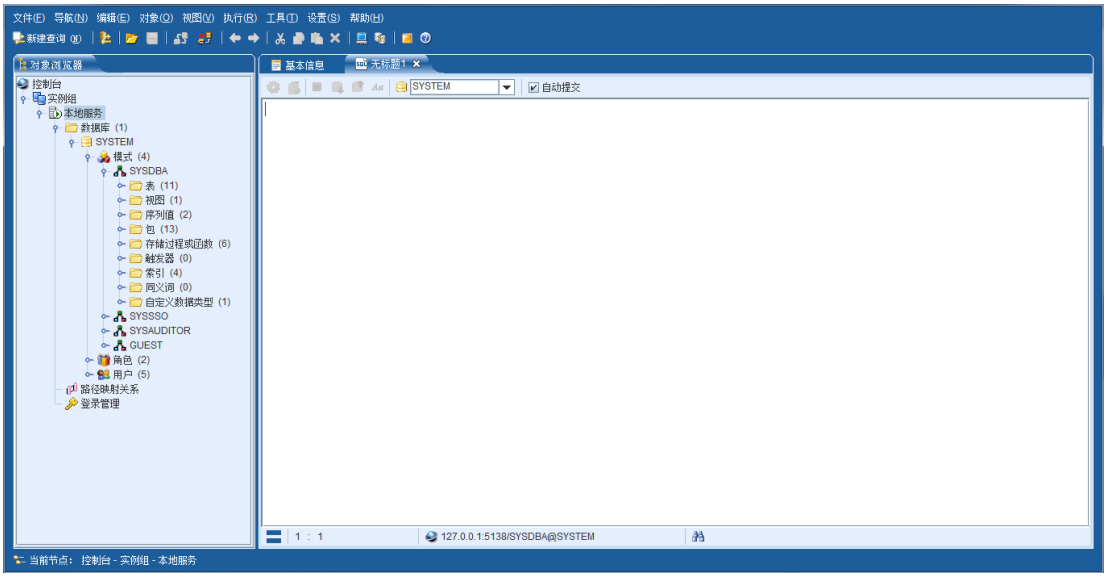
```
-- PostgreSQL自动增长（起始值：字段最大值，默认增长步长为1）
SQL> Create Table my_auto(
a smallserial,
b serial,
c bigserial,
d integer
);

-- PostgreSQL自增信息
Column |   Type   | Collation | Nullable |            Default
--
   ------+----------+-----------+----------+-------------------------------
```

```
a       | integer   |           | not null | nextval('my_auto_a_seq'
   ::regclass)
b       | smallint  |           | not null | nextval('my_auto_b_seq'
   ::regclass)
c       | bigint    |           | not null | nextval('my_auto_c_seq'
   ::regclass)
d       | integer   |           |          |

-- XuGu自动增长（需指定自增起始值及增长步长）
SQL> Create Table my_auto(
id Int identity(1,1) Primary Key,
username Varchar(20)
);

-- XuGu自增信息
Field    |Type         |Null |Key   |Default|Extra   |
---------+-------------+-----+------+-------+--------+
ID       |INTEGER      |NO   |PK_S498816608141247 |        |
   $SYS_SEQ_1048625|
USERNAME|CHAR(20)     |YES
    |                       |        |                 |
```

5.3.3 对象管理

虚谷数据库管理工具是虚谷数据库系统的图形化工具，可以帮助数据库管理员更直观、更方便的管理和维护虚谷数据库。可以借助虚谷数据库管理工具对数据库对象进行管理，如图 5-2 所示。

图 5-2 虚谷数据库管理工具



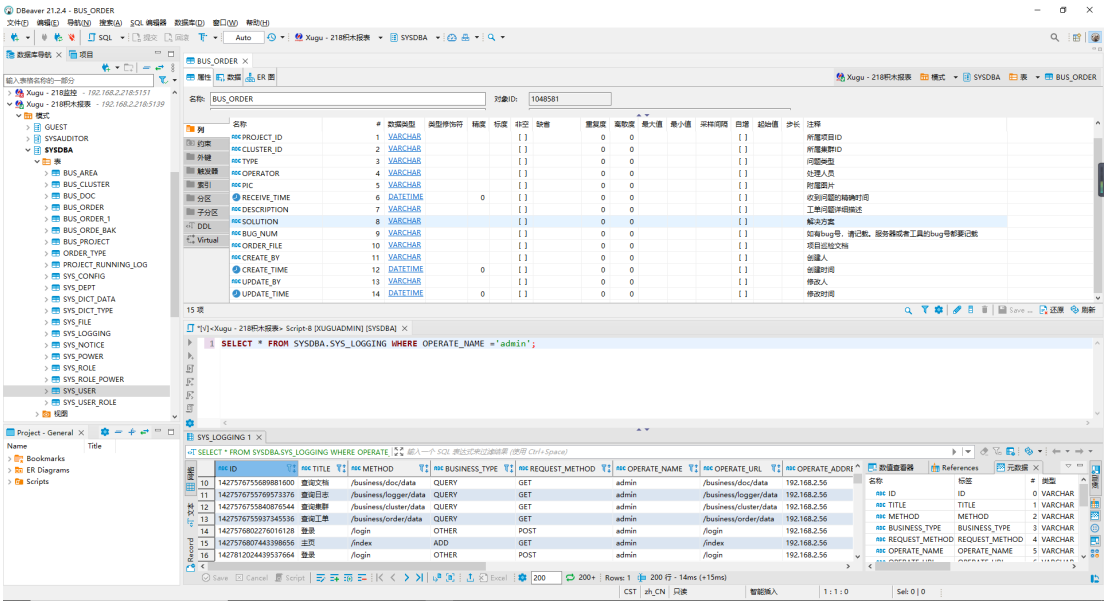
也可以通过 sql 语句进行修改，使用方式如下：

```
-- 创建表
SQL> create table t1(c1 int, c2 int, c3 int);

-- 增加列
```

```
SQL> alter table t1 add column c4 numeric(4,2);

-- 给列增加缺省值
SQL> alter table t1 alter column c4 set default 12.34;

-- 给列增加约束
SQL> alter table t1 alter column c4 set not null;
SQL> alter table t1 add constraint chk_key check(c1 > 10);
```

除此之外，通过虚谷数据库通用管理工具 DBeaver 也可以实现类似的操作，如图 5-3 所示。

图 5-3 通用管理工具 DBeaver



5.4 应用迁移

更换数据库系统软件，需要应用系统代码或配置文件做一定程度的修改，修改的主要内容包含：数据库连接串、URL 信息、数据库方言、不兼容的 SQL 语句、冲突的关键字等。

应用迁移主要工作：

1. 在应用软件工程引用数据库驱动包的位置放入虚谷数据库驱动接口（如：JDBC 驱动接口 JAR 文件）。
2. 应用软件涉及到 Hibernate 时，需指定虚谷数据库 Hibernate 方言驱动包。
3. 在配置数据库连接串相关的源代码或配置文件时，修改虚谷数据库连接配置信息。
4. PostgreSQL 与虚谷数据库有不同的系统关键字定义，出现关键字冲突时，进行调整或加入双引号。
5. 对于一些在应用软件中的虚谷数据库不兼容 PostgreSQL 的 SQL 语句，需进行对等替换。

虚谷数据库 URL 连接串修改方式：

```
//  定义XuGu JDBC驱动串
String jdbcString = "com.xugu.cloudjdbc.Driver";
//  定义XuGu URL连接串
String urlString = "jdbc:xugu://localhost:5138/database";
```

虚谷数据库方言配置方式：

```
<properties>
<property name="hibernate.dialect" value="com.xugu.dialect.
   XuguDialect5" />
<property name="hibernate.connection.driver_class" value="com.xugu.
   cloudjdbc.Driver" />
<property name="hibernate.connection.url" value="jdbc:xugu://
   localhost:5138/database" />
<property name="hibernate.connection.username" value="username" />
<property name="hibernate.connection.password" value="password" />
</properties>
```

虚谷数据库支持的数据库驱动语言种类如表 5-4 所示。

表 5-4 虚谷支持的数据库驱动语言种类

语言	程序包文件	说明
JDBC	xugu-jdbc-*.*.*..jar	Java 类应用数据库驱动程序
Hibernate	xugu-dialect3-*.*.*.jar	Java 类应用数据库方言包（对应 Hibernate3）
Hibernate	xugu-dialect4-*.*.*..jar	Java 类应用数据库方言包（对应 Hibernate4）
Hibernate	xugu-dialect5-*.*.*..jar	Java 类应用数据库方言包（对应 Hibernate5）
ODBC	xugu-odbc-*.*.*	C/C++ 类应用数据库驱动程序
XGCI	xugu-ci-*.*.*	虚谷 CI 类应用数据库驱动程序
接下页		

语言	程序包文件	说明
OCI	xugu-oci-*.*.*	OCI 类应用数据库驱动程序
C-Sharp	xugu-csharp-*.*.*	C# 类应用数据库驱动程序
GO	xugu-go-*.*.*	GO 语言类应用数据库驱动程序
PHP	xugu-php-*.*.*	PHP 类应用数据库驱动程序
Python	xugu-python-*.*.*	Python 类应用数据库驱动程序

各类数据库驱动接口的开发使用，请参照对应的接口开发手册。

6 迁移结果统计

1. 查看数据库及数据库的字符集。

```
SQL> SELECT DB_NAME,CHAR_SET,TIME_ZONE,ONLINE FROM
   DBA_DATABASES WHERE DB_NAME='{数 据 库 名 称}';
```

2. 统计数据库对象及个数。

```
SQL>DECLARE
    CURSOR CUR IS
    SELECT SCH.SCHEMA_NAME OWNER,'TABLE' AS OBJ_TYPE,COUNT
       (*) CNT FROM DBA_TABLES TAB LEFT JOIN DBA_SCHEMAS SCH
        ON TAB.SCHEMA_ID = SCH.SCHEMA_ID GROUP BY SCH.
       SCHEMA_NAME
    UNION ALL
    SELECT SCH.SCHEMA_NAME OWNER,'VIEW' AS OBJ_TYPE,COUNT
       (*) CNT FROM DBA_VIEWS VW LEFT JOIN DBA_SCHEMAS SCH
       ON VW.SCHEMA_ID = SCH.SCHEMA_ID GROUP BY SCH.
       SCHEMA_NAME
    UNION ALL
    SELECT SCH.SCHEMA_NAME OWNER,'SEQUENCES' AS OBJ_TYPE,
       COUNT(*) CNT FROM DBA_SEQUENCES SEQ LEFT JOIN
       DBA_SCHEMAS SCH ON SEQ.SCHEMA_ID = SCH.SCHEMA_ID
       GROUP BY SCH.SCHEMA_NAME
    UNION ALL
    SELECT SCH.SCHEMA_NAME OWNER,'PACKAGE HEAD' AS OBJ_TYPE
       ,COUNT(*) CNT FROM DBA_PACKAGES PAK LEFT JOIN
       DBA_SCHEMAS SCH ON PAK.SCHEMA_ID = SCH.SCHEMA_ID
       WHERE SPEC IS NOT NULL GROUP BY SCH.SCHEMA_NAME
    UNION ALL
    SELECT SCH.SCHEMA_NAME OWNER,'PACKAGE BODY' AS OBJ_TYPE
       ,COUNT(*) CNT FROM DBA_PACKAGES PAK LEFT JOIN
       DBA_SCHEMAS SCH ON PAK.SCHEMA_ID = SCH.SCHEMA_ID
       WHERE BODY IS NOT NULL GROUP BY SCH.SCHEMA_NAME
    UNION ALL
    SELECT SCH.SCHEMA_NAME OWNER,'PROCEDURE' AS OBJ_TYPE,
       COUNT(*) CNT FROM DBA_PROCEDURES PROC LEFT JOIN
       DBA_SCHEMAS SCH ON PROC.SCHEMA_ID = SCH.SCHEMA_ID
       WHERE PROC.RET_TYPE IS NULL GROUP BY SCH.SCHEMA_NAME
    UNION ALL
    SELECT SCH.SCHEMA_NAME OWNER,'FUNCTION' AS OBJ_TYPE,
       COUNT(*) CNT FROM DBA_PROCEDURES FUNC LEFT JOIN
       DBA_SCHEMAS SCH ON FUNC.SCHEMA_ID = SCH.SCHEMA_ID
       WHERE FUNC.RET_TYPE IS NOT NULL GROUP BY SCH.
       SCHEMA_NAME
    UNION ALL
    SELECT SCH.SCHEMA_NAME OWNER,'TRIGGER' AS OBJ_TYPE,
       COUNT(*) CNT FROM DBA_TRIGGERS TRIG LEFT JOIN
       DBA_SCHEMAS SCH ON TRIG.SCHEMA_ID = SCH.SCHEMA_ID
       GROUP BY SCH.SCHEMA_NAME
    UNION ALL
```

```
        SELECT SCH.SCHEMA_NAME OWNER,'SYNONYM' AS OBJ_TYPE,
           COUNT(*) CNT FROM DBA_SYNONYMS SYN LEFT JOIN
           DBA_SCHEMAS SCH ON SYN.SCHEMA_ID = SCH.SCHEMA_ID
           GROUP BY SCH.SCHEMA_NAME
        UNION ALL
        SELECT SCH.SCHEMA_NAME OWNER,'UDT TYPE' AS OBJ_TYPE,
           COUNT(*) CNT FROM DBA_TYPES TYP LEFT JOIN DBA_SCHEMAS
            SCH ON TYP.SCHEMA_ID = SCH.SCHEMA_ID GROUP BY SCH.
           SCHEMA_NAME
        UNION ALL
        SELECT DB.DB_NAME OWNER,'JOB' AS OBJ_TYPE,COUNT(*) CNT
           FROM DBA_JOBS JOB LEFT JOIN DBA_DATABASES DB ON JOB.
           DB_ID = DB.DB_ID GROUP BY DB.DB_NAME;
        VAR_COUNT NUMBER(10);
        VAR_STR VARCHAR(500);
        BEGIN
        FOR VAR_ROW IN CUR LOOP
        SEND_MSG( VAR_ROW.OWNER||'.'||VAR_ROW.OBJ_TYPE || '='
            || VAR_ROW.CNT);
        END LOOP;
        END;
        /
```

3. 统计表数据量。

```
    SQL> DECLARE
        CURSOR CUR IS SELECT S.SCHEMA_NAME || '.' || T.
           TABLE_NAME AS TABLE_NAME FROM DBA_TABLES T
        LEFT JOIN DBA_SCHEMAS S ON S.SCHEMA_ID = T.SCHEMA_ID
           ORDER BY  S.SCHEMA_NAME,T.TABLE_NAME;
        VAR_COUNT NUMBER(10);
        VAR_STR VARCHAR(500);
        BEGIN
        FOR VAR_ROW IN CUR LOOP
        VAR_STR := 'SELECT COUNT(*) FROM ' || VAR_ROW.
           TABLE_NAME;

        EXECUTE IMMEDIATE VAR_STR INTO VAR_COUNT;
        SEND_MSG( VAR_ROW.TABLE_NAME || '=' || TO_CHAR(
           VAR_COUNT));
        END LOOP;
        END;
        /
```

4. 统计约束及定义（约束类型,'F': 外键;'R': 引用外键;'C': 值检查;'D': 默认值;'U': 唯一值;'P': 主键;）。

```
        SELECT
        DB.DB_NAME AS 数据库名,
        CH.SCHEMA_NAME AS 模式名,
        T.TABLE_NAME AS 表名,
        C.CONS_NAME AS 约束名,
        C.CONS_TYPE AS 约束类型,
        REPLACE(C.DEFINE,  '"',  '') AS 约束定义
        FROM
```

```
DBA_CONSTRAINTS C
LEFT JOIN DBA_TABLES T ON
C.TABLE_ID = T.TABLE_ID
LEFT JOIN DBA_DATABASES DB ON
C.DB_ID = DB.DB_ID
LEFT JOIN DBA_SCHEMAS CH ON
CH.USER_ID = T.USER_ID AND CH.DB_ID=DB.DB_ID
WHERE DB.DB_NAME='DBNAME' AND CH.SCHEMA_NAME='USER'
```

5. 统计索引及定义（约束类型,'BTREE': B 树索引;'RTREE': R 树索引;'FULL': 全文索引;'BITMAP': 位图索引;'UNION': 联合索引;）。

```
SQL> SELECT
DB.DB_NAME AS 数据库名,
CH.SCHEMA_NAME AS 模式名,
T.TABLE_NAME AS 表名,
INDEX_NAME AS 索引名,
(CASE INDEX_TYPE
WHEN 0 THEN 'BTREE'
WHEN 1 THEN 'RTREE'
WHEN 2 THEN 'FULLTEXT'
WHEN 3 THEN 'BITMAP'
WHEN 4 THEN 'UNION'
END) AS 索引类型,
IS_PRIMARY AS 是否是主键索引,
IS_UNIQUE AS 是否是唯一索引,
FIELD_NUM AS 索引字段数,
REPLACE(KEYS, '"', '') AS 索引字段
FROM
DBA_INDEXES AS C
LEFT JOIN DBA_TABLES T ON
C.TABLE_ID = T.TABLE_ID
LEFT JOIN DBA_DATABASES DB ON
C.DB_ID = DB.DB_ID
LEFT JOIN DBA_SCHEMAS CH ON
CH.USER_ID = T.USER_ID AND CH.DB_ID = DB.DB_ID
WHERE
DB.DB_NAME = 'DBNAME' AND CH.SCHEMA_NAME = 'USER'
```

7 迁移验证

7.1 一致性校验

PostgreSQL 至虚谷数据库迁移一致性校验如表 7-1 所示。

表 7-1 一致性校验

验证项	描述	校验标准	必选/可选	是否达标	迁移说明
基础环境	软硬件配置	● 硬件服务器字符集一致 ● 硬件服务器时钟/时区一致 ● 硬件服务器数据库用户及权限一致	必选		● 服务器字符集不一致时，可能导致中文字符出现乱码 ● 服务器时区不一致可能导致获取操作系统时间不一致
数据库初始化配置	数据库关键初始化配置	● 字符集兼容一致 ● 名称及大小写兼容 ● 用户权限兼容一致 ● 数据库时区一致	必选		● PostgreSQL 与虚谷的字符集只能设置到库级 ● 客户端时区也应保持一致
数据库对象	数据库对象及状态	● 数据库对象类型及个数一致 ● 数据库对象状态一致 ● 数据库对象执行结果一致	必选		数据库对象统计及执行效果一致
接下页					

验证项	描述	校验标准	必选/可选	是否达标	迁移说明
数据校验	数据校验及比对	● 数据总量一致 ● 数据结果一致 ● 数据无乱码	必选		数据完整迁移校验

7.2 系统验证

7.2.1 系统测试

系统测试环节是迁移关键环节的重中之重，需要投入大量的时间和资源，稍有不慎，可能会导致后续的迁移失败、数据丢失甚至是业务中断、混乱的灾难性后果。全面系统测试通常包含功能测试、性能测试、稳定性测试、可靠性测试、扩展能力测试、安全能力测试、回退方案验证等。

系统测试环节的典型测试类型及测试项如表 7-2、表 7-3、表 7-4、表 7-5 所示。

表 7-2 典型性能测试类型及测试项

序号	测试类型	测试项
1	吞吐量	TPS、QPS
2	响应时间	平均响应时间，最小、最大响应时间，时间百分比
3	并发量	同时处理业务请求数量
4	物理资源使用率	CPU、网络、内存资源、磁盘 I/O 及数据库资源利用率

表 7-3 典型可靠性测试类型及测试项

序号	测试类型	测试项
1	硬件故障	电源插拔、网线插拔、硬盘、交换机、机架以及机房故障下的数据库服务能力
2	操作系统故障	CPU 资源占用、I/O 资源占用、内存资源占用、磁盘空间占用下数据库服务能力
3	数据库服务故障	数据库系统文件被损坏情况下的数据库服务能力
4	应用故障	上层应用与数据交互过程中发生故障下的数据库服务能力

表 7-4 典型扩展能力测试类型及测试项

序号	测试类型	测试项
1	计算均衡能力	具备将接入数据库的计算压力进行平衡的能力
2	数据按需均衡能力	具备将数据库内的数据进行平衡的能力
3	集群在线扩容能力	具备集群扩展能力，且性能能够随之提升
4	集群在线缩容能力	具备集群缩减能力

表 7-5 典型安全能力测试类型及测试项

序号	测试类型	测试项
1	权限验证	能够对数据库内数据操作进行权限验证
2	身份验证	能够对接入数据库用户进行身份认证
3	操作审计	能够对数据库内操作进行审计
4	流量控制能力	能够配置数据库的流量上限，例如提供连接数限制等

慢 SQL 记载方式：

1. 部署虚谷数据库集群监控软件，采集慢 SQL 指标；
2. 使用 Druid 数据库连接池 SQL 命令响应时间采集功能，记载慢 SQL 命令及响应时间。

7.2.2 系统优化

更新统计信息

数据核对无误后，应进行一次全库的统计信息更新工作。统计信息更新脚本示例如下：

```
-- 分析指定表、字段的基础信息
-- 分析TAB_TEST表的FIELD01字段的最大值、最小值、重复度、离散度，采样间隔：每一条采集
SQL> EXEC DBMS_STAT.ANALYZE_TABLE('SYSDBA.DATE_DIM','D_YEAR',1,NULL);

-- 获取表对象统计信息
SQL> SELECT DBMS_STAT.GET_STAT_INFO('SYSDBA.DATE_DIM');
```

更新统计信息的目的在于大批量迁移数据后，可能会导致数据库优化器根据错误的统计信息得到错误的查询计划，严重影响查询性能。

数据备份

更新统计信息后，在数据量不大且磁盘空间足够的情况下应进行一次数据备份工作。数据备份有两种方式：一是正常停止数据库后，拷贝备份 XHOME 文件夹；二是使用数据库提供的备份功能。

> **说明**
>
> 逻辑备份恢复操作请参考《备份恢复指南》。

整理对象脚本

整理所有数据库对象脚本，对项目迁移情况进行记录和备份，方便再次进行数据迁移。备份的数据库对象脚本包括：序列定义及当前值，表定义，索引定义，视图定义，函数定义，存储过程定义，包及包体定义、自定义类型和同义词定义。

7.2.3 业务割接

业务生产割接是指服务提供方将原生产应用环境中的数据库停止，并切换到新环境目标数据库的过程，其中包括应用切换、数据同步切换以及其他相关配置过程切换。

- 割接方案（业务厂家、集成厂商、数据库 DBA）

  割接方案中应包含系统备份方案、应急预案、回退方案，明确割接的操作步骤、操作时间和操作人员，对新系统实施压力测试和破坏性测试，模拟在最极端环境下新系统功能的完整性、稳定性和高可靠性。正式割接前的备份工作必不可少，在新环境上线前务必做好旧程序包的保留和数据同步，以便在紧急情况可以快速回退。

- 模拟割接（业务厂家、集成厂商、数据库 DBA）

  切换演练需制定切换检查清单，演练期间严密监控容灾数据库的系统负载、异常等待事件等内容。割接前通常需要至少 3 次割接演练，以确保割接过程中各个环节没有疏漏。根据不同业务系统情况制定不同的割接流程，为每个流程设定责任人，通关制完成各个环节。正式割接环节分为生产环境准备和按照割接方案正式执行割接两部分。

- 回退方案（业务厂家、集成厂商、数据库 DBA）

  迁移完成后，最重要的环节是切换后生产环境的第一个业务高峰，需要配置专业的数据库专家，快速响应应用和数据库出现的突发问题。之后，需要定期跟踪一段时间，以保障业务系统的稳定运行。如果遇到突发情况且无法处理时，应依据回退方案和演练细则逐步完成回退。

多路数据库服务，数据同步方案：

- 方案一：数据库并行部署

  在所有采集服务器开启两个入库进程，即一份原始结果同时入两套数据库。在没有额外测试系统的条件下，利用现有资源，最大限度模拟仿真正式生产环境的并发压力，同时完成

负载均衡、单点故障模拟。

- 方案二：数据库数据复制

  采用低侵入性的数据迁移同步工具，配置同步规则，将全量数据、增量数据同步至目标端数据库中。即数据库服务有多路 (同构或异构)，业务系统接入其中一路数据库服务，其他数据库服务与业务数据库服务采用数据同步方式实现数据一致性保障，在业务数据库服务出现故障或需要业务割接时，调整连接信息，将业务数据库服务切换至另一路数据库服务。

8 迁移总结

- 功能完善需求

  以业务系统准确上线为前提，进行数据库、应用系统的迁移。在迁移过程中出现不兼容或调用错误时，应分析具体原因，优先给出替代或解决方案，保证业务正常运行，后续再根据未兼容项进行分析，对于非常有必要的功能提交产品研发需求，进行产品兼容开发。

- 迁移效果评估

  针对原有业务系统与迁移后业务系统的功能、性能等指标进行对比，评估是否达到预期的迁移效果。

- 运维服务保障

  迁移完成后，为保障业务系统在新数据库系统的良好运行，应针对新数据库系统、接口对接保障、运维保障等开展系统培训讲解，确保运维保障团队从架构层、流程层、操作层掌握整套系统运维手段，全面接管新业务系统运维保障服务。

9 常见问题

9.1 数据库连接失败

表 9-1 数据库连接失败

可能原因	问题分析	问题处理
驱动加载失败	未识别到数据库驱动程序	确认驱动程序包存放位置是否正确并随应用软件启动正常加载
配置信息错误	数据库连接串或登录信息输入错误	确认连接信息，也可使用客户端工具进行验证
网络通道闭塞	应用软件与数据库服务器之间的 IP 地址或端口访问不可达	检查防火墙设置，确认 IP 地址及 TCP 端口已开放
登录帐户锁定	登录帐户因输入登录信息错误超过三次，拒绝用户登录	等待帐户解锁（3 分钟）

9.2 SQL 命令失效报错

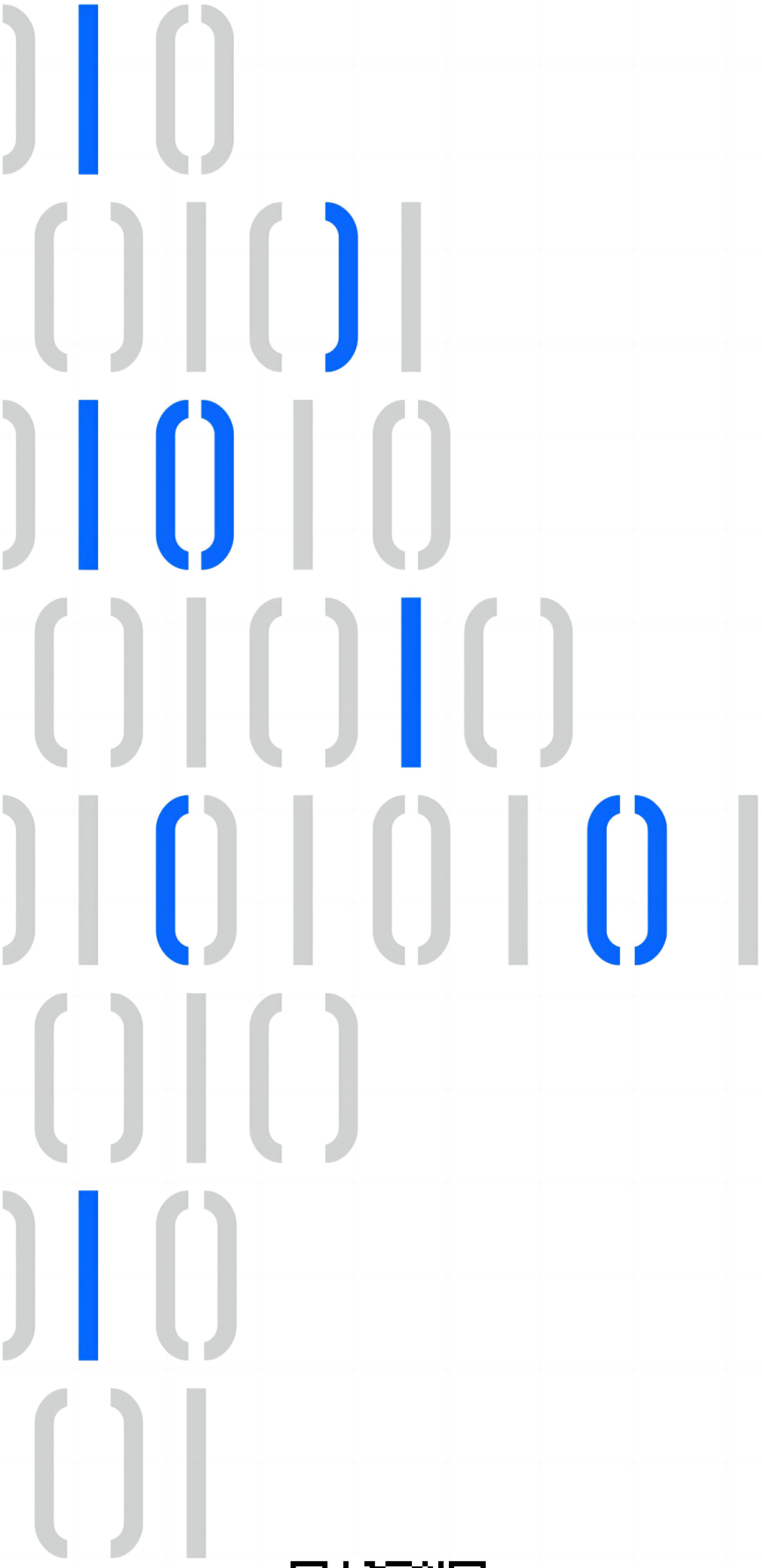
表 9-2 SQL 命令失效报错

可能原因	问题分析	问题处理
语法兼容	SQL 语法或函数不兼容	找出对应的语法或函数，进行 SQL 调整
接下页		

可能原因	问题分析	问题处理
关键字冲突	SQL 语句中包含 XuGu 关键字	对关键字加双引号或进行重命名
系统表不存在	SQL 语句中使用了 PostgreSQL 系统表或系统视图	在 XuGu 中封装对应的元信息表或替换成 XuGu 系统表

Appendix A PostgreSQL 常用函数

PostgreSQL 常用函数参照文档：**http://www.postgres.cn/docs/14/functions.html**



成都虚谷伟业科技有限公司

联系电话：400-8886236

官方网站：www.xugudb.com