**OceanBase SQL权限控制**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **编号** | **文档版本** | **修订章节** | **修订原因** | **修订日期** | **修订人** |
| **1** | 0.1 |  | 新建文档 | 2012/10/22 | 竹翁 |
| **2** | 0.2 |  | 新增ALTER USER用于修改密码及锁定用户；\_\_users表增加locked列；算法流程修订，权限更新不涉及CS, UPS；用户拥有其创建表的所有权限，无需授权。 | 2012/10/23 | 竹翁 |
| **3** | 0.3 |  | 去除\_\_users表的salt字段。\_\_users表字段主键变更为user。增加并发控制小节 | 2012-10-29 | 立德 |

目录

[1. 用户及权限管理语句 3](#_Toc338754038)

[1.1. 新建用户 3](#_Toc338754039)

[1.2. 删除用户 3](#_Toc338754040)

[1.3. 修改密码 3](#_Toc338754041)

[1.4. 重命名用户 3](#_Toc338754042)

[1.5. 锁定用户 4](#_Toc338754043)

[1.6. 授予权限 4](#_Toc338754044)

[1.7. 撤回权限 5](#_Toc338754045)

[1.8. 查看权限 5](#_Toc338754046)

[2. 用例 5](#_Toc338754047)

[3. 实现 6](#_Toc338754048)

[3.1. 内部系统表 6](#_Toc338754049)

[3.1.1. \_\_users 6](#_Toc338754050)

[3.1.2. \_\_table\_privileges 7](#_Toc338754051)

[3.2. 初始化 7](#_Toc338754052)

[3.2.1. \_\_users 7](#_Toc338754053)

[3.2.2. \_\_table\_privileges 7](#_Toc338754054)

[3.2.3. \_\_all\_sys\_stat 7](#_Toc338754055)

[3.3. 算法流程 7](#_Toc338754056)

[3.4. 认证流程 8](#_Toc338754057)

[3.4.1. 建立连接时 8](#_Toc338754058)

[3.4.2. 执行SQL语句时 8](#_Toc338754059)

# 用户及权限管理语句

## 新建用户

CREATE USER ***user\_specification***

[, ***user\_specification***] ...

***user\_specification***:

***user*** IDENTIFIED BY '***password***'

必须拥有全局的CREATE USER权限或对\_\_users表的INSERT权限，才可以使用CREATE USER命令。新建成功后，\_\_users表会新增一行该用户的表项，新建的用户没有任何权限。如果同名用户已经存在，则报错。

用户名格式与一个普通SQL字符串相同，如’Tom123’, ‘Han MeiMei’, 等。

此处密码为明文，存入\_\_users表后，服务器端会变为密文存储下来。

## 删除用户

DROP USER ***user*** [, ***user***] ...

必须拥有全局的CREATE USER权限或对\_\_users表的DELETE权限，才可以使用DROP USER命令。成功删除用户后，这个用户的**所有权限**也会被一同删除。

## 修改密码

SET PASSWORD [FOR ***user***] = '***password***'

或

ALTER USER ***user*** IDENTIFIED BY '***password***'

如果没有For user子句，则修改当前用户的密码。任何成功登陆的用户都可以修改当前用户的密码。

如果有For user子句，或使用第二种语法，则修改指定用户的密码。必须拥有对\_\_users表的UPDATE权限，才可以修改制定用户的密码。

## 重命名用户

RENAME USER ***old\_user*** TO ***new\_user***

[, ***old\_user*** TO ***new\_user***] ...

重命名指定用户的用户名，成功后，新用户拥有原用户的所有权限。必须拥有全局CREATE USER权限或者对\_\_users表的UPDATE权限，才可以使用本命令。

## 锁定用户

ALTER USER ***user*** ***lock\_specification***

***lock\_specification*** :

LOCKED

| UNLOCKED

锁定或者解锁用户。被锁定的用户不允许登陆。必须拥有对\_\_users表的UPDATE权限，才可以执行本命令。

## 授予权限

GRANT

***priv\_type*** [, ***priv\_type***] ...

ON ***priv\_level***

TO ***user\_specification*** [, ***user\_specification***] ...

***priv\_level***:

\*

| ***tbl\_name***

***user\_specification***:

***user***

给特定用户授予权限。如果用户不存在则报错。当前用户必须拥有被授予的权限，并且拥有GRANT OPTION权限，才能授予成功。

**可以GRANT 和REVOKE的权限**

| **权限** | **说明** |
| --- | --- |
| [**ALL [PRIVILEGES]**](http://dev.mysql.com/doc/refman/5.0/en/privileges-provided.html#priv_all) | 授予除了 [**GRANT OPTION**](http://dev.mysql.com/doc/refman/5.0/en/privileges-provided.html#priv_grant-option)以外所有权限 |
| [**ALTER**](http://dev.mysql.com/doc/refman/5.0/en/privileges-provided.html#priv_alter) | 授权使用 [**ALTER TABLE**](http://dev.mysql.com/doc/refman/5.0/en/alter-table.html) |
| [**CREATE**](http://dev.mysql.com/doc/refman/5.0/en/privileges-provided.html#priv_create) | 授权使用**CREATE TABLE** |
| [**CREATE USER**](http://dev.mysql.com/doc/refman/5.0/en/privileges-provided.html#priv_create-user) | 授权使用[**CREATE USER**](http://dev.mysql.com/doc/refman/5.0/en/create-user.html), [**DROP USER**](http://dev.mysql.com/doc/refman/5.0/en/drop-user.html), [**RENAME USER**](http://dev.mysql.com/doc/refman/5.0/en/rename-user.html), and [**REVOKE ALL PRIVILEGES**](http://dev.mysql.com/doc/refman/5.0/en/revoke.html) |
| [**DELETE**](http://dev.mysql.com/doc/refman/5.0/en/privileges-provided.html#priv_delete) | 授权使用 [**DELETE**](http://dev.mysql.com/doc/refman/5.0/en/delete.html) |
| [**DROP**](http://dev.mysql.com/doc/refman/5.0/en/privileges-provided.html#priv_drop) | 授权使用**DROP TABLE** |
| [**GRANT OPTION**](http://dev.mysql.com/doc/refman/5.0/en/privileges-provided.html#priv_grant-option) | 授权使用**GRANT**和**REVOKE** |
| [**INSERT**](http://dev.mysql.com/doc/refman/5.0/en/privileges-provided.html#priv_insert) | 授权使用[**INSERT**](http://dev.mysql.com/doc/refman/5.0/en/insert.html) |
| [**SELECT**](http://dev.mysql.com/doc/refman/5.0/en/privileges-provided.html#priv_select) | 授权使用[**SELECT**](http://dev.mysql.com/doc/refman/5.0/en/select.html) |
| [**UPDATE**](http://dev.mysql.com/doc/refman/5.0/en/privileges-provided.html#priv_update) | 授权使用[**UPDATE**](http://dev.mysql.com/doc/refman/5.0/en/update.html) |

这些是基本的权限，以后会根据需求逐步增加新的权限。

本用户自动拥有自己创建的对象（目前基本上只有表）。例如，用户user1创建了表t1,t2,那用户user1应该自动就有对t1,t2的ALL PRIVILEGES及GRANT OPTION权限，不再需要额外去授权。

## 撤回权限

REVOKE

***priv\_type*** [, ***priv\_type***] ...

ON ***priv\_level***

FROM ***user*** [, ***user***] ...

REVOKE ALL PRIVILEGES, GRANT OPTION

FROM ***user*** [, ***user***] ...

第一种语句撤销指定用户的特定权限。对于第一种语法，当前用户必须拥有被撤销的权限，并且拥有GRANT OPTION权限。

第二种语句撤销指定用户的所有权限。对于这种语句，当前用户必须拥有全局GRANT OPTION权限，或者对权限表的UPDATE及DELETE权限。

注意，撤回操作**不会**级联。例如，用户user1给user2授予了某些权限，撤回user1的权限**不会**同时也撤回user2的相应权限。

## 查看权限

SHOW GRANTS [FOR ***user***]

如果不指定用户名，则缺省显示当前用户的权限。对于当前用户，总可以查看自己的权限。如果要查看其他指定用户的权限，必须拥有对权限表的SELECT权限。

# 用例

系统初始化后，只有一个管理员账户admin，初始密码admin。管理员用这组用户名密码登录系统。

1. 修改admin管理员的初始密码

SET PASSWORD ‘mypass’

1. 新建系统管理员账户并授予管理权限

CREATE USER ‘Li Lei’ IDENTIFIED BY ‘I love Han Meimei’

GRANT ALL PRIVILEGES, GRANT OPTION ON \* TO ‘Li Lei’

注意，管理员’Li Lei’拥有和admin相同的权限。

1. 新建应用管理员账户app1admin

CREATE USER ‘app1admin’ IDENTIFIED BY ‘app1pass’

GRANT ALL PRIVILEGES, GRANT OPTION ON t1 TO ‘app1admin’

1. app1admin新建普通用户账户

CREATE USER ‘app1user’ IDENTIFIED BY ‘app1user’

GRANT SELECT, INSERT, UPDATE, ALTER ON t1 TO ‘app1user’

1. 系统管理员轮值，对管理员账户’Li Lei’改名

RENAME USER ‘Li Lei’ TO ‘Han Meimei’

1. 撤销app1user账户部分权限

REVOKE ALTER ON t1 FROM ‘app1user’

SET PASSWORD FOR ‘app1user’ = ‘newpass’

1. app1的权限收归系统管理员，删除账户app1admin

DROP USER app1admin

目前设计的权限控制方案无法直接控制用户对于某个表格中某些列的权限。但可以利用将来支持的view来达到某种程度的控制。例如

1. 表格t1有c1, c2, c3, …, c8等列，用户user1对t1没有任何权限
2. 新建视图v1包含t1中c1, c2两列

CREATE VIEW v1 …

1. 授予user1对于v1的访问权限

GRANT SELECT, UPDATE ON v1 TO ‘user1’

这样，用户user1就间接拥有了对t1的列c1, c2的SELECT, UPDATE权限。

# 实现

权限控制的对象有两层：单个表格，或者全部表格（全局权限）。每个用户的全局权限直接存储在\_\_users表中。某个用户对于单个表格的权限存储在系统表\_\_table\_privileges中。

## 内部系统表

### \_\_users

|  |  |  |
| --- | --- | --- |
| 列名 | 类型 | 说明 |
| user | vchar | 用户名 |
| user\_id | int64 | 用户ID |
| password | vchar | 加密后的口令串 |
| comment | vchar | 说明 |
| privileges | vchar(32) | BitSet，每一位表示一种权限，代表全局权限，如果在这里设置了INSERT权限，那说明对所有的表都具有INSRET权限 |
| locked | int64 | 1表示被锁定，0表示未锁定 |

主键为user

### \_\_table\_privileges

|  |  |  |
| --- | --- | --- |
| 列名 | 类型 | 说明 |
| user\_id | int64 | 用户ID |
| table\_id | int64 | 表ID |
| privileges | vchar(32) | BitSet，每一位表示一种权限 |

主键为(user\_id, table\_id)

## 初始化

boot strap过程中，RS需要新建权限控制表，并对权限表内容进行初始化，初始化时各个表的内容如下。

### \_\_users

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| user | user\_id | password | comment | privileges | locked |
| admin | 1 | ‘admin’的密文 | administrator | 全1 | 0 |

### \_\_table\_privileges

空

### \_\_all\_sys\_stat

|  |  |  |
| --- | --- | --- |
| name | value1 | info |
| ob\_max\_user\_id | 1 | 当前最大用户ID |

## 算法流程

采用与目前Schema类似的架构实现权限管理：

* 在每个MS上有一个内存数据结构缓存当前整个系统的权限控制信息ObPrivileges。
* 所有MS向RS请求获得最新权限信息。RS在发给MS的心跳中带上当前权限信息的版本号。
* RS通过读取内部表来构造自己内存中缓存的ObPrivileges结构，并维护一个全局的版本号（时间戳）。
* 所有用户和权限管理语句都有MS解释后发送给RS执行。RS执行成功后，要立即通知所有Server更新权限控制信息。

## 认证流程

### 建立连接时

客户端与MS建立连接时，在login消息中带上用户名和加密后的密码，MS通过本地的ObPrivileges验证用户名和密码。如果用户不存在、或者密码错误，则拒绝连接，返回权限错误。如果认证成功，则把用户名和用户ID记录在当前Session中。

### 执行SQL语句时

执行任何SQL语句时，根据当前Session中记录的用户名和用户ID，检查该用户是否有权限执行本SQL语句，如果没有权限则拒绝执行，返回权限错误，如果有，则将权限相关的SQL语句转成对具体的对权限表的操作，然后再执行。

## 并发控制

MS接收到客户端的权限控制相关的SQL语句时，将SQL语句解析成一组SQL语句，组成一个读写事务，里面包含对相应的权限表的DML操作。交由sql engine去执行。

最终由UPS来保证事务。

问题：UPS目前暂时不支持DB语义的INSERT, 所以目前暂时规定只有数据库管理员能进行这种权限控制操作。