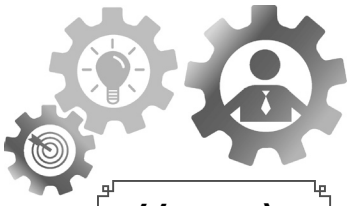

第 5 章

管理用户与会话连接

本章主要讨论关于 KingbaseES 数据库的用户管理和会话连接管理，包括用户和角色的管理、会话连接的管理以及与用户相关的系统配置文件的使用方式。

5.1 用户管理

5.1.1 创建数据库用户

在 KingbaseES 数据库中，可以使用 CREATE USER 或 CREATE ROLE 语句创建一个 KingbaseES 数据库的新用户或新角色。实际上，CREATE USER 是 CREATE ROLE 的别名，它们可以互换使用。用户和角色的区别就是能否登录数据库。无论是使用 CREATE USER 语句还是使用 CREATE ROLE 语句来创建用户(角色)，如果不授予它连接到数据库的权限，那么就可以将这个新创建的用户(角色)，仅当作一个权限集合的角色。

创建数据库集簇的时候，系统会默认创建以下用户：

```
[kingbase@dbsvr ~]$ ksql -d test -U system
system@test=# \du
                                  List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+---------
 kcluster  | Cannot login                                               | {}
 sao       | No inheritance                                             | {}
 sso       | No inheritance                                             | {}
 system    | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
```

从输出可以看到，系统默认创建的用户有 4 个：kcluster、sao、sso 和 system。

KingbaseES 数据库定义了多个系统权限，在创建用户时可以指定这些权限。

(1) 超级用户(superuser)权限：具有该权限的超级用户操作数据库不需要数据库的权限检查。建议非必须最好不要以超级用户的身份执行数据库的管理操作，避免误操作。

(2) 创建用户/角色(createrole)权限：具有该权限的用户可以创建新的用户/角色。

(3) 登录(login)权限：具有该权限的用户才可以建立数据库连接。CREATE USER 语句和 CREATE ROLE 语句的区别就是在创建用户/角色时，是否具有 login 权限。使用 CREATE USER 创建的用户默认具有 login 权限。

(4) 创建数据库(createdb)权限：具有该权限的用户才可以创建一个新的数据库。

(5) 发起流复制(replication)权限：具有该权限的用户能够发起流复制。

在进行系统初始化时，创建了超级用户 system，该用户可以创建其他新用户并执行其系统权限。新创建的用户在任何数据库上都有创建自己的数据库对象(如表)的权限，但不具有其他数据库对象的权限。使用 ALTER USER/ALTER ROLE 语句可以给用户增加或减少系统权限。

例 5.1　创建用户 user1，并给其相应的权限。

打开第 1 个终端，创建用户 newuser，创建数据库 newdb：

```
[kingbase@dbsvr ~]$ ksql -d test -U system
system@test=# CREATE USER newuser WITH PASSWORD 'newuser123';
CREATE ROLE
system@test=# CREATE DATABASE newdb;
CREATE DATABASE
```

打开第 2 个终端，使用用户 newuser 连接到数据库 newdb：

```
[kingbase@dbsvr ~]$ ksql -d newdb -U newuser
Password for user newuser:        #输入用户 newuser 的密码 newuser123
newuser@newdb=> CREATE TABLE ttt1(col int);
newuser@newdb=> CREATE TABLE ttt2(col int);
newuser@newdb=> CREATE DATABASE newdb1;
ERROR:  permission denied to create database
                                  #目前数据库用户 newuser 没有创建数据库的权限
newuser@newdb=> CREATE USER tpchuser;
ERROR:  permission denied to create role
                                  #目前数据库用户 newuser 没有创建数据库用户的权限
newuser@newdb=> \du
          List of roles
 Role name | Attributes | Member of
-------- +----------+-----------
 newuser   |            | {}
```

从输出可以看到，用户 newuser 在数据库 newdb 上成功地创建了两个表，但是没有创建数据库和用户的系统权限。

回到第 1 个终端，给用户 newuser 授予 CREATEROLE 权限，并把数据库 newdb 的属主给 newuser：

```
system@test=# ALTER USER newuser CREATEROLE ;
ALTER ROLE
system@test=# ALTER DATABASE newdb OWNER TO newuser ;
ALTER DATABASE
```

回到第 2 个终端，查看用户 newuser 的属性，并创建新用户：

```
newuser@test=> \du
           List of roles
 Role name | Attributes   | Member of
-------- +-----------+-----------
```

```
 newuser   | Create role  | {}
newuser@test=> CREATE USER tpchuser;
CREATE ROLE
```

从输出可以看到，用户 newuser 有 Create role 的权限，并且现在可以成功地创建用户 tpchuser 了。

回到第 1 个终端，回收用户 newuser 的 CREATEROLE 权限：

```
system@test=# ALTER USER newuser NOCREATEROLE ;
ALTER ROLE
```

例 5.2 设置用户的连接属性。

可以设置用户的属性，例如，该用户可以建立的数据库连接的个数、用户口令的有效期等。

在第 1 个终端，设置用户 newuser 只能建立 1 个数据库连接：

```
system@test=# ALTER USER newuser CONNECTION LIMIT 1;
ALTER ROLE
```

打开第 3 个终端，尝试使用用户 newuser 连接到数据库 test：

```
[kingbase@dbsvr ~]$  ksql -d test -U newuser
Password for user newuser: #输入用户 newuser 的密码 newuser123
ksql: error: could not connect to server: FATAL:  too many connections for role
"newuser"
[kingbase@dbsvr ~]$
```

因为设置了用户 newuser 只能建立 1 个数据库连接，而在第 2 个终端上 newuser 已经建立了一个数据库连接，所以这里会报错。

在第 1 个终端，还可以设置用户 newuser 的口令有效期：

```
system@test=# ALTER USER newuser VALID UNTIL '2030-12-31';
ALTER ROLE
system@test=#
```

为了后面示例方便，不再限制用户 newuser 建立数据库连接的个数：

```
system@test=# ALTER USER newuser CONNECTION LIMIT -1;
ALTER ROLE
```

例 5.3 锁定和解锁数据库用户。

锁定一个用户意味着保留该数据库用户账号，并且不会修改账号的密码，不仅仅是不允许使用该数据库用户账号登录到 KingbaseES 数据库，对于已经建立的数据库连接不受影响。

在第 1 个终端，锁定用户 newuser：

```
system@test=# ALTER USER newuser ACCOUNT LOCK;
```

打开第 3 个终端，尝试使用用户 newuser 连接到数据库 test：

```
[kingbase@dbsvr ~]$ ksql -d test -U newuser
```

```
Password for user newuser: #输入用户 newuser 的密码 newuser123
ksql: error: could not connect to server: FATAL:  role "newuser" is not permitted
to log in
[kingbase@dbsvr ~]$
```

从输出可以看到，由于用户账号 newuser 被锁定，因此无法创建与 KingbaseES 数据库的连接。

回到第 1 个终端，解锁用户 newuser：

```
system@test=# ALTER USER newuser ACCOUNT UNLOCK;
ALTER ROLE
system@test=#
```

回到第 3 个终端，再次尝试使用用户 newuser 连接到数据库 test：

```
[kingbase@dbsvr ~]$ ksql -d test -U newuser
Password for user newuser: #输入用户 newuser 的密码 newuser123
newuser@test=> \q
[kingbase@dbsvr ~]$
```

从输出可以看到，解锁用户账号 newuser 后，可以创建与 KingbaseES 数据库的连接。

5.1.2　删除数据库用户

如果用户正在连接某个数据库，或用户是数据库对象的属主，则不能删除该用户。

例 5.4　删除数据库的用户。

在第 1 个终端上，执行下面的 SQL 语句，删除用户 newuser：

```
system@test=# DROP USER newuser;
ERROR:  current logined user cannot be dropped
system@test=#
```

从输出可以看到，不能删除一个已经连接到 KingbaseES 数据库的用户。

回到第 2 个终端，执行下面的元命令，退出 ksql：

```
newuser@newdb=> \q
[kingbase@dbsvr ~]$
```

再次回到第 1 个终端，继续执行删除用户 newuser 的 SQL 语句：

```
system@test=# DROP USER newuser;
ERROR:  role "newuser" cannot be dropped because some objects depend on it
DETAIL: owner of database newdb
2 objects in database newdb
```

从输出可以看到，现在还是不能删除用户 newuser，原因是一些数据库对象依赖于用户 newuser：

(1) 用户 newuser 是数据库 newdb 的属主；

(2) 数据库 newdb 中有 2 个属于用户 newuser 的数据库对象。

要删除用户 newuser，必须先将数据库的属主修改为其他的用户，或者删除。这里选择将数据库 newdb 的属主修改为用户 system。

在第 1 个终端上运行下面的 SQL 语句：

```
system@test=# ALTER DATABASE newdb OWNER TO system;
```

对数据库 newdb 中属于用户 newuser 的数据库对象，有以下两种处理方法：

(1) 假如该对象(如表 ttt1)没用了，可以直接删除掉该对象；

(2) 假如该对象(如表 ttt2)仍然有用，可以将该对象转移给其他用户(如 system)。

转到第 2 个终端，执行下面的 ksql 命令和 SQL 语句：

```
[kingbase@dbsvr ~]$ ksql -d newdb -U system
system@newdb=# \dt
        List of relations
 Schema | Name | Type  |  Owner
--------+------+-------+---------
 public | ttt1 | table | newuser
 public | ttt2 | table | newuser
(2 rows)
system@newdb=# DROP TABLE ttt1;
system@newdb=# REASSIGN OWNED BY newuser to system;
system@newdb=# \dt
       List of relations
 Schema | Name | Type  | Owner
--------+------+-------+--------
 public | ttt2 | table | system
(1 row)
system@newdb=# \q
[kingbase@dbsvr ~]$
```

转回到第 1 个终端，执行下面的 SQL 语句，删除用户 newuser：

```
system@test=#   DROP USER newuser;
DROP ROLE
system@test=# \du newuser
           List of roles
 Role name | Attributes | Member of
---------- +------------+-----------


system@test=#
```

从输出可以看到，数据库用户 newuser 已经被删除。

5.1.3 查询用户信息

通过系统视图 sys_shadow、sys_user、sys_roles 可以查询用户的相关信息。使用 ksql 的元命令\du 或\dg，也可以查看到 Kingbase 数据库上所有的用户和角色。

例 5.5 查看系统上的用户/角色信息。

查看不可以登录到 KingbaseES 数据库的用户/角色：

```
system@test=# SELECT rolname FROM sys_roles WHERE rolcanlogin='f';
          rolname
----------------------------
```

```
 pg_monitor
--省略了许多输出
(9 rows)
```

例 5.6　查看为用户设置的系统配置参数信息。

```
system@test=# ALTER USER tpchuser SET work_mem='10MB';
system@test=# SELECT  *  FROM sys_user;
usename  | usesysid | usecreatedb | usesuper | userepl | usebypassrls |  passwd  | valuntil |    useconfig
-------+-------+---------+-------+------+----------+-------+-------+---------------
 system  |       10 | t           | t        | t       | t            | ********  |          |
 sao     |        9 | f           | f        | f       | f            | ********  |          |
 sso     |        8 | f           | f        | f       | f            | ********  |          |
 tpchuser|    16512 | f           | f        | f       | f            | ********  |          | {work_mem=10MB}
(4 rows)
```

从输出可以看到,为用户 tpchuser 设置了参数 work_mem。

例 5.7　查看用户是否设置了密码。

```
system@test=# CREATE USER newrole;
system@test=# SELECT usename, passwd FROM sys_shadow;
usename  |                                              passwd
-------+------------------------------------------------------------------------------------------
 system  | SCRAM-SHA-256$4096:R7zhQyJxrwhc8sCMDxSCoQ==$D7oLvisZNWnN3zKM4OBgG72VkWAd1Ba2qZus275HLUE=:5Uo+fSQwzuQc9tSClgtdvy+e9GcwVc9076MO85mUzds=
 sao     | SCRAM-SHA-256$4096:6pOpAwuNdkwNHyJ1W6MvZA==$WsUmBS7H11GZtlyFoLyXL3B0jBBOdo8nZrn8vYq4xH8=:5GQlsf1MMGvb27e3yAhs9rgQo0Av0wpunTHBlzMSURo=
 sso     | SCRAM-SHA-256$4096:oW48mmQKo4yxibNDiSD9Aw==$NgpPSuOK1iSQ+CppdhUGUgNkW9ZnQ8rU3w0c3ImlqPg=:2ezBhkZTp48TnPqoak1WQSOpBwqYUBaA1waXLCp54B0=
 tpchuser|
 newrole |
(5 rows)
```

从输出可以看到,目前还没有为 newrole 和 tpchuser 设置过密码。在 KingbaseES 数据库巡检时,可以使用这条 SQL 语句,检查是否存在系统安全漏洞。

例 5.8　查看指定名字的用户有哪些系统权限。

```
system@test=# SELECT rolname AS username,
test-#       CASE WHEN rolsuper THEN 'superuser' ELSE '' END AS superuser,
test-#       CASE WHEN rolcreaterole THEN 'create role' ELSE '' END AS create_role,
test-#       CASE WHEN rolcreatedb THEN 'create db' ELSE '' END AS create_db,
test-#       CASE WHEN rolreplication THEN 'replication' ELSE '' END AS replication
test-#     FROM sys_roles
test-#     WHERE rolname = 'newrole';
 username | superuser | create_role | create_db | replication
--------+--------+--------- +--------+-------------
 newrole  |           |             |           |
(1 row)
```

从输出可以看到,当前用户 newrole 还没有任何系统权限。

5.2　管理连接会话

用户访问数据库,首先要建立与数据库服务器的连接,称为连接会话。

5.2.1　设置与会话连接相关的系统配置参数

在系统配置文件 kingbase.conf 中必须进行正确的配置,用户才能建立与数据库服务器

的连接，相关的配置如下。

1. 监听的 IP 地址和端口号

KingbaseES 数据库使用参数 listen_addresses 和 port 来配置监听的 IP 地址和端口号。默认的配置如下：

```
listen_addresses = '*'
port=54321
```

默认配置的含义是 KingbaseES 数据库服务器的 kingbase 进程，将监听服务器上所有的 IP 地址，监听 TCP 端口号 54321。

可以修改系统配置文件中的这两个参数的值，让 KingbaseES 数据库服务器只监听指定的 IP 地址和 TCP 端口号，例如，可将这两个参数的值按如下设置：

```
listen_addresses = '192.168.100.22' # 设置为 KingbaseES 数据库服务器对外服务的网卡地址
port=54321                          # 设置监听端口
```

2. 最大连接数

在 kingbase.conf 文件中设置参数 max_connections，指定 KingbaseES 数据库服务器允许的最大连接数，参数 max_connections 的默认值为 100，可以在 kingbase.conf 文件中根据应用的实际需求修改该值，例如，将其设置为 300：

```
max_connections=300                 # 设置最大连接会话数
```

该值需要在数据库重新启动后生效。

3. 为超级用户保留的最大连接数

参数 superuser_reserved_connections 设置了为超级用户保留的连接数，默认值是 3。为超级用户保留连接数的目的是如果应用程序把系统所有的用户连接数使用完后，超级用户还有机会连接数据库进行数据库管理。

如果设置了参数 superuser_reserved_connections，则普通用户可以使用的最大连接数为 max_connections － superuser_reserved_connections。

注意：参数 superuser_reserved_connections 设置的是为超级用户保留的连接数，并不是超级用户可以建立的最大连接数。

5.2.2 查看会话连接信息

KingbaseES 数据库的动态系统视图 sys_stat_activity 提供了关于当前数据库服务器所有进程的信息。每一行对应一个数据库服务器进程，主要包括以下类型的信息。

1. 数据库服务器进程的基本信息

(1) backend_type：数据库服务器进程的类型，例如，client backend 表示服务进程，autovacuum launcher、logical replication launcher 等，表示后台进程。

(2) pid：数据库服务器进程 ID，是操作系统的进程号。

2. 会话连接的信息

如果是服务进程，则包含该连接会话的用户信息、客户端的地址等信息、连接数据库信

息以及应用程序的名称。如果是其他后台进程,则这些字段为空。

(1) usesysid：用户的 OID。

(2) usename：执行当前进程的用户的名称。

(3) client_addr：发起数据库连接的客户端的 IP 地址。

(4) client_hostname：发起连接的客户端的主机名。

(5) client_port：与数据库通信的客户端的端口号。

(6) datid：数据库的 OID。

(7) datname：当前进程所在数据库的名称。

(8) application_name：连接到数据库的应用程序的名称。

3. 会话连接的状态信息

会话连接的状态信息包括进程的当前状态、进程启动时间、事务的启动时间、当前正在执行的 SQL 语句以及查询的启动时间等。

(1) state：进程的当前状态,有以下取值。

① active：正在执行一个查询。

② idle：正在等待客户端发送新的命令。

③ idle in transaction：在事务中,但是没有执行命令。

④ idle in transaction (aborted)：在事务中,没有执行命令,事务中有语句出现了错误。

⑤ fastpath function call：在执行一个 fastpath 函数。

(2) backend_start：当前数据库后台进程启动的时间。

(3) xact_start：当前进程开始最新事务的时间。

(4) query_start：当前正在执行的查询开始的时间。

(5) state_change：进程最后一次改变状态的时间。

(6) backend_xid：当前事务的事务 ID(如果进程在一个事务中)。

(7) backend_xmin：当前进程可见的最早未提交的事务 ID。

(8) query：当前进程正在执行的 SQL 语句。

4. 会话连接的等待时间

(1) wait_event_type：进程当前正在等待的事件类型(如果有),事件类别包括轻量级锁((wlock),I/O,客户端,IPC)等。

(2) wait_event：进程当前正在等待的具体事件。

例 5.9　查看数据库服务器进程的状态信息。

系统视图 sys_stat_activity 反映的是数据库进程的实时信息。

首先,在 **KingbaseES 数据库客户端**上,执行一条会持续运行很长时间的 SQL 语句：

```
[kingbase@dbclient ~]$ ksql -h 192.168.100.22 -d test -U system
system@test=# DROP INDEX tpch.lineitem_l_partkey_idx;
--说明:删除该索引将导致接下来的 SQL 语句需要运行很长的时间
system@test=# select sum(l_extendedprice) / 7.0 as avg_yearly
test-#     from tpch.lineitem,tpch.part
test-#     where p_partkey = l_partkey and p_brand = 'Brand#45' and p_container = 'LG PKG'
test-#       and l_quantity < ( select 0.2 * avg(l_quantity)
```

```
test(#                          from tpch.lineitem
test(#                          where l_partkey = p_partkey );
--SQL语句运行中……
```

然后，在 **KingbaseES 数据库服务器**上使用系统视图 sys_stat_activity 查看信息：

```
[kingbase@dbsvr ~]$ ksql -d test -U system
system@test=# \! date
Sat Mar  2 02:51:02 CST 2024
system@test=# SELECT pid,state,query_start, query  FROM sys_stat_activity;
  pid  | state  |         query_start          |                                    query
-------+--------+------------------------------+-------------------------------------------------------------------
  2040 |        |                              |
  2048 | idle   |                              |
  2049 |        |                              |
 69028 | active | 2024-03-02 02:51:09.967257+08 | SELECT pid,state,query_start, query  FROM sys_stat_activity;
 68855 | active | 2024-03-02 02:41:11.076161+08 | select sum(l_extendedprice) / 7.0 as avg_yearly                    +
       |        |                              | from tpch.lineitem,tpch.part                                       +
       |        |                              | where p_partkey = l_partkey and p_brand = 'Brand#45' and p_container = 'LG PKG'    +
       |        |                              |       and l_quantity < ( select 0.2 * avg(l_quantity)              +
       |        |                              |                          from tpch.lineitem                        +
       |        |                              |                          where l_partkey = p_partkey );
  2038 |        |                              |
  2037 |        |                              |
  2039 |        |                              |
(8 rows)
```

从输出可以看出，查询的开始时间为 02:41:11.07，查询会话信息的时间为 02:51:02，语句运行了约 10min，目前还在运行中。

例 5.10　查找长时间运行 SQL 查询的会话连接。

查找长时间(大于 5min)运行 SQL 查询的会话：

```
system@test=# \x
Expanded display is on.
system@test=# SELECT * FROM sys_stat_activity
test-#        WHERE state = 'active' AND now() - query_start > INTERVAL '5 minutes';
-[ RECORD 1 ]----+-----------------------------------------------------------------------
datid            | 14386
datname          | test
pid              | 68855
usesysid         | 10
usename          | system
application_name | ksql
client_addr      | 192.168.100.18
client_hostname  |
client_port      | 58822
backend_start    | 2024-03-02 02:41:02.905953+08
xact_start       | 2024-03-02 02:41:11.076161+08
query_start      | 2024-03-02 02:41:11.076161+08
state_change     | 2024-03-02 02:41:11.076163+08
wait_event_type  |
wait_event       |
state            | active
backend_xid      |
backend_xmin     | 1119
query            | select sum(l_extendedprice) / 7.0 as avg_yearly                    +
                 | from tpch.lineitem,tpch.part                                       +
                 | where p_partkey = l_partkey and p_brand = 'Brand#45' and p_container = 'LG PKG'    +
                 |       and l_quantity < ( select 0.2 * avg(l_quantity)              +
                 |                          from tpch.lineitem                        +
```

```
                |                        where l_partkey = p_partkey );
backend_type    | client backend

system@test=# \x
Expanded display is off.
```

从输出可以看到，这个长时间执行 SQL 查询的会话进程的 PID 是 68855。

例 5.11　查看系统等待事件的情况。

可以使用下面的查询，查看系统等待事件的情况：

```
system@test=# SELECT pid, wait_event_type, wait_event
test-#         FROM pg_stat_activity
test-#         WHERE wait_event is NOT NULL;
  pid  | wait_event_type |     wait_event
------ +-----------------+---------------------
--省略了一些输出
 68855 | IO              | DataFileRead
--省略了一些输出
(7 rows)
```

5.2.3　处理有问题的连接会话

当一个会话长时间不响应时，KingbaseES 数据库的 DBA 可能需要通过取消这个会话中正在运行的 SQL 语句(会话继续保持)，或者删除这个会话来释放占用的数据库资源。

例 5.12　取消会话中运行的 SQL 语句。

在 5.2.2 节中执行时间超过 5min 的会话进程的 PID 是 68855。执行下面的 SQL 语句，取消这个正在运行的 SQL 语句：

```
system@test=# SELECT pg_cancel_backend(68855);
 pg_cancel_backend
--------------------
 t
(1 row)
```

运行 SQL 语句的会话终端，将显示如下信息：

```
ERROR:  canceling statement due to user request
system@test=#
```

在这个会话上重新运行这条 SQL 语句。

例 5.13　终止执行时间比较长的会话连接。

如果想终止这个会话(进程 PID 是 68855)，执行以下步骤：

在 KingbaseES 数据库服务器上，执行 kbps 脚本，再次确认系统有 PID 是 68855 的会话：

```
[kingbase@dbsvr ~]$ kbps
--省略了许多输出
kingbase  68855   1887 99 02:41 ?        00:21:14 kingbase: system test 192.168.
100.18(58822) SELECT
--省略了输出
```