pg默认端口

5432

pg插件有3个：

geos-3.5.0

proj-4.7.0

postgis-1.5.2

windows测试机器的密码

Adminstrator

Welcome8881234

# 1. psql: FATAL: database “<user>” does not exist

**参考文章：**

<https://stackoverflow.com/questions/17633422/psql-fatal-database-user-does-not-exist>

**问题原因：**

没有通过-d指定数据库的话，默认会连接到与用户名同名的数据库，而此数据库不存在，所以报了这个错误。

**解决方案：**

通过-d 参数指定要连接的数据库，否则默认会连接与用户名同名的数据库。

|  |
| --- |
| psql -d template1 |

# 2. **删除postgresql数据库时报错database xxx is being accessed by other users**

**参考文章：**

<https://blog.csdn.net/qq_24095055/article/details/90713678>

**问题原因：**

有其他session正在连接着这个数据库，因此不让删。

**解决方案：**

执行如下语句

|  |
| --- |
| SELECT pg\_terminate\_backend(pg\_stat\_activity.pid)  FROM pg\_stat\_activity  WHERE datname='emap' AND pid<>pg\_backend\_pid();  -再删除库  drop database emap; |

# 3. psql 导入数据报错 invalid command \N

**参考文章：**<https://www.cnblogs.com/abclife/p/15104748.html>

**问题原因：**

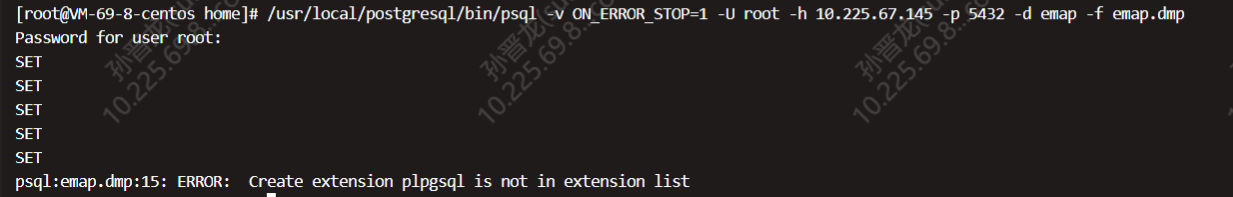
由于源端和目标数据库的权限设置可能不一致，在数据导入过程当中可能会出现一些与权限相关的 WARNING 或 ERROR，可以忽略。

**解决方案：**

通过-v ON\_ERROR\_STOP=1 参数，只要导入一遇到ERROR就会停止了，而且能看到具体什么原因导致导入数据报错，根据具体错误查找解决办法。

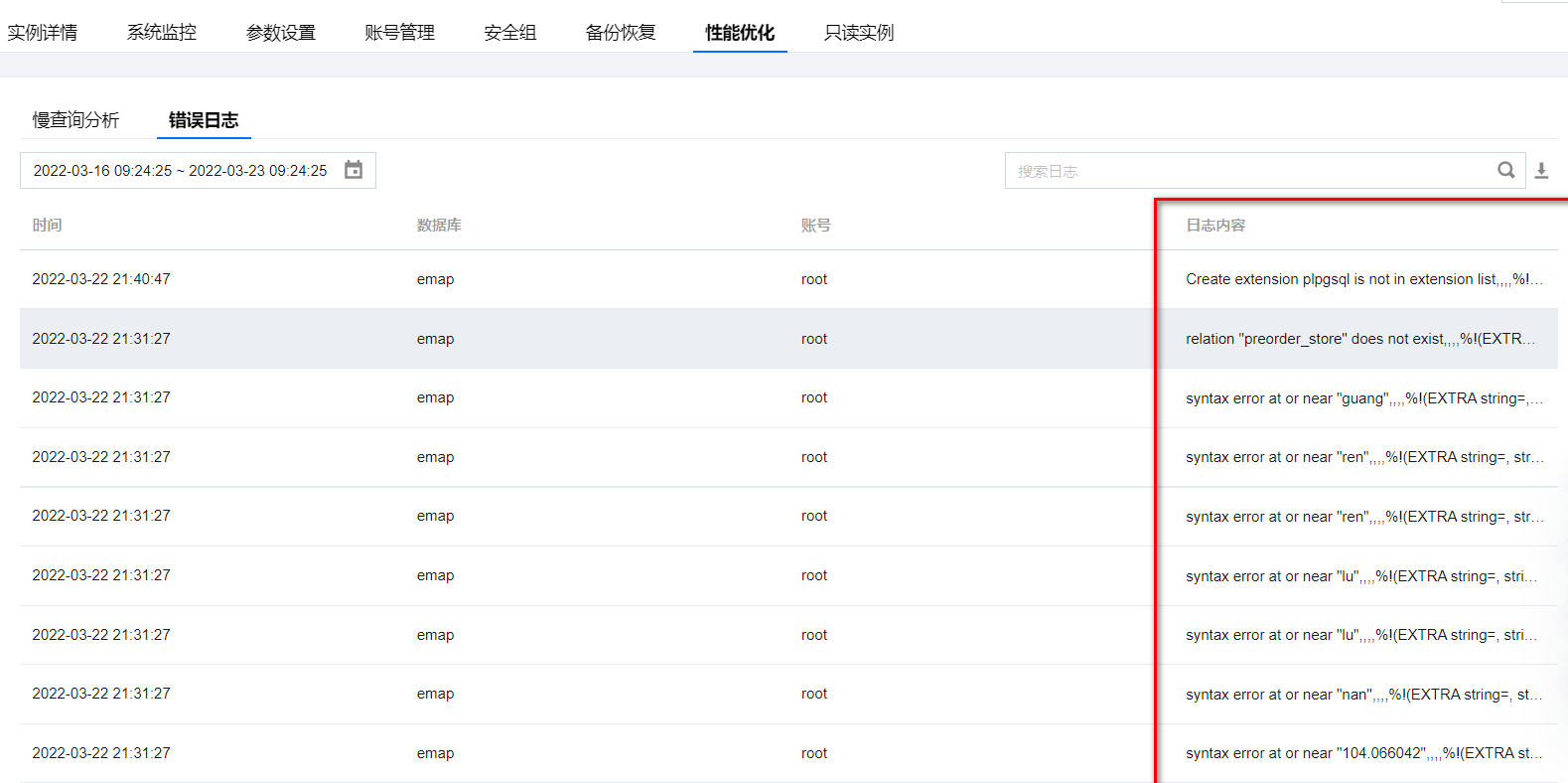
|  |
| --- |
| /usr/local/postgresql/bin/psql -v ON\_ERROR\_STOP=1 -U root -h 10.225.67.145 -p 5432 -d emap -f emap.dmp |

如下图所示：



# 4. ERROR: Create extension plpgsql is not in extension list

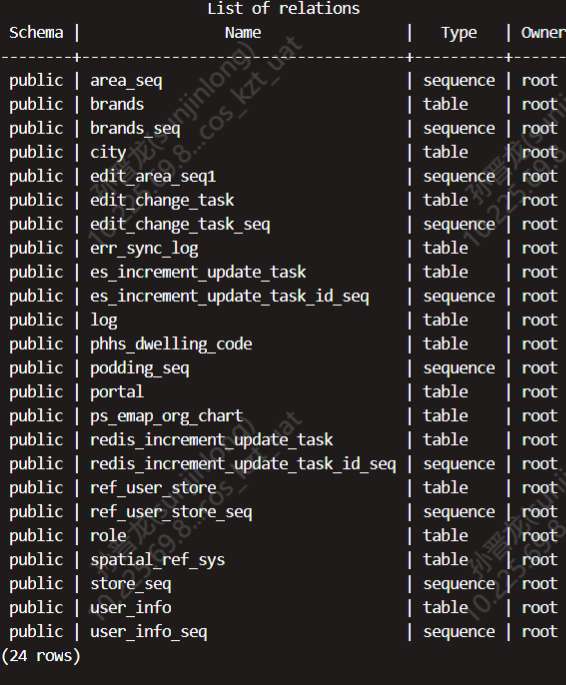
这个跟咱们后端核实了下 是创建插件 可以忽略的。----来自腾讯云售后群



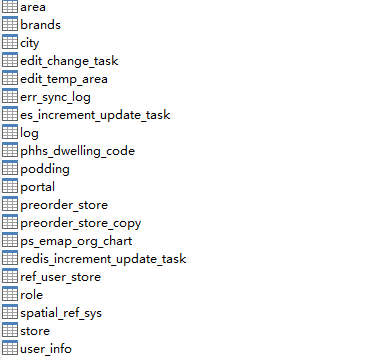
在腾讯云rds的postgresql中的日志中出现的这些ERROR也是可以忽略的，主要是权限的问题。

# 5. psql导入pg\_dump导出的数据文件之后，发现少了几个表

目标端：



源端：



问题原因：

|  |
| --- |
| 这种数据迁移的方式实际是通过逻辑备份（我理解就是建表/建type/建function等语句建立对应的table/type/function，）然后在目标库执行这些语句来实现的。  由于源库和目标库owner的不同，以及目标库没有指定type，权限等问题，导致create table的时候报错，创建表失败，因此少了很多表。 |

解决方案：

|  |
| --- |
| 手动搞  1. 在阿里云源端导出用户信息  pg\_dumpall -g -h 172.16.4.81 -p 5432 -U postgres > pg\_user\_dump.sql  在腾讯云目标端创建用户，随意命名（因为原来用的postgres用户，与腾讯云上paas冲突，需要在腾讯云paas上重新用一个用户），如pguser  CREATE ROLE pguser;  ALTER ROLE pguser WITH tencent\_superuser INHERIT CREATEROLE CREATEDB LOGIN REPLICATION;  alter user pguser with password 'pguser';  2. 从源库pg\_dump -t 导出所有的table,view, sequence  pg\_dump -U postgres -t area -t brands -t city -t edit\_change\_task -t edit\_temp\_area -t err\_sync\_log -t es\_increment\_update\_task -t geography\_columns -t geometry\_columns -t log -t phhs\_dwelling\_code -t podding -t portal -t preorder\_store -t preorder\_store\_copy -t ps\_emap\_org\_chart -t redis\_increment\_update\_task -t ref\_user\_store -t role -t spatial\_ref\_sys -t store -t user\_info -t area\_seq -t brands\_seq -t edit\_area\_seq1 -t edit\_change\_task\_seq -t es\_increment\_update\_task\_id\_seq -t podding\_seq -t redis\_increment\_update\_task\_id\_seq -t ref\_user\_store\_seq -t store\_seq -t user\_info\_seq emap > ./emap2.dmp    pg\_dump -U postgres -t area -t area\_seq -t brands -t brands\_seq -t city -t edit\_area\_seq1 -t edit\_change\_task -t edit\_change\_task\_seq -t edit\_temp\_area -t err\_sync\_log -t es\_increment\_update\_task -t es\_increment\_update\_task\_id\_seq -t geography\_columns -t geometry\_columns -t log -t phhs\_dwelling\_code -t podding -t podding\_seq -t portal -t preorder\_store -t ps\_emap\_org\_chart -t redis\_increment\_update\_task -t redis\_increment\_update\_task\_id\_seq -t ref\_user\_store -t ref\_user\_store\_seq -t role -t spatial\_ref\_sys -t store -t store\_seq -t user\_info -t user\_info\_seq emap\_ksf > ./emap\_ksf.dmp    pg\_dump -U postgres -t area -t area\_seq -t brands -t brands\_seq -t city -t edit\_area\_seq1 -t edit\_change\_task -t edit\_change\_task\_seq -t edit\_temp\_area -t err\_sync\_log -t es\_increment\_update\_task -t es\_increment\_update\_task\_id\_seq -t geography\_columns -t geometry\_columns -t log -t phhs\_dwelling\_code -t podding -t podding\_seq -t portal -t preorder\_store -t preorder\_store\_copy -t ps\_emap\_org\_chart -t redis\_increment\_update\_task -t redis\_increment\_update\_task\_id\_seq -t ref\_user\_store -t ref\_user\_store\_seq -t role -t spatial\_ref\_sys -t store -t store\_seq -t user\_info -t user\_info\_seq emap\_npls > ./emap\_npls.dmp  3. 向目标端pg导入数据  jumpserver登录10.225.69.8    /usr/local/postgresql/bin/psql -U pguser -h 10.225.67.145 -p 5432 -d template1  --创建数据库  create database emap;  --进入数据库  \c emap;  --创建插件  create extension postgis;    cd /home  vim emap2.dmp  :%s/postgres/pguser/g  :%s/readaccess/pguser/g    nohup /usr/local/postgresql/bin/psql -U pguser -h 10.225.67.145 -p 5432 -d emap -f emap2.dmp > load\_emap.log 2>&1    4. 手动创建function  从日志文件中找到报错的function，在 emap.dmp 中找到此function的建立语句和trigger的语句，在目标端pg执行即可    psql:emap2.dmp:6834012: ERROR: function adddelareatask() does not exist  psql:emap2.dmp:6834019: ERROR: function adddelpoddingtask() does not exist  psql:emap2.dmp:6834026: ERROR: function adddelportaltask() does not exist  psql:emap2.dmp:6834033: ERROR: function adddelpreorderstoretask() does not exist  psql:emap2.dmp:6834040: ERROR: function adddelpreorderstoretask() does not exist  psql:emap2.dmp:6834047: ERROR: function adddelstoretask() does not exist  psql:emap2.dmp:6834054: ERROR: function addinsertareatask() does not exist  psql:emap2.dmp:6834061: ERROR: function addinsertpoddingtask() does not exist  psql:emap2.dmp:6834068: ERROR: function addinsertportaltask() does not exist  psql:emap2.dmp:6834075: ERROR: function addinsertpreorderstoretask() does not exist  psql:emap2.dmp:6834082: ERROR: function addinsertpreorderstoretask() does not exist  psql:emap2.dmp:6834089: ERROR: function addinsertstoretask() does not exist  psql:emap2.dmp:6834096: ERROR: function addupdareatask() does not exist  psql:emap2.dmp:6834103: ERROR: function addupdpoddingtask() does not exist  psql:emap2.dmp:6834110: ERROR: function addupdportaltask() does not exist  psql:emap2.dmp:6834117: ERROR: function addupdpreorderstoretask() does not exist  psql:emap2.dmp:6834124: ERROR: function addupdpreorderstoretask() does not exist  psql:emap2.dmp:6834131: ERROR: function addupdstoretask() does not exist  psql:emap2.dmp:6834138: ERROR: function updatexy() does not exist        CREATE FUNCTION adddelareatask() RETURNS trigger  LANGUAGE plpgsql  AS $$  DECLARE  recodeCount integer :=1;  BEGIN  INSERT INTO es\_increment\_update\_task(table\_name,table\_id,operate) values('area',OLD.area\_id,'del');  return OLD;  END;  $$;  CREATE TRIGGER adddelareatasktrigger BEFORE DELETE ON area FOR EACH ROW EXECUTE PROCEDURE adddelareatask();      CREATE FUNCTION adddelpoddingtask() RETURNS trigger  LANGUAGE plpgsql  AS $$  DECLARE  recodeCount integer :=1;  BEGIN  INSERT INTO redis\_increment\_update\_task(table\_name,table\_id,operate) values('area',OLD.area\_id,'upd');  INSERT INTO es\_increment\_update\_task(table\_name,table\_id,operate) values('podding',OLD.podding\_id,'del');  return OLD;  END;  $$;  CREATE TRIGGER adddelpoddingtasktrigger BEFORE DELETE ON podding FOR EACH ROW EXECUTE PROCEDURE adddelpoddingtask();      CREATE FUNCTION adddelportaltask() RETURNS trigger  LANGUAGE plpgsql  AS $$  BEGIN  INSERT INTO redis\_increment\_update\_task(table\_name,table\_id,operate) values('portal',OLD.portal\_source,'del');  return OLD;  END;  $$;  CREATE TRIGGER adddelportaltasktrigger BEFORE DELETE ON portal FOR EACH ROW EXECUTE PROCEDURE adddelportaltask();      CREATE FUNCTION adddelpreorderstoretask() RETURNS trigger  LANGUAGE plpgsql  AS $$  BEGIN  INSERT INTO es\_increment\_update\_task(table\_name,table\_id,operate) values('preorder\_store',OLD.store\_code,'del');  INSERT INTO redis\_increment\_update\_task(table\_name,table\_id,operate) values('preorder\_store',OLD.store\_code,'del');  return OLD;  END;  $$;  CREATE TRIGGER adddelpreorderstoretasktrigger BEFORE DELETE ON preorder\_store FOR EACH ROW EXECUTE PROCEDURE adddelpreorderstoretask();  CREATE TRIGGER adddelpreorderstoretasktrigger BEFORE DELETE ON preorder\_store\_copy FOR EACH ROW EXECUTE PROCEDURE adddelpreorderstoretask();      CREATE FUNCTION adddelstoretask() RETURNS trigger  LANGUAGE plpgsql  AS $$  BEGIN  INSERT INTO es\_increment\_update\_task(table\_name,table\_id,operate) values('store',OLD.store\_id,'del');  INSERT INTO redis\_increment\_update\_task(table\_name,table\_id,operate) values('store',OLD.store\_id,'del');  return OLD;  END;  $$;  CREATE TRIGGER adddelstoretasktrigger BEFORE DELETE ON store FOR EACH ROW EXECUTE PROCEDURE adddelstoretask();      CREATE FUNCTION addinsertareatask() RETURNS trigger  LANGUAGE plpgsql  AS $$  DECLARE  recodeCount integer :=1;  BEGIN  INSERT INTO es\_increment\_update\_task(table\_name,table\_id,operate) values('area',NEW.area\_id,'add');  return NEW;  END;  $$;  CREATE TRIGGER addinsertareatasktrigger AFTER INSERT ON area FOR EACH ROW EXECUTE PROCEDURE addinsertareatask();      CREATE FUNCTION addinsertpoddingtask() RETURNS trigger  LANGUAGE plpgsql  AS $$  DECLARE  recodeCount integer :=1;  BEGIN  INSERT INTO redis\_increment\_update\_task(table\_name,table\_id,operate) values('area',NEW.area\_id,'upd');  INSERT INTO es\_increment\_update\_task(table\_name,table\_id,operate) values('podding',NEW.podding\_id,'add');  return NEW;  END;  $$;  CREATE TRIGGER addinsertpoddingtasktrigger AFTER INSERT ON podding FOR EACH ROW EXECUTE PROCEDURE addinsertpoddingtask();      CREATE FUNCTION addinsertportaltask() RETURNS trigger  LANGUAGE plpgsql  AS $$  BEGIN  INSERT INTO redis\_increment\_update\_task(table\_name,table\_id,operate) values('portal',NEW.portal\_source,'add');  return NEW;  END;  $$;  CREATE TRIGGER addinsertportaltasktrigger AFTER INSERT ON portal FOR EACH ROW EXECUTE PROCEDURE addinsertportaltask();      CREATE FUNCTION addinsertpreorderstoretask() RETURNS trigger  LANGUAGE plpgsql  AS $$  BEGIN  INSERT INTO es\_increment\_update\_task(table\_name,table\_id,operate) values('preorder\_store',NEW.store\_code,'add');  INSERT INTO redis\_increment\_update\_task(table\_name,table\_id,operate) values('preorder\_store',NEW.store\_code,'add');  return NEW;  END;  $$;  CREATE TRIGGER addinsertpreorderstoretasktrigger AFTER INSERT ON preorder\_store FOR EACH ROW EXECUTE PROCEDURE addinsertpreorderstoretask();  CREATE TRIGGER addinsertpreorderstoretasktrigger AFTER INSERT ON preorder\_store\_copy FOR EACH ROW EXECUTE PROCEDURE addinsertpreorderstoretask();      CREATE FUNCTION addinsertstoretask() RETURNS trigger  LANGUAGE plpgsql  AS $$  BEGIN  INSERT INTO es\_increment\_update\_task(table\_name,table\_id,operate) values('store',NEW.store\_id,'add');  INSERT INTO redis\_increment\_update\_task(table\_name,table\_id,operate) values('store',NEW.store\_id,'add');  return NEW;  END;  $$;  CREATE TRIGGER addinsertstoretasktrigger AFTER INSERT ON store FOR EACH ROW EXECUTE PROCEDURE addinsertstoretask();        CREATE FUNCTION addupdareatask() RETURNS trigger  LANGUAGE plpgsql  AS $$  BEGIN  INSERT INTO es\_increment\_update\_task(table\_name,table\_id,operate) values('area',NEW.area\_id,'upd');  INSERT INTO redis\_increment\_update\_task(table\_name,table\_id,operate) values('area',NEW.area\_id,'upd');  NEW.updatetime := current\_timestamp;  return NEW;  END;  $$;  CREATE TRIGGER addupdareatasktrigger BEFORE UPDATE ON area FOR EACH ROW EXECUTE PROCEDURE addupdareatask();      CREATE FUNCTION addupdpoddingtask() RETURNS trigger  LANGUAGE plpgsql  AS $$  BEGIN  INSERT INTO es\_increment\_update\_task(table\_name,table\_id,operate) values('podding',NEW.podding\_id,'upd');  INSERT INTO redis\_increment\_update\_task(table\_name,table\_id,operate) values('area',NEW.area\_id,'upd');  NEW.updatetime := current\_timestamp;  return NEW;  END;  $$;  CREATE TRIGGER addupdpoddingtasktrigger BEFORE UPDATE ON podding FOR EACH ROW EXECUTE PROCEDURE addupdpoddingtask();      CREATE FUNCTION addupdportaltask() RETURNS trigger  LANGUAGE plpgsql  AS $$  BEGIN  INSERT INTO redis\_increment\_update\_task(table\_name,table\_id,operate) values('portal',NEW.portal\_source,'upd');  return NEW;  END;  $$;  CREATE TRIGGER addupdportaltasktrigger AFTER UPDATE ON portal FOR EACH ROW EXECUTE PROCEDURE addupdportaltask();      CREATE FUNCTION addupdpreorderstoretask() RETURNS trigger  LANGUAGE plpgsql  AS $$  BEGIN  INSERT INTO es\_increment\_update\_task(table\_name,table\_id,operate) values('preorder\_store',NEW.store\_code,'upd');  INSERT INTO redis\_increment\_update\_task(table\_name,table\_id,operate) values('preorder\_store',NEW.store\_code,'upd');  NEW.updatetime := current\_timestamp;  return NEW;  END;  $$;  CREATE TRIGGER addupdpreorderstoretasktrigger BEFORE UPDATE ON preorder\_store FOR EACH ROW EXECUTE PROCEDURE addupdpreorderstoretask();  CREATE TRIGGER addupdpreorderstoretasktrigger BEFORE UPDATE ON preorder\_store\_copy FOR EACH ROW EXECUTE PROCEDURE addupdpreorderstoretask();      CREATE FUNCTION addupdstoretask() RETURNS trigger  LANGUAGE plpgsql  AS $$  BEGIN  INSERT INTO es\_increment\_update\_task(table\_name,table\_id,operate) values('store',NEW.store\_id,'upd');  INSERT INTO redis\_increment\_update\_task(table\_name,table\_id,operate) values('store',NEW.store\_id,'upd');  NEW.updatetime := current\_timestamp;  return NEW;  END;  $$;  CREATE TRIGGER addupdstoretasktrigger BEFORE UPDATE ON store FOR EACH ROW EXECUTE PROCEDURE addupdstoretask();      CREATE FUNCTION updatexy() RETURNS trigger  LANGUAGE plpgsql  AS $$  BEGIN  UPDATE preorder\_store set the\_geom=NEW.the\_geom,longitude=ST\_X(NEW.the\_geom),latitude=ST\_Y(NEW.the\_geom) where upper(preorder\_store.store\_code)=upper(NEW.store\_code);  RETURN NEW;  END;  $$;  CREATE TRIGGER updatexytrigger AFTER UPDATE OF the\_geom ON store FOR EACH ROW EXECUTE PROCEDURE updatexy(); |

emap\_ksf.dmp和emap\_npls.dmp也是同样的做法