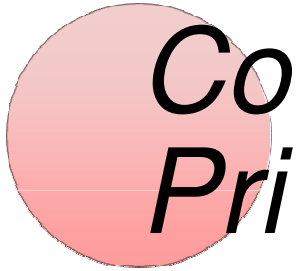
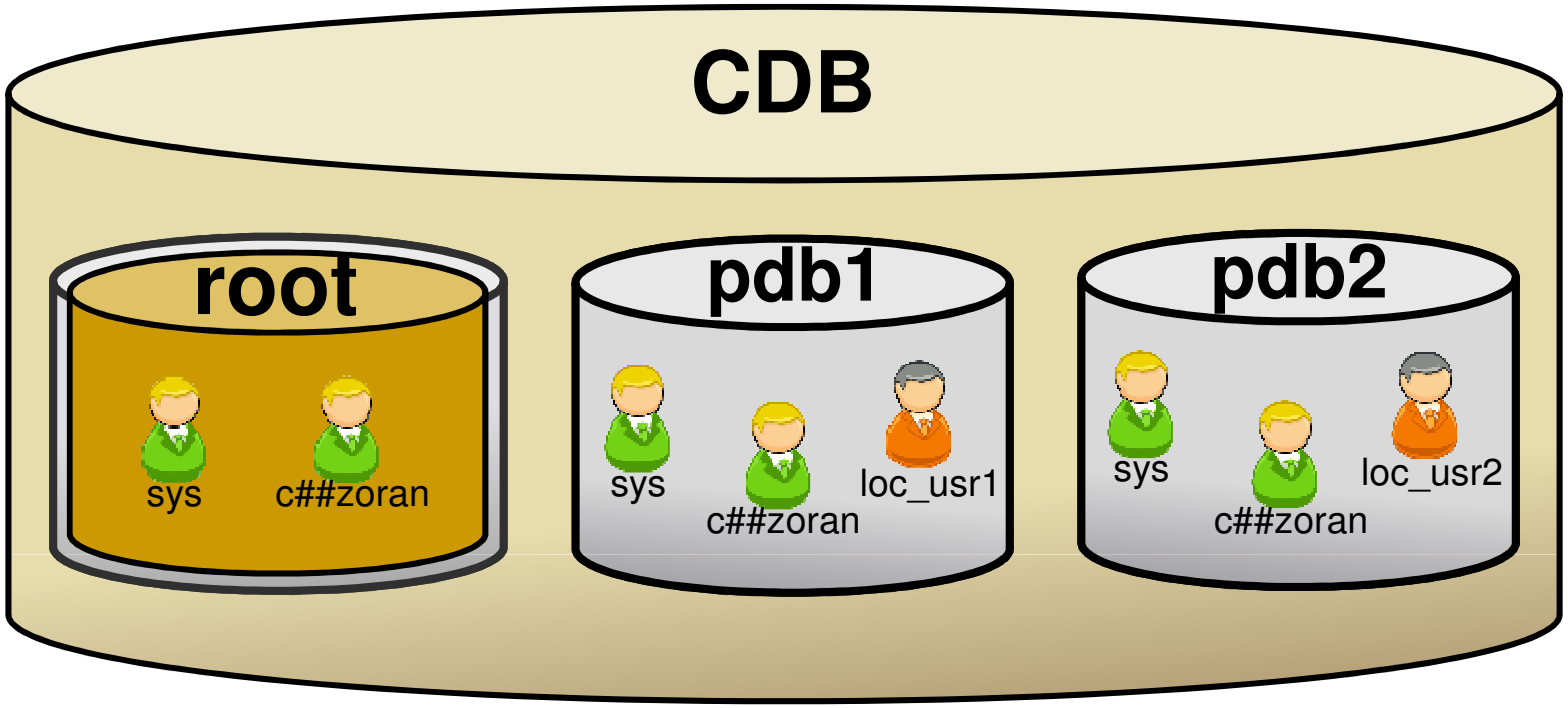
New Security Features in Oracle Database 12c

Zoran Pavlović, Security Team Lead, Parallel
Maja Veselica, Security Consultant, Parallel



Common and Local Users, Privileges and Roles

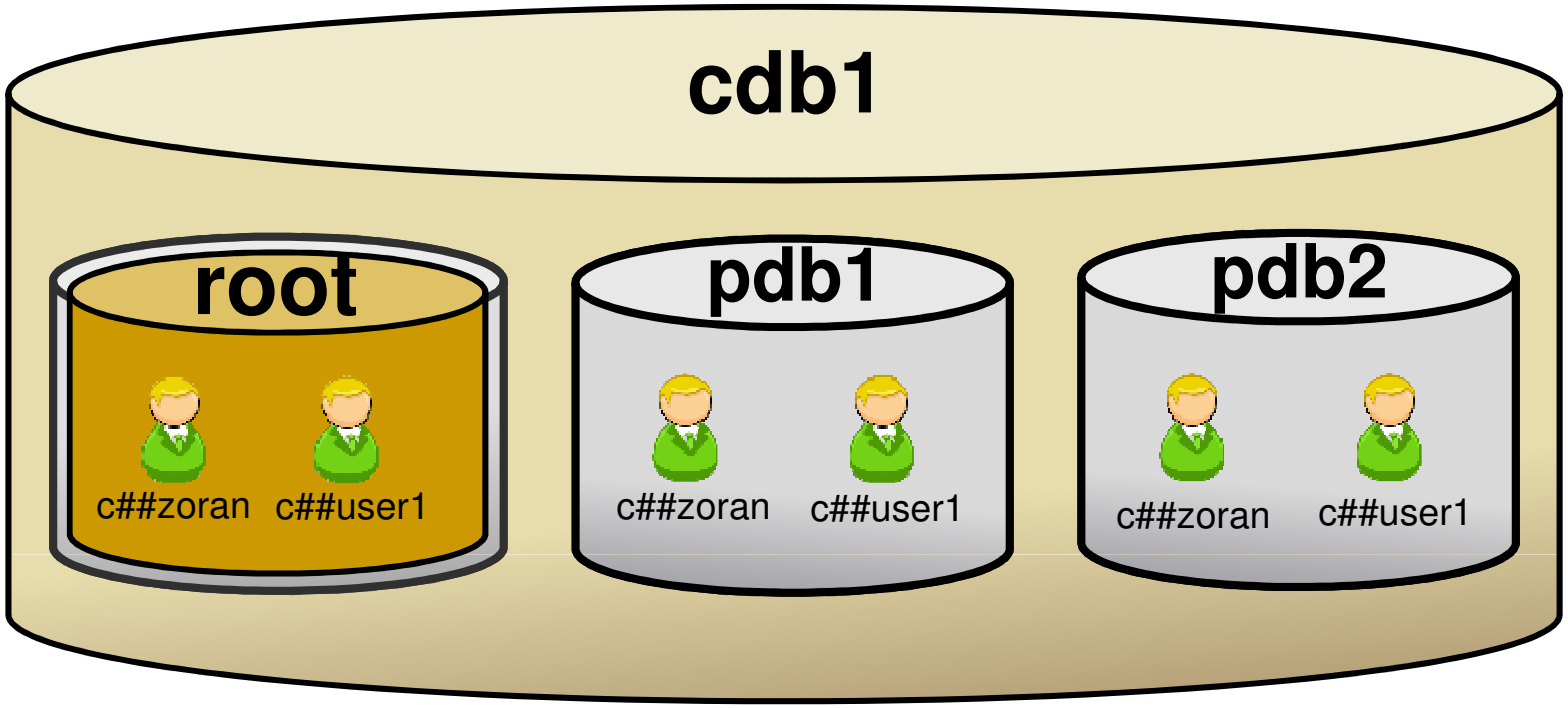
Common and Local Users



Common users are users created in root container, that have same identity across all containers.

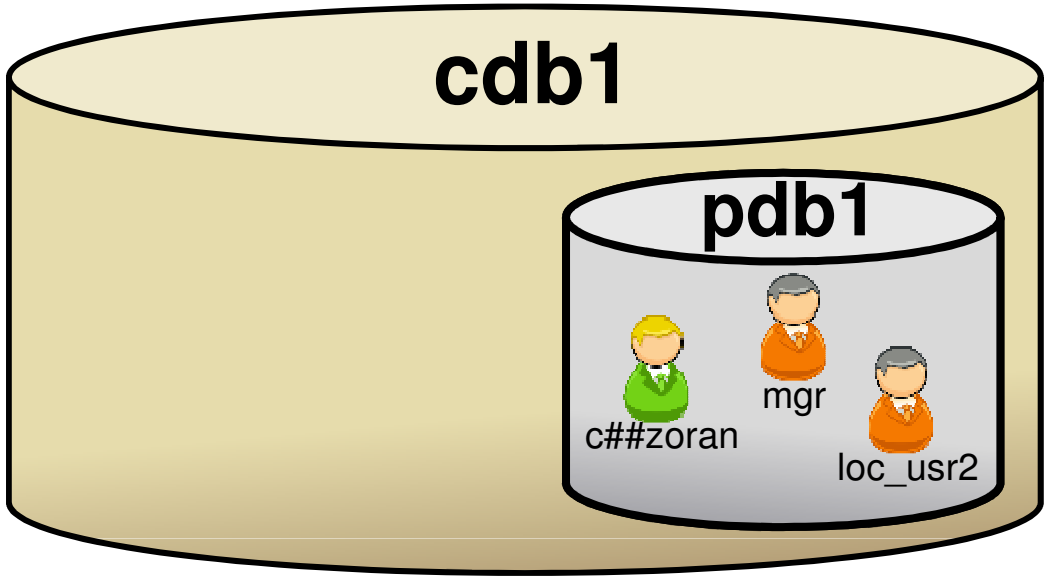
Local users are users that are created and exist in only one ***PDB***. They can't be created in root.

Creating Common Users




```
c##zoran@CDB1> CREATE USER c##user1 IDENTIFIED BY oracle1
CONTAINER = ALL;
```

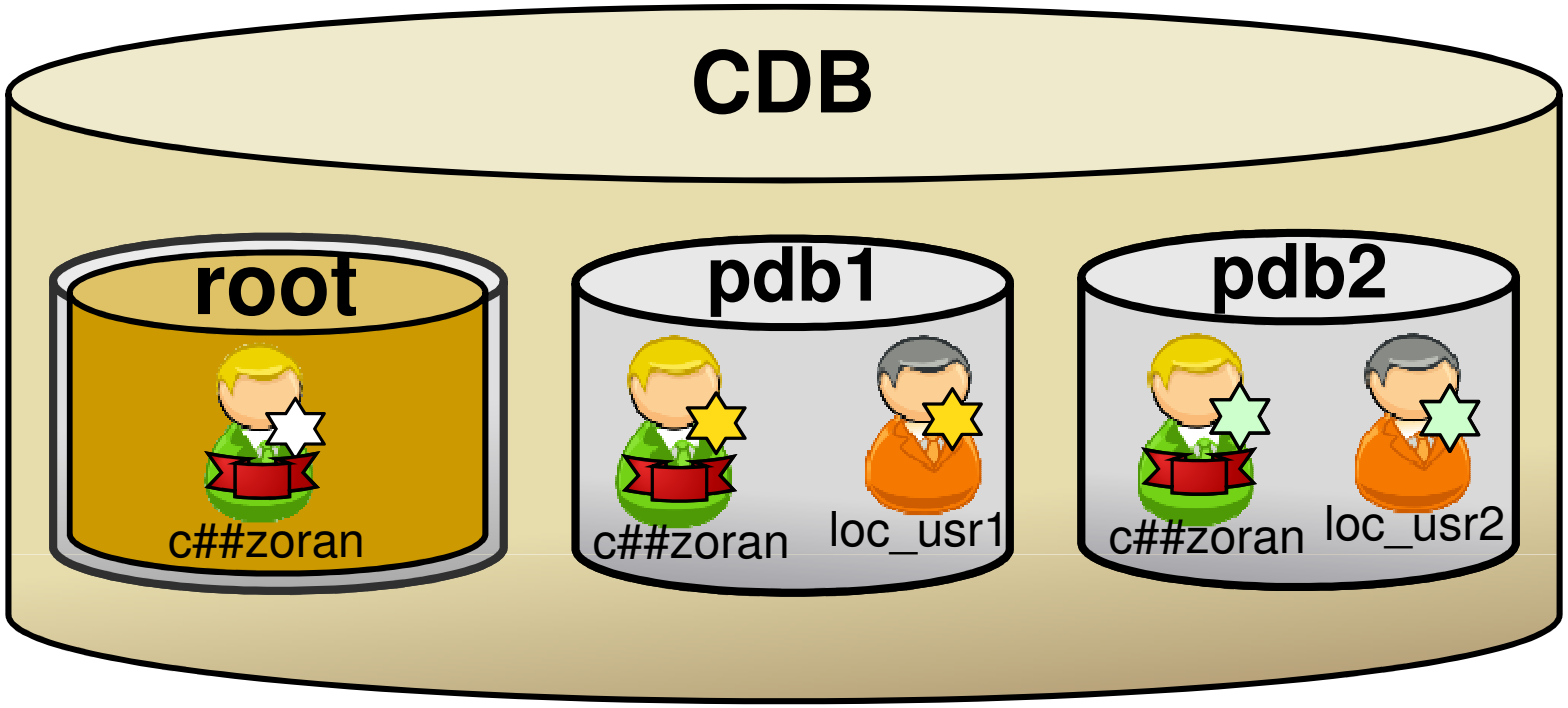
Creating Local Users




```
c##zoran@PDB1> CREATE USER mgr IDENTIFIED BY oracle1
CONTAINER = CURRENT;
```

```
mgr@PDB1> CREATE USER loc_usr1 IDENTIFIED BY password;
```

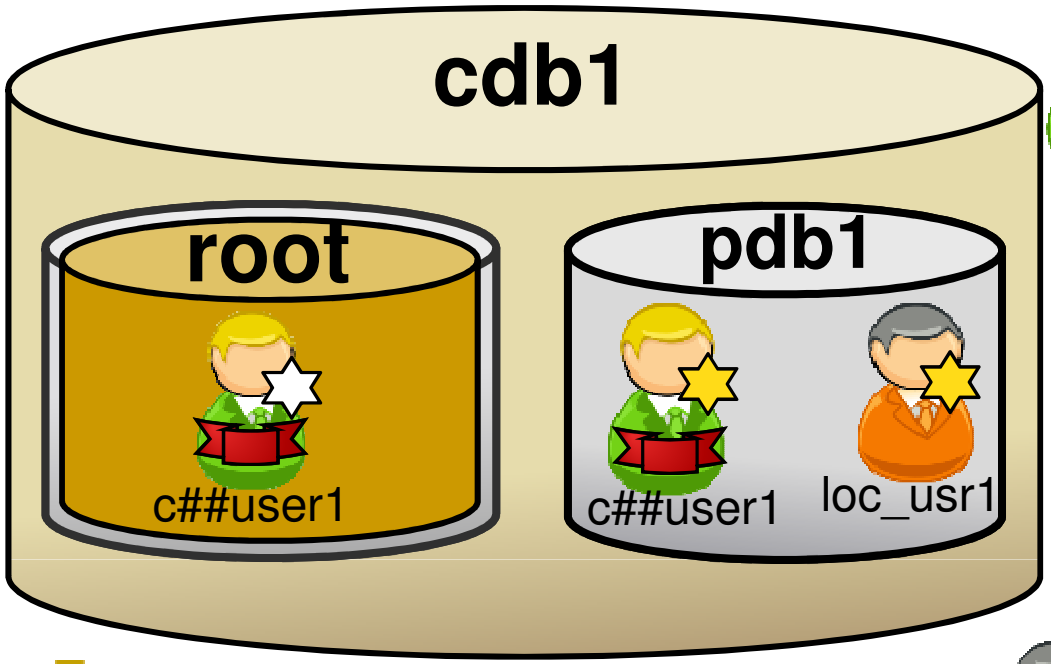
Common and Local Privileges



Common privileges are privileges, that when granted can be exercised across all containers.

Local privileges are privileges, that when granted can be exercised in context of a single ***PDB.***

Common and Local Privileges



Local privilege granted by common user to common user:

```
c##zoran@PDB1> GRANT
UPDATE ANY TABLE TO
c##user1
CONTAINER = CURRENT;
```

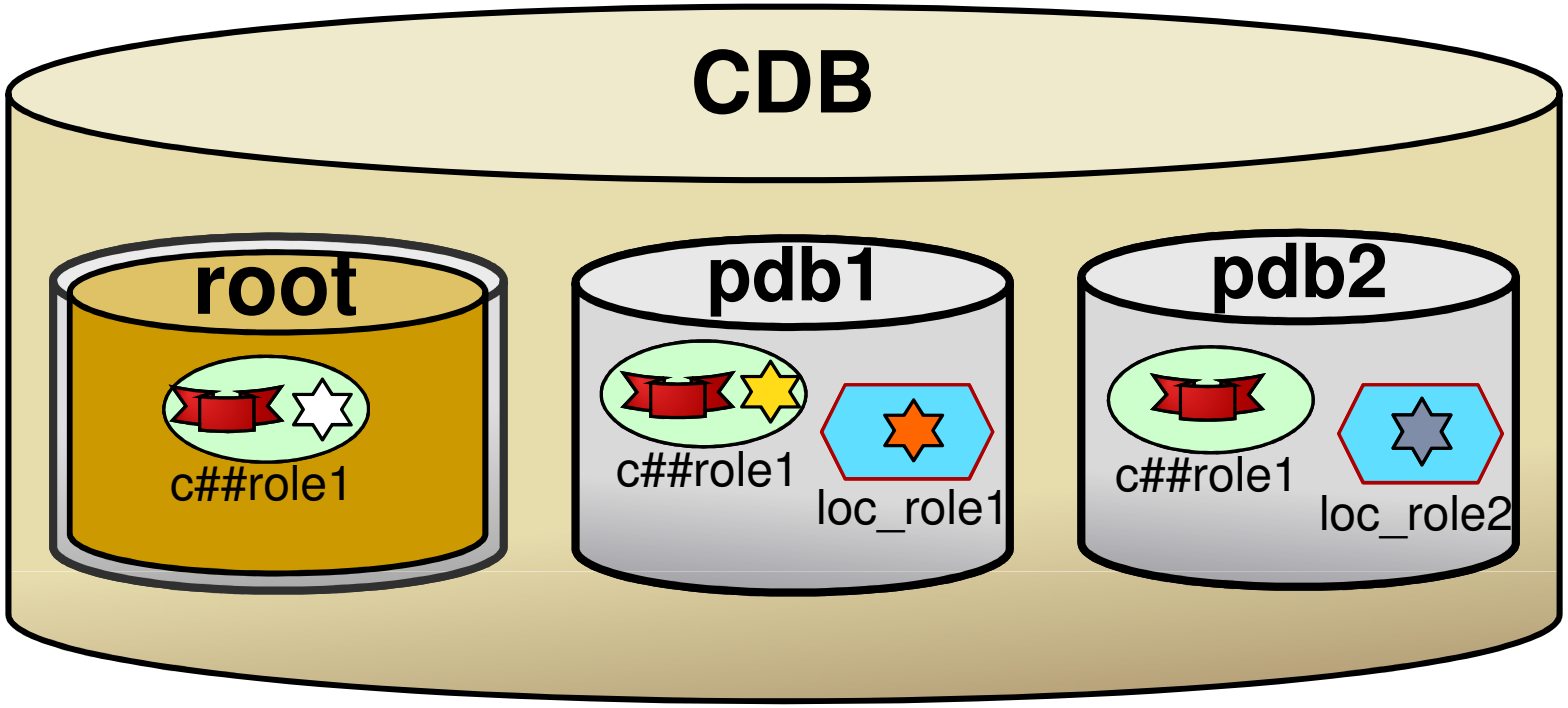
Common privilege granted by common user to common user:

```
c##zoran@CDB1> GRANT SELECT
ANY TABLE TO c##user1
CONTAINER = ALL;
```

Local privilege granted by local user to local user:

```
mgr@PDB1> GRANT UPDATE
ANY TABLE TO loc_usr1;
```

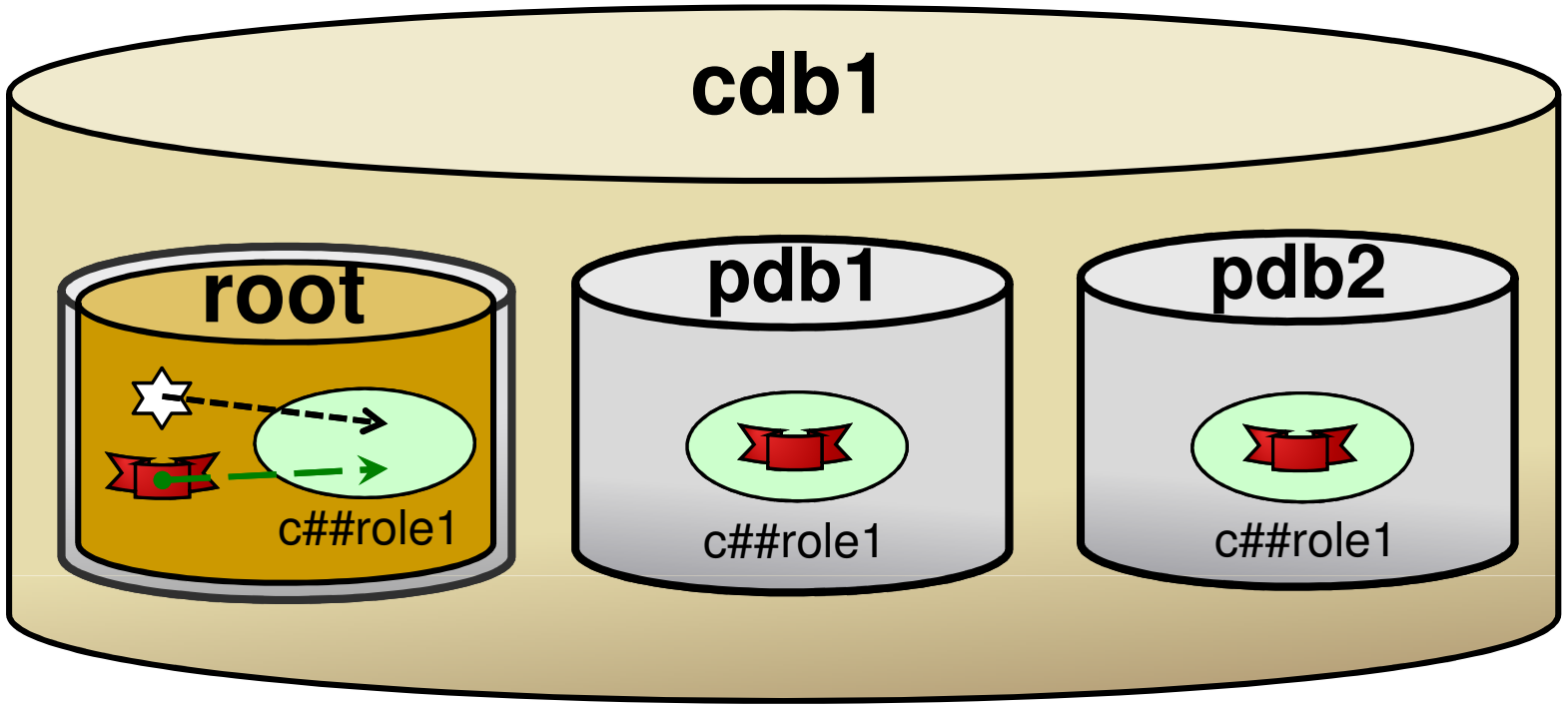
Common and Local Roles



Common roles are roles created in root container, that exist in all containers. These roles can have different set of privileges in different containers, and can be granted to either common or local users or roles.

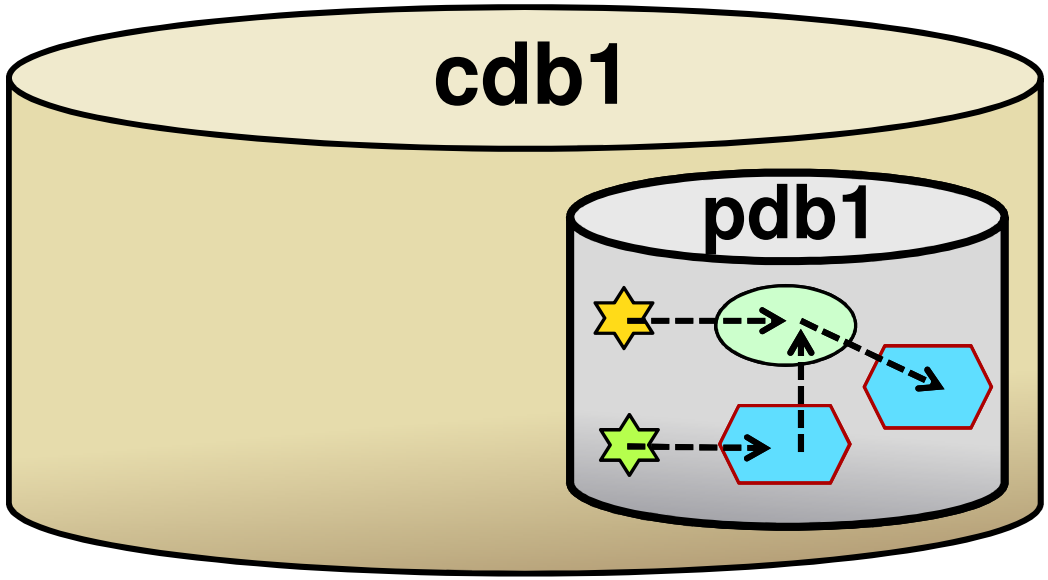
Local roles are roles created in PDB that exist in only one container. These roles can be granted only locally to either common or local users or roles.

Adding Privs to Common and Local Roles



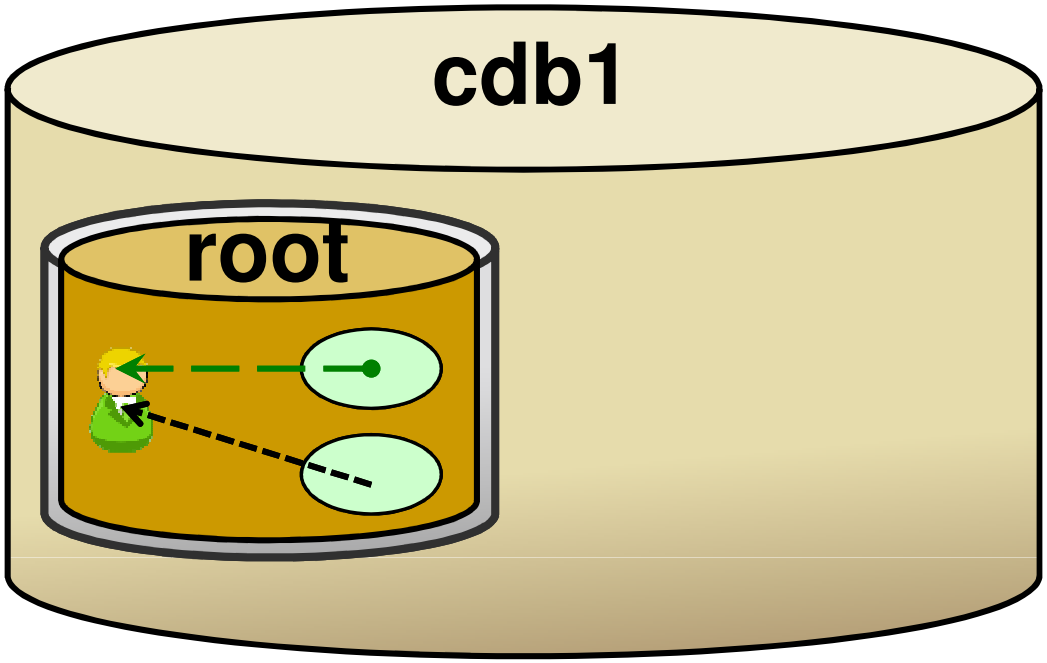

```
c##zoran@CDB1> CREATE ROLE c##role1 CONTAINER = ALL;
Role created.
c##zoran@CDB1> GRANT SELECT ANY TABLE TO c##role1 CONTAINER = ALL;
Grant succeeded.
c##zoran@CDB1> GRANT CREATE TABLE TO c##role1;
```

Adding Privs to Common and Local Roles



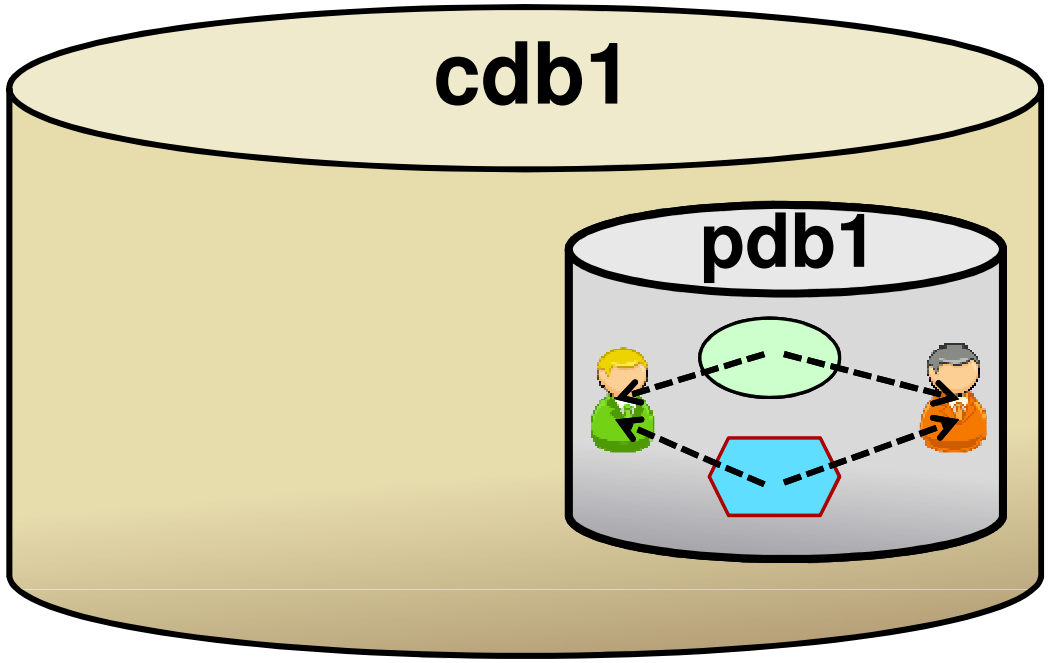

```
c##zoran@PDB1> CREATE ROLE loc_role1 CONTAINER = CURRENT;
Role created.
c##zoran@PDB1> GRANT UPDATE ANY TABLE TO loc_role1;
Grant succeeded.
c##zoran@PDB1> GRANT ALTER USER TO c##role1;
Grant succeeded.
c##zoran@PDB1> GRANT loc_role1 TO c##role1 CONTAINER = CURRENT;
Grant succeeded.
c##zoran@PDB1> GRANT c##role1 TO loc_role2 CONTAINER = CURRENT;
Grant succeeded.
```

Granting Common and Local Roles




```
c##zoran@CDB1> GRANT c##role1 to c##user1 CONTAINER = ALL;
Grant succeeded.
c##zoran@CDB1> GRANT c##role2 to c##user1 CONTAINER = CURRENT;
Grant succeeded.
```

Granting Common and Local Roles



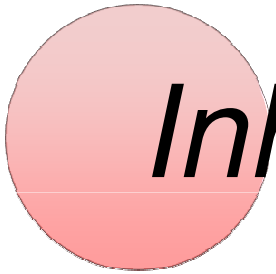

```
c##zoran@PDB1> GRANT c##role2 to c##user1 CONTAINER = CURRENT;
Grant succeeded.
c##zoran@PDB1> GRANT loc_role to c##user1 CONTAINER = CURRENT;
Grant succeeded.
c##zoran@PDB1> GRANT c##role2 to loc_usr1 CONTAINER = CURRENT;
Grant succeeded.
c##zoran@PDB1> GRANT loc_role to loc_usr1 CONTAINER = CURRENT;
Grant succeeded.
```

```
c##zoran@CDB1> create user c##usr identified by oracle1 container=all;
User created.
c##zoran@CDB1> grant create session, drop any synonym to c##usr
container=all;
Grant succeeded.
```

```
c##zoran@PDB1> grant drop any table to c##usr container=current;
Grant succeeded.
```

```
c##usr@PDB1> drop synonym customers_syn;
Synonym dropped.
c##usr@PDB1> drop table gldb.customers;
Table dropped.
```

```
c##usr@PDB2> drop synonym test_syn;
Synonym dropped.
c##usr@PDB2> drop table test.a;
drop table test.a
                *
ERROR at line 1:
ORA-00942: table or view does not exist
```



Inherit Privileges

```
ops$zoran@ORCL11GR2> create user maja identified by oracle1;
User created.
ops$zoran@ORCL11GR2> grant create session, create procedure to maja;
Grant succeeded.
ops$zoran@ORCL11GR2> connect maja/oracle1;
Connected.
maja@ORCL11GR2> select * from session_roles;
No rows selected.
maja@ORCL11GR2> create or replace procedure evil_proc
  2  authid current_user
  3  as
  4  begin
  5     execute immediate 'grant dba to maja';
  6 end;
  7 /
Procedure created.
maja@ORCL11GR2> grant execute on evil_proc to zoran;
Grant succeeded.
```

```
ops$zoran@ORCL11GR2> exec maja.evil_proc;
```

Inherit Privileges

```
ops$zoran@ORCL11GR2> exec maja.evil_proc;
PL/SQL procedure successfully completed.
ops$zoran@ORCL11GR2> connect maja/oracle1
Connected.
maja@ORCL11GR2> select * from session_roles;
ROLE
------------------------------
DBA
SELECT_CATALOG_ROLE
EXECUTE_CATALOG_ROLE
...
XDB_SET_INVOKER
OLAP_DBA
OLAP_XS_ADMIN

19 rows selected.
```

```
c##zoran@PDB1> create user maja identified by oracle1
container=current;
User created.
c##zoran@PDB1> grant create session, create procedure to maja;
Grant succeeded.
c##zoran@PDB1> connect maja/oracle1@PDB1;
Connected.
maja@PDB1> select * from session_roles;
No rows selected.
maja@PDB1> create or replace procedure evil_proc
  2  authid current_user
  3  as
  4  begin
  5     execute immediate 'grant dba to maja';
  6 end;
  7 /
Procedure created.
maja@PDB1> grant execute on evil_proc to zoran;
Grant succeeded.
```

```
c##zoran@PDB1> exec maja.evil_proc;
```

Inherit Privileges


```
c##zoran@PDB1> exec maja.evil_proc;
ERROR at line 1:
ORA-06598: insufficient INHERIT PRIVILEGES privilege
ORA-06512: at "MAJA.EVIL_PROC", line 1
ORA-06512: at line 1
c##zoran@PDB1> grant inherit privileges on user c##zoran to maja;
Grant succeeded.
c##zoran@PDB1> exec maja.evil_proc;
PL/SQL procedure successfully completed.
c##zoran@PDB1> connect maja/oracle1@pdb1
Connected.
maja@PDB1> select * from session_roles;
ROLE
------------------------------
DBA
SELECT_CATALOG_ROLE
...

19 rows selected.
```

New PL/SQL Privilege Checking

maja

```
SQL> create or replace procedure evil_proc
  2  authid current_user
  3  as
  4  begin
  5     execute immediate 'grant dba to maja';
  6 end;
  7 /
```

c##zoran

New PL/SQL Privilege Checking

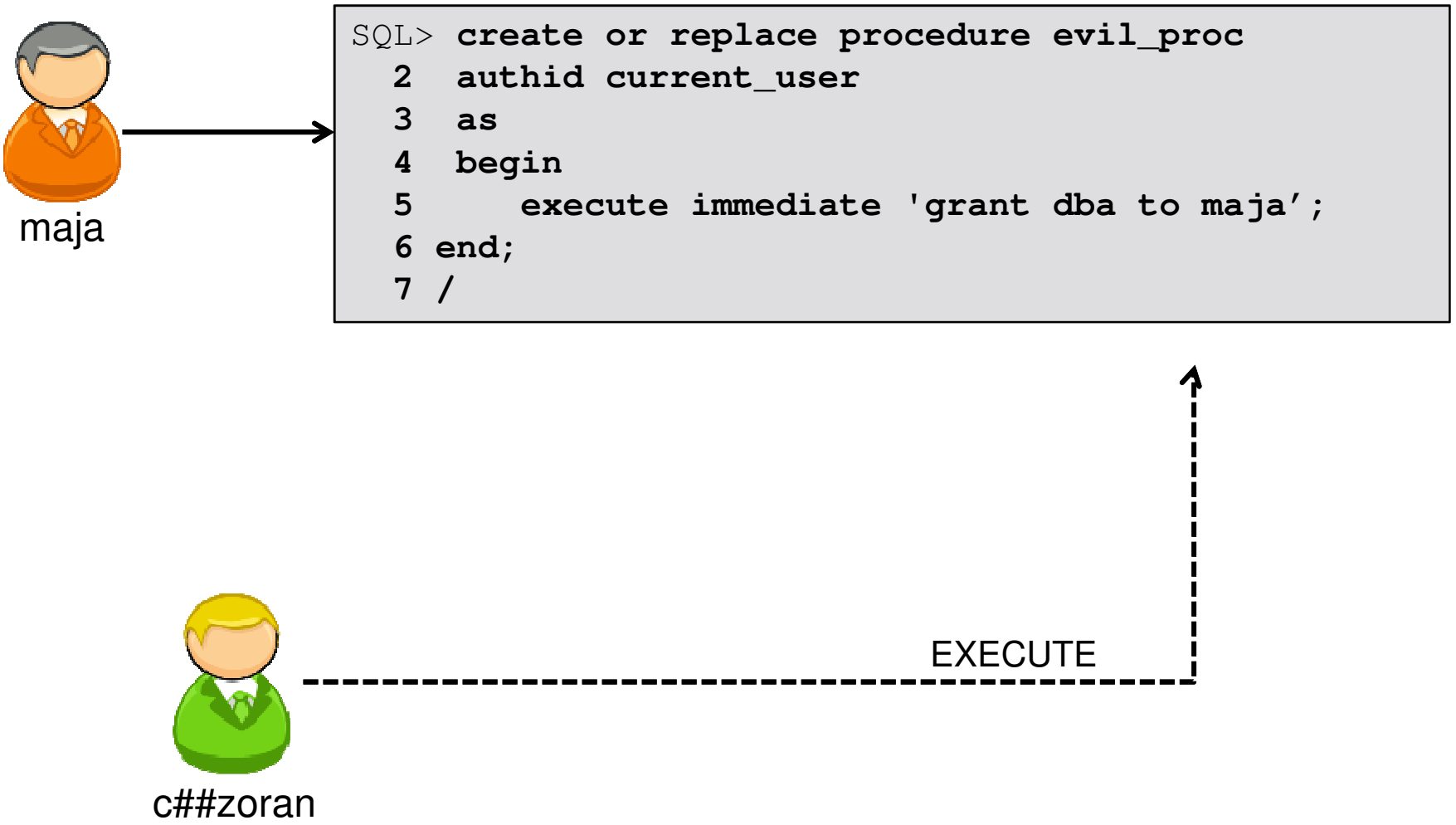
maja

```
SQL> create or replace procedure evil_proc
  2  authid current_user
  3  as
  4  begin
  5     execute immediate 'grant dba to maja';
  6 end;
  7 /
```

```
SQL> GRANT EXECUTE ON MAJA.EVIL_PROC TO
c##zoran;
```

c##zoran

New PL/SQL Privilege Checking



New PL/SQL Privilege Checking

maja

```
SQL> create or replace procedure evil_proc
  2  authid current_user
  3  as
  4  begin
  5     execute immediate 'grant dba to maja';
  6 end;
  7 /
```

EXECUTE

c##zoran

```
ERROR at line 1:
ORA-06598: insufficient INHERIT PRIVILEGES privilege
ORA-06512: at "maja.evil_proc", line 1
ORA-06512: at line 1
```

New PL/SQL Privilege Checking

maja

```
SQL> create or replace procedure evil_proc
  2  authid current_user
  3  as
  4  begin
  5     execute immediate 'grant dba to maja';
  6 end;
  7 /
```

```
SQL> GRANT INHERIT PRIVILEGES ON USER c##zoran TO
maja;
```

c##zoran

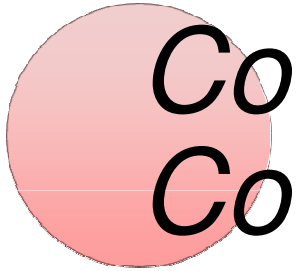
New PL/SQL Privilege Checking

maja

```
SQL> create or replace procedure evil_proc
  2  authid current_user
  3  as
  4  begin
  5     execute immediate 'grant dba to maja';
  6  end;
  7  /
```

EXECUTE

c##zoran



Code Based Access Control

Code Based Access Control


```
ops$zoran@ORCL11GR2> create user mike identified by oracle1;
User created.
ops$zoran@ORCL11GR2> create role proc_role;
Role created.
ops$zoran@ORCL11GR2> grant create session, create procedure, create
table to proc_role;
Grant succeeded.
ops$zoran@ORCL11GR2> grant proc_role to mike;
Grant succeeded.
```

```
mike@ORCL11GR2> create or replace procedure c_table
  2  as
  3  begin
  4  execute immediate 'create table test(a int)';
  5  end;
  6  /
Procedure created.
mike@ORCL11GR2> exec c_table;
```

Code Based Access Control

```
mike@ORCL11GR2> exec c_table;
BEGIN c_table; END;
*
ERROR at line 1:
ORA-01031: insufficient privileges
ORA-06512: at "MIKE.C_TABLE", line 4
ORA-06512: at line 1
```

Code Based Access Control

```
c##zoran@PDB1> create user mike identified by oracle1
container=current;
User created.
c##zoran@PDB1> create role proc_role container=current;
Role created.
c##zoran@PDB1> grant create session, create procedure, create table to
proc_role;
Grant succeeded.
c##zoran@PDB1> grant proc_role to mike;
Grant succeeded.
```

```
mike@PDB1> create or replace procedure c_table
  2  as
  3  begin
  4  execute immediate 'create table test(a int)';
  5  end;
  6  /
Procedure created.
mike@PDB1> exec c_table;
```

Code Based Access Control

```
mike@PDB1> exec c_table;
BEGIN c_table; END;
*
ERROR at line 1:
ORA-01031: insufficient privileges
ORA-06512: at "MIKE.C_TABLE", line 4
ORA-06512: at line 1
```

```
c##zoran@PDB1> grant proc_role to procedure c_table;
```

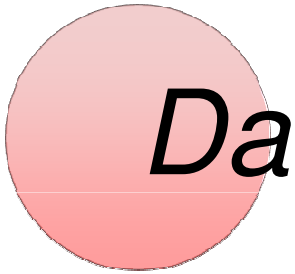
```
mike@PDB1> exec c_table;
PL/SQL procedure successfully completed.
mike@PDB1> desc test
Name                      Null?            Type
------------------------  --------         -------------
A                                          NUMBER(38)
```

Code Based Access Control


```
c##zoran@PDB1> create or replace procedure test
  2  authid current_user
  3  as
  4  begin
  5  execute immediate 'create table tjohn(z int)';
  6  end;
  7  /
Procedure created.
c##zoran@PDB1> create user john identified by oracle1
container=current;
User created.
c##zoran@PDB1> grant create session to john;
Grant succeeded.
c##zoran@PDB1> create role test_role container=current;
Role created.
c##zoran@PDB1> grant create table to test_role;
Grant succeeded.
c##zoran@PDB1> grant test_role to procedure test;
Grant succeeded.
c##zoran@PDB1> grant execute on test to john;
Grant succeeded.
```

Code Based Access Control

```
john@PDB1> exec c##zoran.test;
PL/SQL procedure successfully completed.

john@PDB1> desc tjohn
Name                      Null?            Type
------------------------  --------         -------------
Z                                          NUMBER(38)
```



Data Redaction

Data Redaction - Full

```
SQL> SELECT * FROM CUSTOMERS;
```

CUSTOMERS

NAME	CREDIT_CARD
tom	3455647456589132
mike	3734982321225691
john	3472586894975806

NAME	CREDIT_CARD
tom	0
mike	0
john	0

```
DBMS_REDACT.ADD_POLICY
(object_schema => 'GLDB',
object_name => 'CUSTOMERS',
policy_name => 'CCN_POLICY',
column_name => 'CREDIT_CARD',
function_type => DBMS_REDACT.FULL,
expression => '7=7');
```

Data Redaction - Partial

```
SQL> SELECT * FROM CUSTOMERS;
```

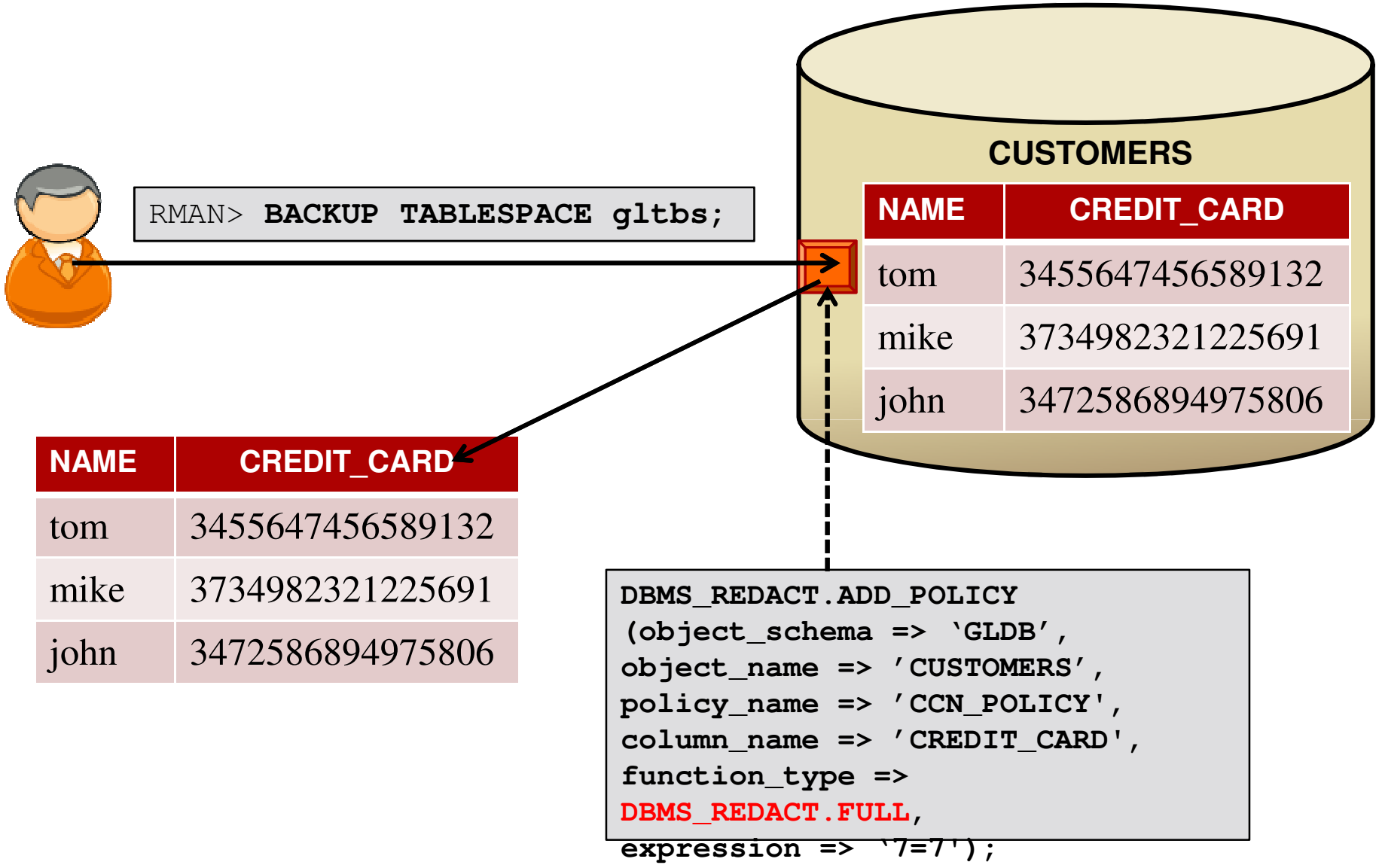
CUSTOMERS

NAME	CREDIT_CARD
tom	3455647456589132
mike	3734982321225691
john	3472586894975806

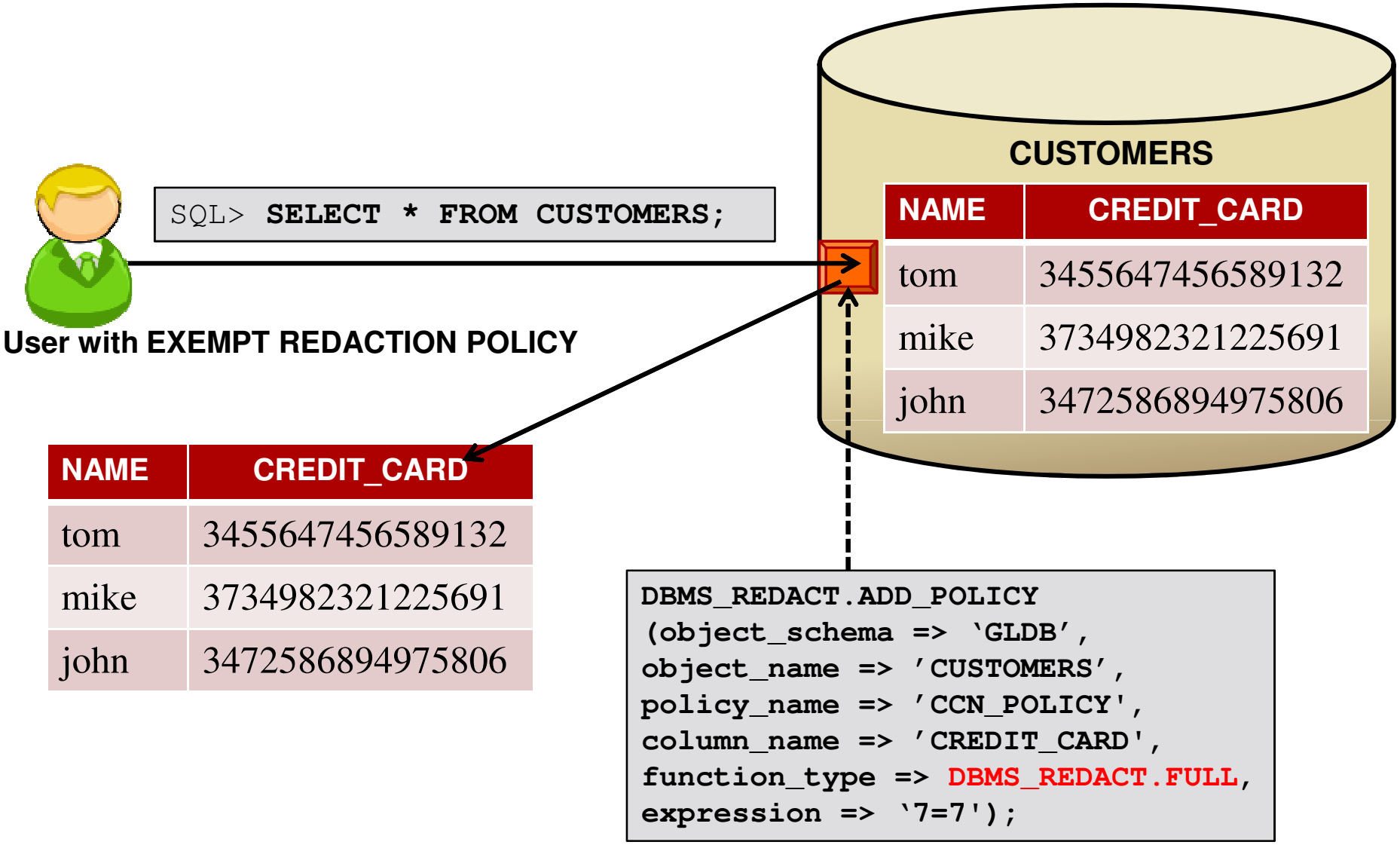
NAME	CREDIT_CARD
tom	####-####-####-9132
mike	####-####-####-5691
john	####-####-####-5806

```
DBMS_REDACT.ADD_POLICY
(object_schema => 'GLDB',
object_name => 'CUSTOMERS',
policy_name => 'CCN_POLICY',
column_name => 'CREDIT_CARD',
function_type => DBMS_REDACT.PARTIAL,
function_parameters => 'VVVVVVVVVVVVVVV,
VVVV-VVVV-VVVV-VVVV, #,1,12'
expression => '7=7');
```

Data Redaction - Exemptions



Data Redaction - Exemptions



Data Redaction - Example

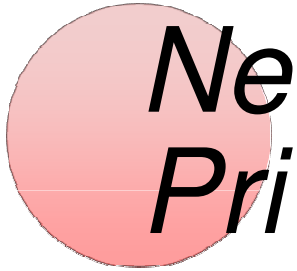
```
c##zoran@PDB1> BEGIN
  2  DBMS_REDACT.ADD_POLICY (object_schema => 'GLDB',
  3  object_name => 'CUSTOMERS',
  4  policy_name => 'CCN_POLICY',
  5  column_name => 'CREDIT_CARD',
  6  function_type => DBMS_REDACT.PARTIAL,
  7  function_parameters => 'VVVVVVVVVVVVVVV, VVVV-VVVV-VVVV-VVVV,
#,1,12'
  8  expression => '7=7');
  9  END;
  10 /
PL/SQL procedure successfully completed.

c##zoran@PDB1> select * from gldb.customers;

NAME                 CREDIT_CARD
---------------      --------------------
tom                  3455647456589132
mike                 3734982321225691
john                 3472586894975806

c##zoran@PDB1> grant select on gldb.customers to maja;
Grant succeeded.
```

Data Redaction - Example

```
maja@PDB1> select * from gldb.customers;

NAME                 CREDIT_CARD
---------------      --------------------
tom                  ####-####-####-9132
mike                 ####-####-####-5691
john                 ####-####-####-5806

maja@PDB1> select * from gldb.customers where credit_card like '3472%';

NAME                 CREDIT_CARD
---------------      --------------------
john                 ####-####-####-5806
```

Available Redaction Types

None	Full	Partial	Regular Expression	Random
• Redaction is NOT applied	• Columns are redacted to **constant values** depending on column data type	• User-specified **positions** are replaced by a user-specified **character**	• Pattern for matching and replacing is defined and used for reduction	• Preserves data types • Randomizes output



New Administrative Privileges

New Administrative Privileges

PRIVILEGE	USERNAME	DUTIES
SYSBACKUP	SYSBACKUP	Backup and recovery operations in RMAN and SQL.
SYSDG	SYSDG	Managing Data Guard with Data Guard Broker.
SYSKM	SYSKM	Managing keys for TDE.

Password file format_12c

New SYSBACKUP Privilege

```
SQL> connect / as SYSBACKUP
Connected.
SQL> show user
USER is "SYSBACKUP"
SQL> select * from session_privs;
PRIVILEGE
----------------------------------------
SYSBACKUP
SELECT ANY TRANSACTION
SELECT ANY DICTIONARY
RESUMABLE
CREATE ANY DIRECTORY
ALTER DATABASE
AUDIT ANY
CREATE ANY CLUSTER
CREATE ANY TABLE
UNLIMITED TABLESPACE
DROP TABLESPACE
ALTER TABLESPACE
ALTER SESSION
ALTER SYSTEM
14 rows selected.
SQL>
```

New SYSBACKUP Privilege

```
$ rman target '"zoran/passwd@orcldb AS SYSBACKUP"'
Recovery Manager: Release 12.1.0.1.0 - Beta on Tue
May 07 17:41:37 2013
Copyright (c) 1982, 2012, Oracle and/or its
affiliates. All rights reserved. connected to
target database: ORCLDB (DBID=1625181741)
RMAN> select user from dual;
using target database control file instead of
recovery catalog

USER
------------------------------
SYSBACKUP
RMAN>
```

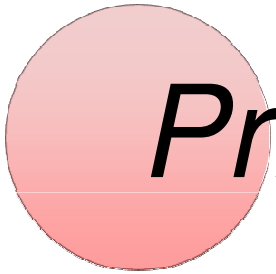
```
SQL> connect / as SYSBACKUP
Connected.
SQL> SELECT TABLE_NAME FROM DBA_TABLES
  2 WHERE OWNER = 'GLDB';

TABLE_NAME
----------------------------------------
CUSTOMERS
ORDERS

SQL> SELECT * FROM GLDB.CUSTOMERS;
SELECT * FROM GLDB.CUSTOMERS
                  *
ERROR at line 1:
ORA-01031: insufficient privileges
```

New SYSDG Privilege

```
SQL> connect / as SYSDG
Connected.
SQL> show user
USER is “SYSDG"
SQL> select * from session_privs;
PRIVILEGE
----------------------------------------
SYSDG
ALTER SYSTEM
ALTER SESSION
ALTER DATABASE
SELECT ANY DICTIONARY
5 rows selected.
SQL>
```

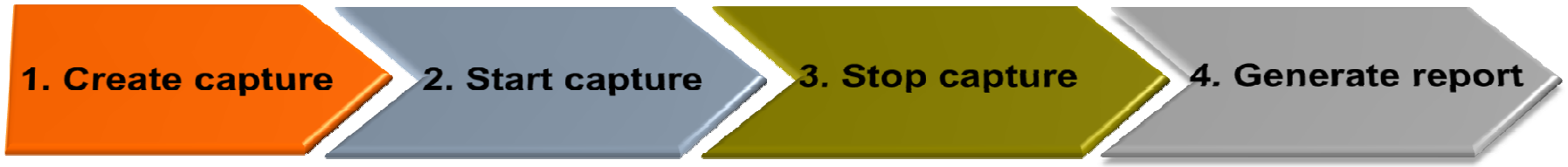
New SYSKM Privilege

```
SQL> connect / as SYSKM
Connected.
SQL> show user
USER is "SYSKM"
SQL> select * from session_privs;
PRIVILEGE
----------------------------------------
SYSKM
ADMINISTER KEY MANAGEMENT
2 rows selected.
SQL>
```

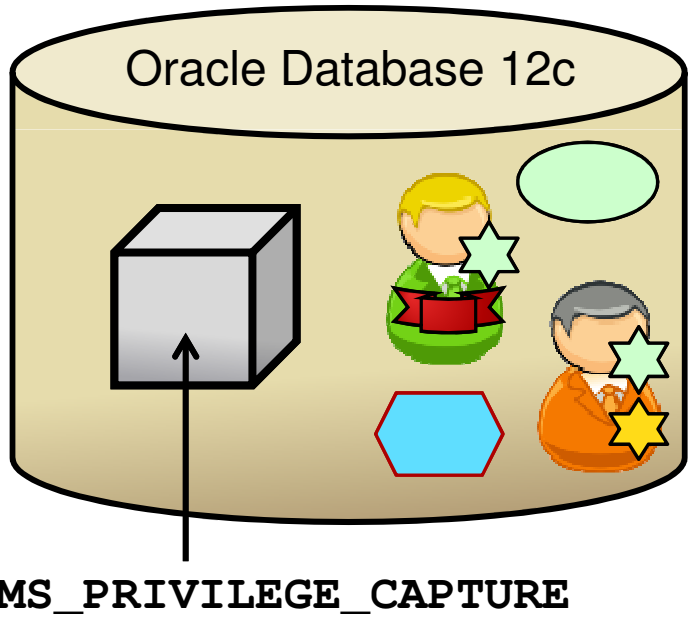


Privilege Analysis

Privilege Analysis

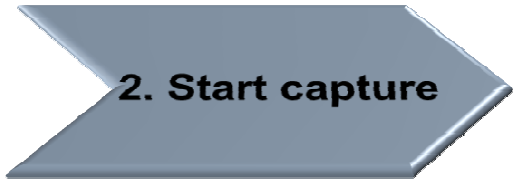


Privilege analysis is process of capturing and analyzing used privileges in order to revoke unused privileges.



Privilege Analysis


```
SYS.DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE
(name => 'AllPrivilleges',
description => 'All used privilleges',
type =>
dbms_privilege_captu .g_database)
```




```
SYS.DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE
(name => 'AllPrivilleges')
```


```
SYS.DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTUR
E
(name => 'AllPrivill es')
```


```
SYS.DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT
(name => 'AllPrivilleges')
```

Privilege Analysis Report

```
SQL> SELECT username, object_owner, object_name, obj_priv
 2 FROM DBA_USED_OBJPRIVS
 3 WHERE username IN ('ZORAN', 'MAJA');

USERNAME   OBJECT_OWNER OBJECT_NAME                    OBJ_PRIV
---------- ------------ ------------------------------ -------
ZORAN      SYS          DUAL                           SELECT
ZORAN      SYSTEM       PRODUCT_PRIVS                  SELECT
MAJA       SYS          ORA$BASE                       USE
MAJA       SYSTEM       PRODUCT_PRIVS                  SELECT
ZORAN      SYS          ORA$BASE                       USE
ZORAN      SYS          DBMS_APPLICATION_INFO          EXECUTE
MAJA       SYS          DUAL                           SELECT
MAJA       GLDB         CUSTOMERS                      SELECT
ZORAN      GLDB         CUSTOMERS                      SELECT
ZORAN      GLDB         ORDERS                         INSERT
MAJA       GLDB         ORDERS                         SELECT
ZORAN      SYS          DUAL                           SELECT
MAJA       SYS          DMBS_APPLICATION_INFO          EXECUTE
MAJA       SYS          DUAL                           SELECT
```

Privilege Analysis Report

```
SQL> SELECT USERNAME, SYS_PRIV
 2 FROM DBA_USED_SYSPRIVS
 3 WHERE username IN ('ZORAN', 'MAJA');

USERNAME     SYS_PRIV
------------ --------------------
ZORAN        CREATE SESSION
MAJA         CREATE SESSION
```

Privilege Analysis Report

```
SQL> SELECT USERNAME, OBJ_PRIV, OBJECT_NAME, PATH
  2 FROM  DBA_UNUSED_PRIVS
  3 WHERE  USERNAME IN (`ZORAN', `MAJA');

USERNAME   OBJ_PRIV   OBJECT_NAME    PATH
--------   --------   -----------    ----------------------------------
ZORAN      INSERT     CUSTOMERS      GRANT_PATH('ZORAN', 'GLDB_ROLE')
ZORAN      UPDATE     CUSTOMERS      GRANT_PATH('ZORAN', 'GLDB_ROLE')
MAJA       INSERT     CUSTOMERS      GRANT_PATH('MAJA', 'GLDB_ROLE')
ZORAN      UPDATE     ORDERS         GRANT_PATH('ZORAN', 'GLDB_ROLE')
MAJA       INSERT     CUSTOMERS      GRANT_PATH('MAJA', 'GLDB_ROLE')
MAJA       INSERT     ORDERS         GRANT_PATH('MAJA', 'GLDB_ROLE')
```

```
SQL> SELECT USERNAME, SYS_PRIV
  2 FROM  DBA_UNUSED_PRIVS
  3 WHERE  USERNAME IN (`ZORAN', `MAJA');

USERNAME   SYS_PRIV
--------   ----------------
ZORAN      SELECT ANY TABLE
ZORAN      UPDATE ANY TABLE
MAJA       SELECT ANY TABLE
```

Contact

Zoran Pavlović, *Security Team Lead*
zoran.pavlovic@parallel.rs

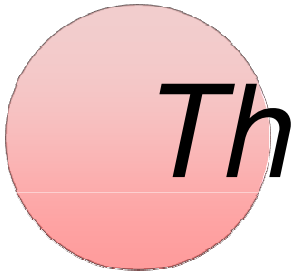
Twitter: @ChallengeZoran

Maja Veselica, *Security Consultant*
maja.veselica@parallel.rs

Twitter: @orapassion

Our Websites

www.optimasec.com - blog
www.challengezoran.com - ask your questions
www.parallel.rs - Company



Thank you!