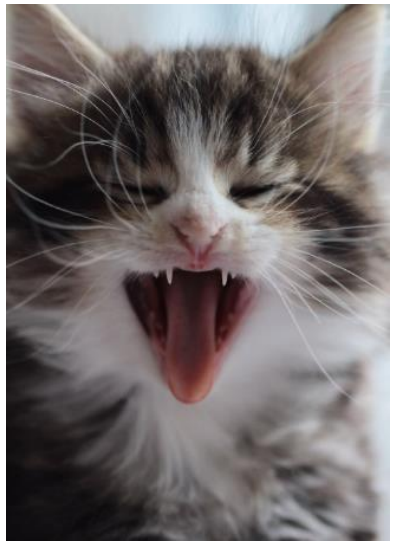
DOAG

Deutsche ORACLE-Anwendergruppe e.V.



DOAG Konferenz 20.11.2018

Die Anwender in der DB im Zaum halten

ÜBER RECHTE/ROLLEN UND DEN SICHEREN BETRIEB DER DATENBANK

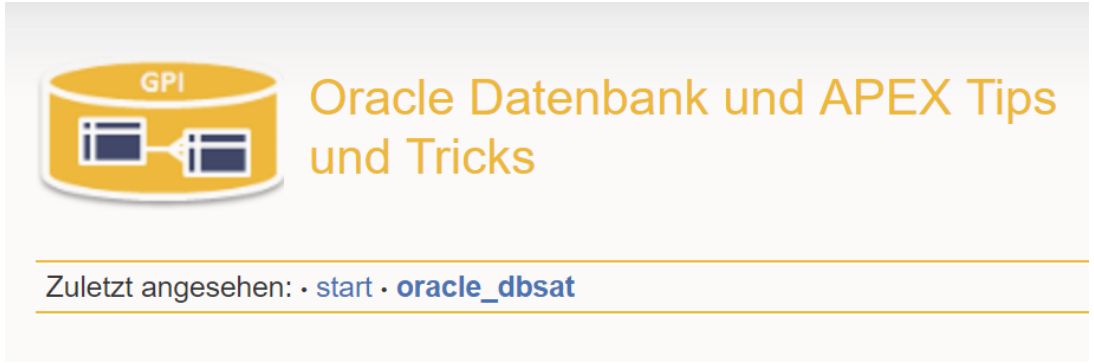
GPI Consult Gunther Pippèrr

gunther@pipperr.de

Bergweg 14
37216 Witzenhausen/Roßbach

Mein Blog

https://www.pipperr.de/dokuwiki/



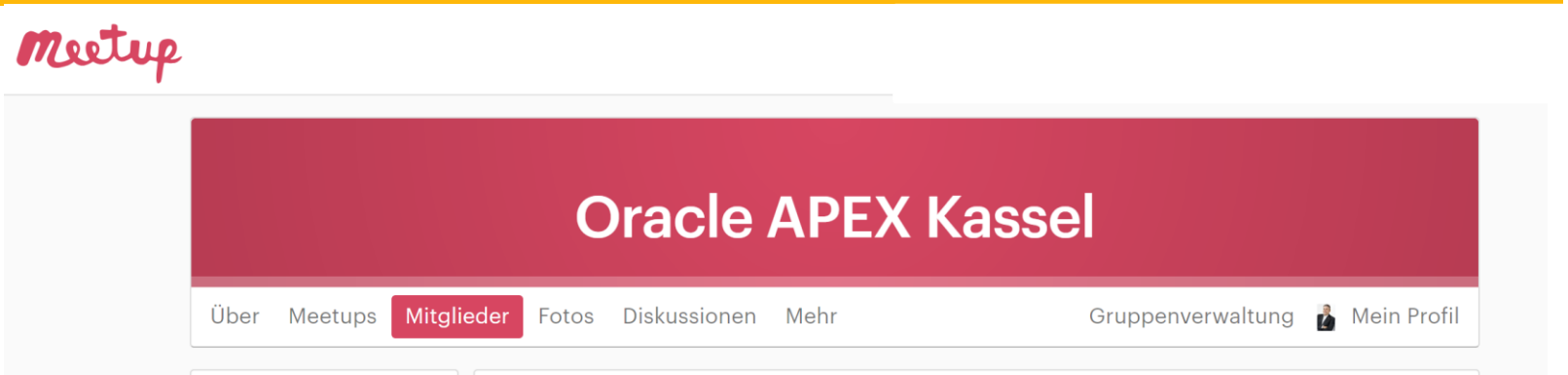
Freiberuflicher Oracle Datenbank Experte - Ich unterstütze Sie gerne in ihren Projekten.

APEX Meetup Gruppe Kassel

Raum für Veranstaltungen in Kassel gesucht, können Sie unterstützen?

Mitglieder gesucht!

https://www.meetup.com/de-DE/Oracle-APEX-Kassel/



Die Anwender in der DB im Zaum halten

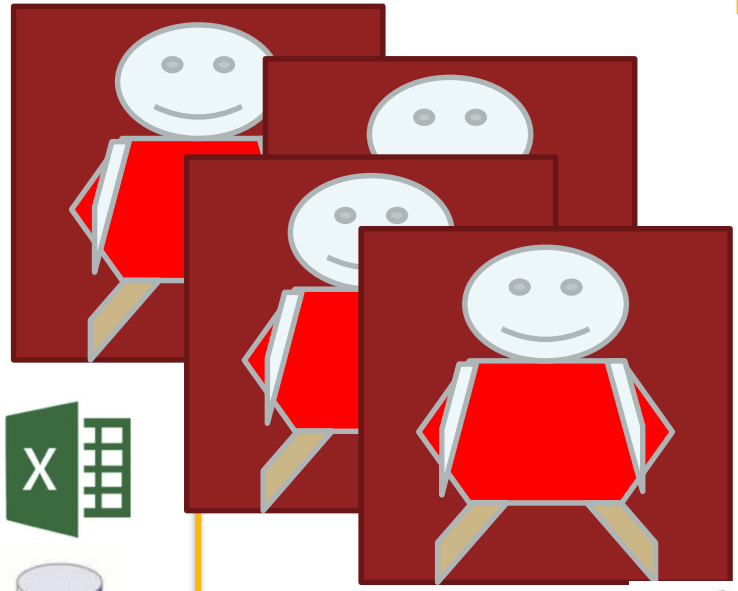
ÜBER RECHTE/ROLLEN UND DEN SICHEREN BETRIEB DER DB

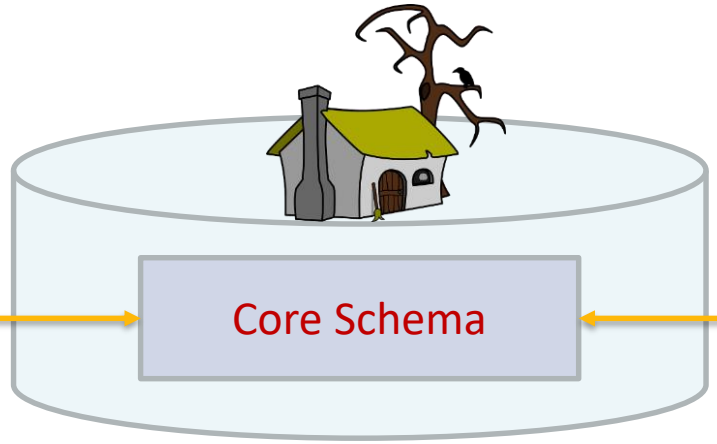
Der normale Alltag – Die Ausgangslage

Applikation

Im normalen Fall => Schema Name = PASSWORT

- **Passwort des zentralen Schemas allgemein bekannt**
- **Änderungen auf Grund von vielen Abgängigkeiten**
- **nicht möglich**





Oracle 10g R2 ohne Patch

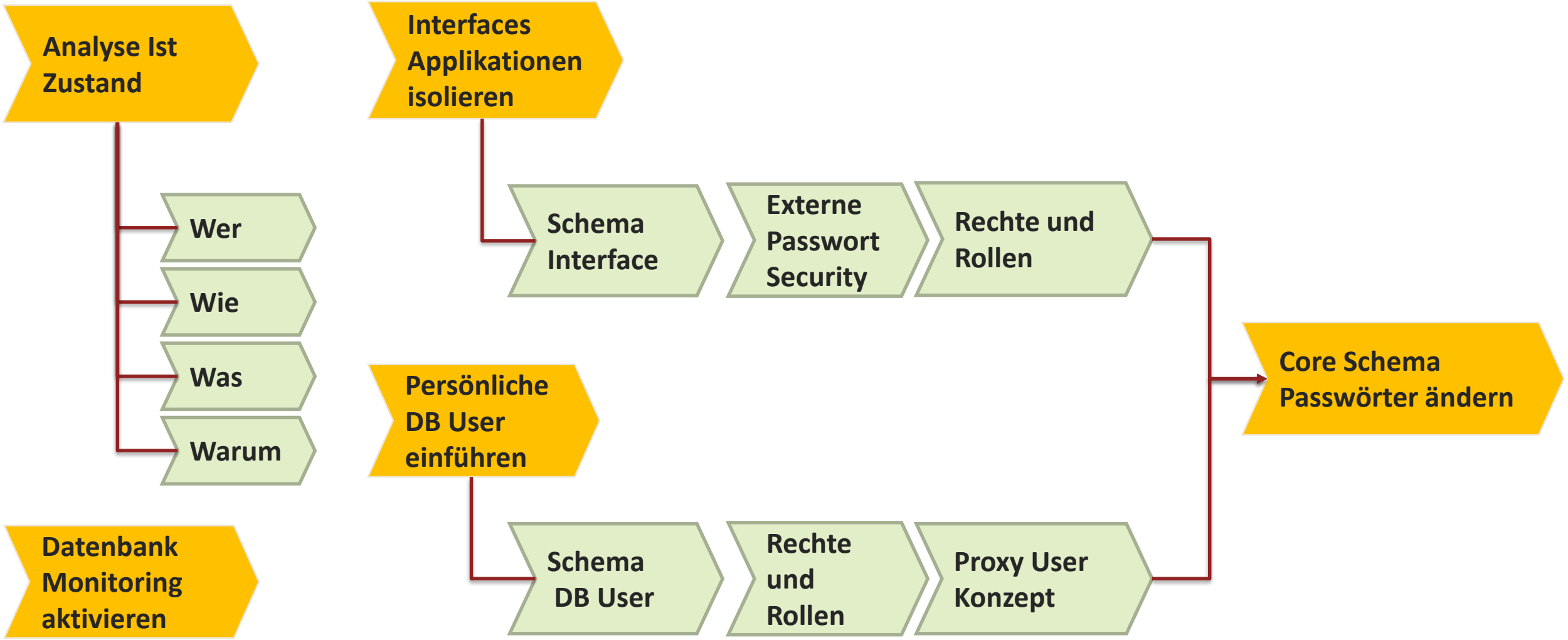
Wer hat wann und was in der Datenbank “angepasst”? Im Zweifelsfall war es mal wieder keiner ….

Unser Ziel

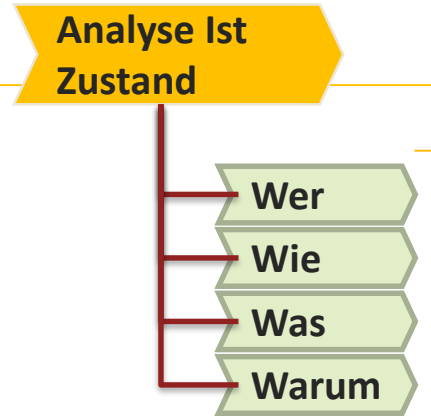
- Einzelne Anwender in der Datenbank identifizieren
- Zugriff auf die Daten auf so wenig Personen wie möglich/nötig einschränken
- Passwörter nicht in der ganzen Firma verteilen
- Passwörter entsprechend der Firmen Regularien verwenden
- Die DSGVO nachvollziehbar einhalten

- Das Arbeiten mit der Datenbank für Administratoren und Entwickler/Support so einfach wie möglich gestalten

Projekt zur Optimierung der DB aufsetzen



1- Analyse



- Analyse der Systemumgebung
  - Wer verwendet die Datenbank?
    - Warum werden Daten aus der Datenbank ausgelesen?
  - Was verwendet die Datenbank?
    - Welche Applikationen greifen auf die Datenbank zu
  - Werkzeug:
    - v$session in Hilfstabelle schreiben
    - Audit Log auswerten,
    - AWR Tabellen (falls Lizenz vorhanden) auswerten
    - Listener Log analysieren
    - DDL Trigger / Error Trigger

1 – v$session in Tabelle schreiben

- V$SESSION in eine einfache Hilfstabelle schreiben
  - Vorteil: ALLE Sessions, auch die nicht aktiven, werden protokolliert
  - Das Audit Log der DB ist meist nicht über Monate hinweg verfügbar
  - Keine Diagnostic Pack Lizenz notwendig für v$active_session_history!
  - Einfach zu implementieren und auszuwerten
  - Code Fragment:

```
create table mon_user.log_v_sessions as select s.*, '1' as snap,
sysdate as  snap_date from v$session s where username is not
null;

-- Job mit:
Insert into mon_user.log_v_sessions  select s.*,
snap_seq.nextval, sysdate from v$session s;
```

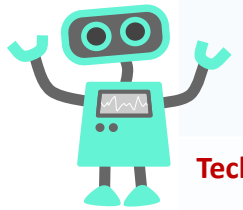
  - Nachteil : Rechte müssen beim DBA erkämpft werden

1 – Die Datenbank proaktiv überwachen

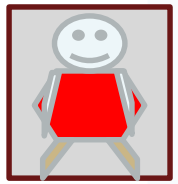
- Oracle listener.log regelmäßig auswerten
- DDL Logging einschalten
  - $ADR_HOME/rdbms/<sid>/<sid>/log/ddl/log.xml
  - DB Parameter ENABLE_DDL_LOGGING = true (EE Feature!)
- Standard AUDIT (pre 12c default) optimieren bzw. 12c unified auditing verwenden
- System Trigger für das Logging einsetzen
  - Wie On Logon, on DDL , on error

1 – Technische und persönliche User gruppieren

- Eine Matrix **was** und **wer** in der Datenbank **wie** arbeiten soll



User Type	Data Type	Example	Concept	Role Name	DB Profile Name
Technical	Data Owner	Application Core Schema	Very Strong Password with min. 20 letters		data_owner
Technical	Application (Select/DML)	Interfaces to other programs	Synonym Schema, minimal rights on table and PL/SQL	<appname>_role	interface
Personal	Only Select	Analysts	Synonym Schema, minimal rights on table and PL/SQL	For example Analyst_role_<schema>	personal_user
Personal	Select / DML /DDL	Developer and Deployment User	**Oracle Proxy Authentication**	Personal user only create Session	personal_user



Technische User

- Es wird zwischen zwei Gruppen unterschieden:
  - Technische User mit Passwort für Interfaces und Applikationen
  - Technische User, die die Daten halten. Hier sollte die direkte Anmeldung nicht mehr so “einfach” möglich sein
    - Gutes und geheimes Passwort
    - User locken und bei Bedarf entsperren

- New Feature 18c - Schema Only Accounts
  - Anlegen mit :

```
CREATE USER schema_owner NO AUTHENTICATION
  QUOTA UNLIMITED ON users;
```

Zugriff von Administratoren oder Entwicklern nur über PROXY Rechte !

1 – Wie lief das im Projekt

Dauer für eine DWH Umgebung >>> 3 Monate

- Keine Doku, keine Ansprechpartner, Business Owner kennen die Applikationen nicht, externer Dienstleister “unbedarft” …..

- Im ersten Schritt wurde eine Bestandsanalyse durchgeführt, Ansprechpartner identifiziert und alle Interfaces und Applikationen als “Bild” pro Interface erstellt

=> Erster Einstieg in eine Betriebsdokumentation

Ziel – eine Matrix von persönlichen und technischen User identifizieren

Einführen der Persönlicher User (1)

- Jeder Anwender, der direkt mit den Daten der Datenbank arbeiten muss, erhält ein persönliches Schema in der Datenbank
- Anwender wird über sein zugeordnetes Profil und den Präfix im Schema Namen identifiziert
  - Schema Name entspricht möglichst dem AD Account, um später diesen User mit dem AD auch automatisch abgleichen zu können
  - Z.B., Windows User “pipperr” => DB User **P**_PIPPERR

Einführen Persönlicher User (2) – Zwei Gruppen

Read Only User

- Fragt nur Daten ab, ändert keine Werte in der DB
- Benötigt keine Proxy Rechte
- Hat eine Lese Rolle auf die Schemas, die er sehen soll
- Views um die Arbeit zu erleichtern

User muss Daten anpassen

- Verwendet Proxy um sich direkt auf dem Schema anzumelden
- Rollen mit Passwort (Secure Application Roles) verwenden um z.B. DML Operationen zu verbieten
- VPD – Virtual Private Data Feature einsetzen um kritische Daten vor dem User zu verbergen

Einführen einer Lese Rolle für Persönliche User

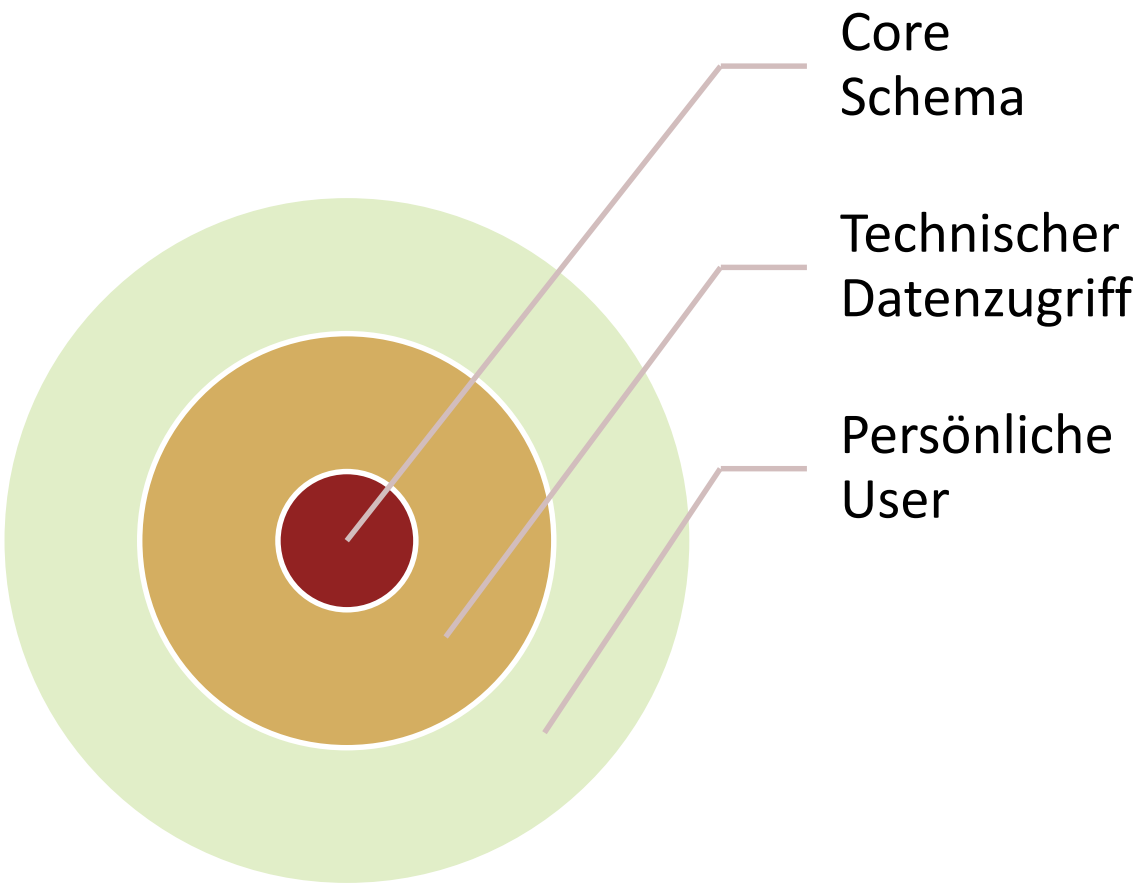
- Um die Verwaltung in der Datenbank zu vereinfachen wird für jedes technische Core Schema eine Lese Rolle definiert

- Feste Namesgebung, wird beim Anlegen eines Schemas gleich mit beantragt => ROLE_READ_<SCHEMA>

- Rolle wird automatisch mit allen Tabellen des Schemas befüllt über PL/SQL Routine
  - Vorteil => Wartungsarm
  - Nachteil => Nicht sehr feingranular

2 – Technische Lösung finden

- Ziel: Zugriff auf die Daten so stark schützen wie möglich

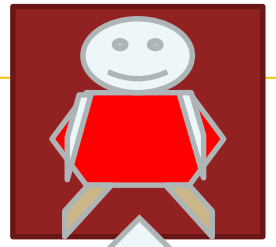


2 – Überblick

Interface Script

Keine Passwörter in Skripts!

Oracle Wallet



SSL Verschlüsselung

Restrict IP address — **Listener**

Restrict on Business Rules — **LogOn Trigger**

Report Schema

Synonym

Personal Schema

Interface Schema

Synonym

Role

Role

Core Data Schema

Table

Direct Connect only
As Proxy
over personal user

Welche Funktionen stehen uns dazu zur Verfügung

- Schema und Passwort
- Oracle Wallet für Oracle Secure External Passwort Store
- Sicheres Handling von Passwörtern in Scripts
- Objects Grants und Synonyms
- VPD – Virtual Private Database und Fine Access Control (Nur EE!)
- Datenbank Rollen
- Profile und Oracle Workload Management
- Oracle Proxy User Concept
- Login Trigger
- SQL*NET Verschlüsselung und Listener Security auf IP Adressen

Schema User und Passwort

- Wo wird am besten die Information hinterlegt, welcher wirklicher User im Unternehmen hinter einem DB User steht?
  - Unterschiedliche Profile als Flag verwenden
    - Für jede Nutzerklasse ein eigenes Profil definieren und den Usern zuweisen
  - Windows Login im Namen codieren und Schema-Namen flaggen
    - Wie für PIPPERR ein P_PIPPERR

- Ab 12c – Column ORACLE_MAINTAINED für interne DB User

18c – New Feature – Schema OHNE Authentifzierung

- Schema OHNE jede Authentifizierung
  - Kann NUR mit Proxy Connect genützt werden

```
CREATE USER schema_owner NO AUTHENTICATION
  QUOTA UNLIMITED ON users;
```

Oracle Secure External Passwort Store (1)

- Wallet anlegen

```
# Wallet nur unter diesem User und auf diesem Server gültig
orapki wallet create -wallet "." -auto_login_local
# Wallet kann verteilt werden
mkstore -wrl . -create
```

- Username und Passwort hinterlegen – Schlüssel ist ein TNS Alias

```
mkstore -wrl <wallet_location> -createCredential <db_connect_string> <user_name> <password>

mkstore  -wrl .  -createCredential orapkigpi gpi passwOrdF0rGPISecurePAsSwOrD
```

- Zugriff auf Wallet in sqlnet.ora konfigurieren

```
WALLET_LOCATION =
  (SOURCE =
    (METHOD = FILE)
    (METHOD_DATA =
      (DIRECTORY = c:\oracle\client_wallet)
    )
  )

SQLNET.WALLET_OVERRIDE=TRUE
```

Oracle Secure External Passwort Store (1)

- TNSAlias in der tnsnames.ora setzen

```
orapkigpi =
(DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCPS)(HOST = 10.10.10.1)(PORT = 2484))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = GPI)
    )
  )
```

- Ohne Angabe eines Users oder Passworts kann nun auf die DB zugegriffen werden

```
Sqlplus /@orapkigpi
```

https://www.pipperr.de/dokuwiki/doku.php?id=dba:oracle_secure_external_passwort_store

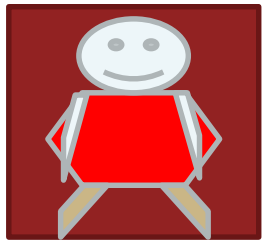
Passwort Sicherheit in Skripts

- Siehe dazu ausführlichen Artikel in der Red Stack
  - **Red Stack Magazin der DOAG, September 2018**

- Link GPI Wiki => https://www.pipperr.de/dokuwiki/doku.php?id=dba:passwort_verschluesselt_hinterlegen

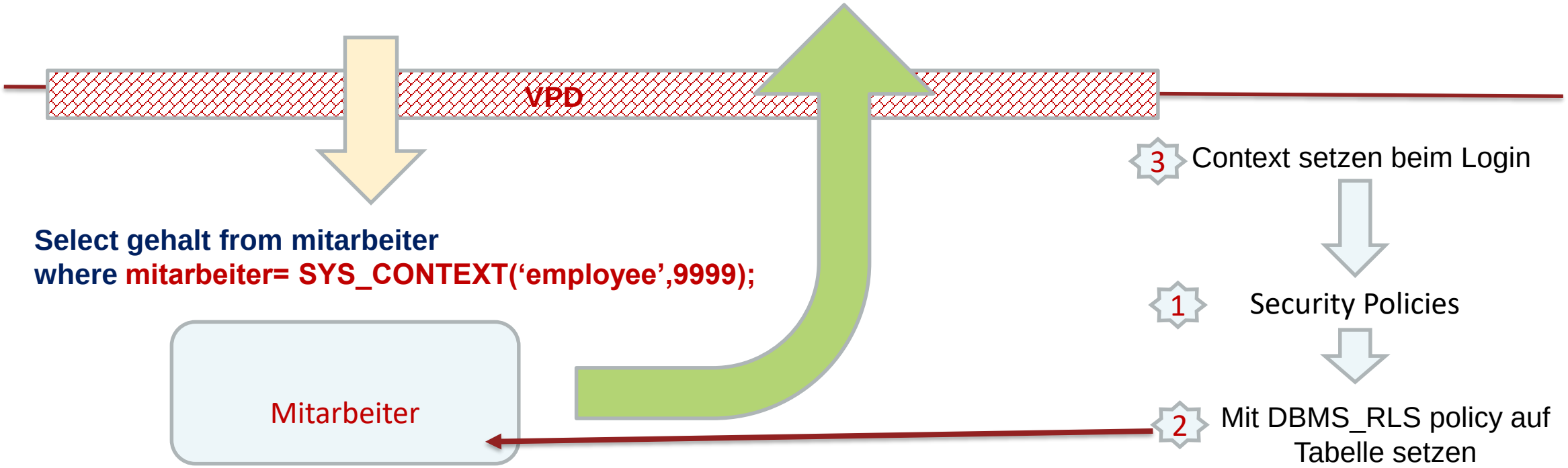
VPD - Virtual Private Database

Abfrage:



Select gehalt from mitarbeiter

Nur die eingeschränkten Daten sind sichtbar



Die Virtual Private Database (1)

Virtual Private Database – NUR EE Edition notwenig!

Oracle Label Security => EE + Extra Option notwendig

- Zugriff auf Tabellenebene prinzipiell einschränken
  - Problem:
    - Über normale Rechte kann das Lesen auf einer Tabelle nur auf alles oder nichts gesetzt werden
    - Rechte auf Zeilen können über die Anwendung geprüft werden, aber die Anwendung kann mit SQL*Plus umgangen werden
    - Mit Views sind nur bedingte Einschränkungen möglich
    - Views sind nur schwer zu warten bei sehr vielen unterschiedlichen Berechtigungen
  - Lösung: Virtual Private Database
    - Bei jedem Zugriff auf eine Tabelle wird zuvor eine versteckte Funktion ausgewertet, die das SQL Statement entsprechend einschränkt
    - Siehe auch Oracle Label Security

https://docs.oracle.com/en/database/oracle/oracle-database/18/dbseg/using-oracle-vpd-to-control-data-access.html

Die Virtual Private Database (2)

- Fine Grained Access Control
  - Package DBMS_RLS
  - Eine Funktion wird einer Tabelle zugeordnet
  - Beim Parsen eines SQL-Statements wird das Ergebnis der Funktion in die Where-Bedingung mit eingearbeitet
  - Beispiel: (Angestellter darf nur seinen Datensatz sehen)
    - Funktion erzeugt Prädikat:
      empno=(select empno from emp
      where ename = sys_context('userenv','session_user')
    - Angestellter sucht mit select * from emp
    - Automatisch wird das SQL-Statement erweitert um:
      select * from emp where empno=(select empno from emp
      where ename = sys_context('userenv','session_user')
- Application-Context
  - sys_context benutzerdefiniert erweitern

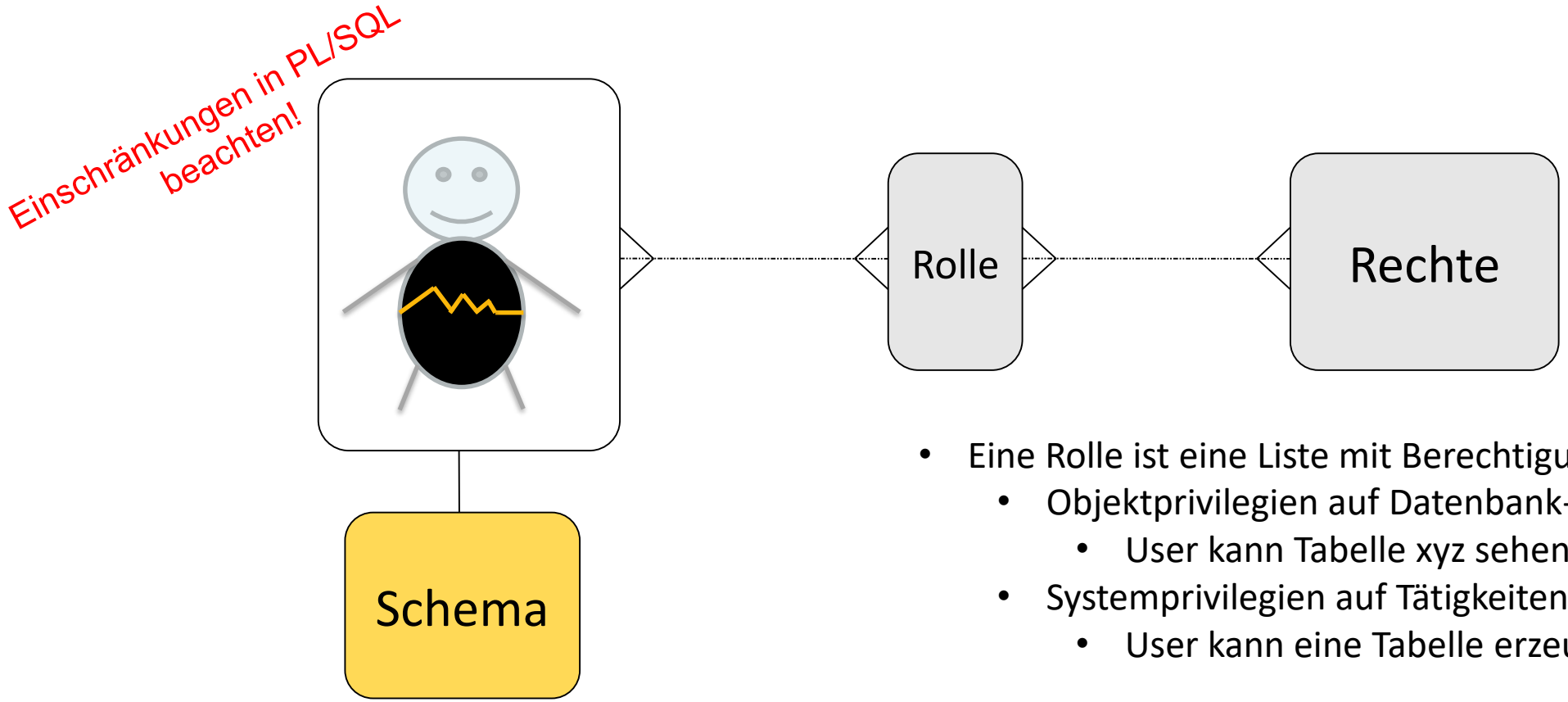
Kritische Daten maskiert ausgeben

EE + Oracle Advanced Security notwendig

- Kritische Daten wie Kreditkarten Daten in der Select Liste eines SQL maskieren
- Lösung: Data Redaction mit Policies in der DB
  - EE Feature Advanced Securtiy
  - Beispiel => https://oracle-base.com/articles/12c/data-redaction-12cr1

Arbeiten mit Rollen

- Einer Rolle werden diverse Rechte zugeordnet
- Einem Benutzer werden eine oder mehrere Rollen zugeordnet



- Eine Rolle ist eine Liste mit Berechtigungen:
  - Objektprivilegien auf Datenbank-Objekte
    - User kann Tabelle xyz sehen
  - Systemprivilegien auf Tätigkeiten in der Datenbank
    - User kann eine Tabelle erzeugen

PLSQL und Rollen – ein Drama

- In PL/SQL können Rechte über Rollen nicht verwendet werden
  - Rollen sind nicht statisch und können sich ändern!
  - Explizite Rechte an den Tabellen sind für das jeweilige andere Schema notwendig!

Welche Rolle habe ich nun in meiner Session?

- Tabellen session_privs und session_roles abfragen

```
prompt ... will show you are privileges on the database

select * from session_privs
/

select * from session_roles
/
```

https://github.com/gpipperr/OraPowerShell/blob/master/Ora_SQLPlus_SQLcL_sql_scripts/my_user.sql

Rollen einschränken

- Nicht per default zuweisen
  - Rolle muss vom Anwender explizit gesetzt werden
    - Set role xxxxxxx
  - Security by obfusikation
- Rolle mit einem Passwort versehen
  - User / Applikation muss das Passwort kennen um die Rolle zu verwenden
  - Schutz z.b. “versehentlichen” DLM wie drop table im falschen Schema
  - Falle beim Upgrade auf 12c! Passwort muss neu gesetzt werden!
- PL/SQL Regel zum aktivieren einer Rolle definieren – Secure Application Role
  - Selbstdefinierte Regeln legen fest, was zutreffen muss, damit eine Rolle auch aktiv wird
  - Rolle kann nur über dieses Stück PL/SQL aktiviert werden

Rolle mit einem eigenen Passwort versehen

- Das aktiveren einer Rolle kann mit einem Passwort geschützt werden:

```
SYS> create role tester identified by tester;
```

```
SCOTT> set role tester identified by tester;
```

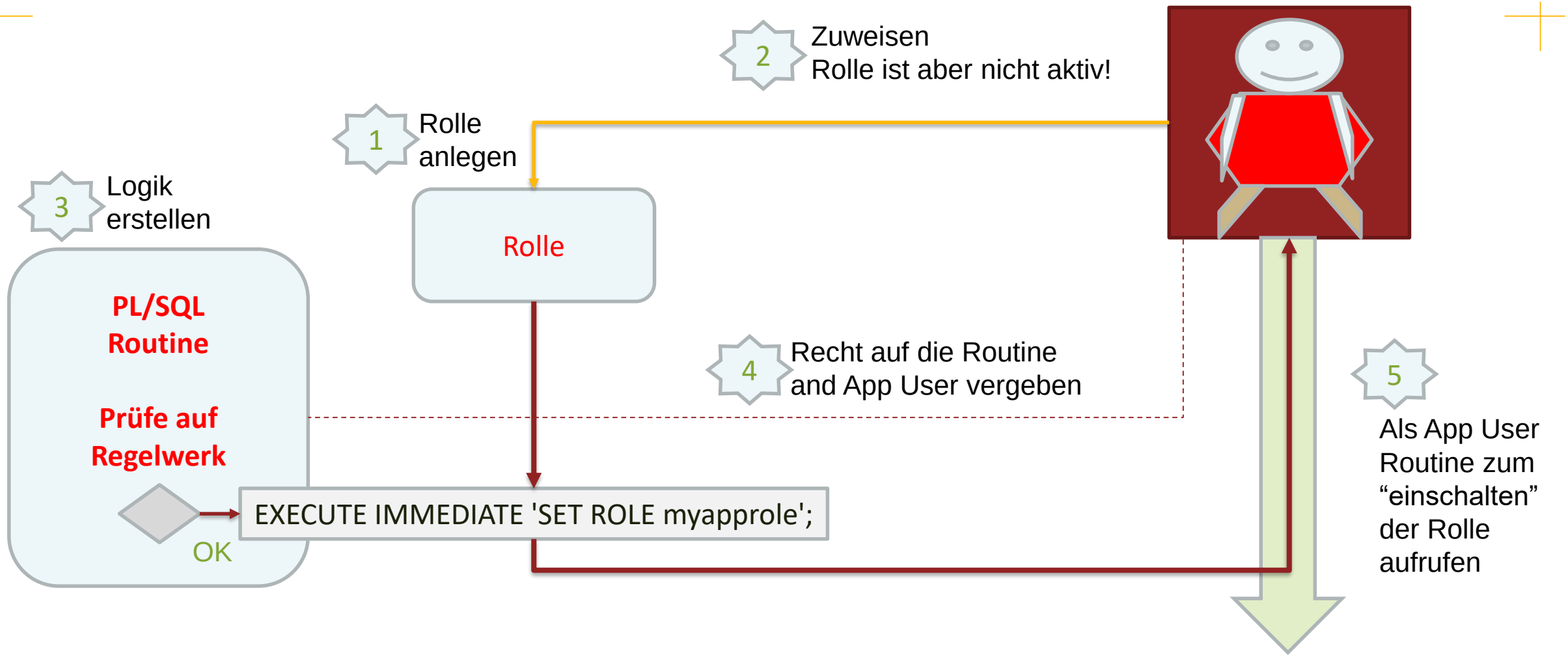
AUTID Definer Package Beispiel

```
CREATE OR REPLACE PROCEDURE set_read_role
AUTHID DEFINER
is
 cursor c_tables
  is select table_name from user_tables;
  v_schema varchar2(32):=sys_context( 'userenv', 'current_schema' );
begin
 for rec in c_tables
  loop
    execute immediate
      'grant select on ,
     ||rec.table_name
     ||' to ROLE_READ_'||substr(v_schema,1,20);
 end loop;
end;
/
```

Idee : Rechte in der DB vergeben ohne DBA oder irgendwelche ANY Rechte

Ein weiters Beispiel : Als normaler User DBA Rechte zum Killen von Sessions verwenden:
https://www.pipperr.de/dokuwiki/doku.php?id=dba:kill_session_without_dba_rights

Secure Application Role - Rolle per PL/SQL aktiveren



https://www.pipperr.de/dokuwiki/doku.php?id=dba:secure_application_role

Secure Application Role (1) - Code

– Definieren mit

```
create role myAPPRole identified using checkMyAppRules;
```

– Für die Aktivierung einer Rolle, kann mit einem selbstgeschriebenen PL/SQL Package die Regelkonformität überwacht werden

```
CREATE OR REPLACE PROCEDURE secadmin.checkMyAppRules
 AUTHID CURRENT_USER
 AS
 BEGIN
  IF (SYS_CONTEXT ('userenv','ip_address')
    BETWEEN '192.0.2.10' and '192.0.2.20'
     AND
    TO_CHAR (SYSDATE, 'HH24') BETWEEN 8 AND 17)
  THEN
    EXECUTE IMMEDIATE 'SET ROLE myAPPRole ';
  END IF;
 END;
/
Grant execute on secadmin.checkMyAppRules to scott;
```

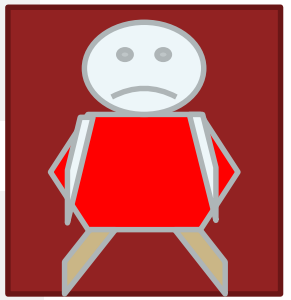
Hier kann jede Logik stehen

Secure Application Role (2) - ORA-28201 Fehler

- User versucht die Rolle einfach so zu aktivieren

```
BEGIN dbms_session.set_role('myAPPRole'); END;

ERROR at line 1:
ORA-28201: invalid command to enable secure application role myAPPRole'
ORA-06512: at "SYS.DBMS_SESSION", line 194
ORA-06512: at line 1
```



```
set role myAPPRole
*
ERROR at line 1:
ORA-28201: invalid command to enable secure application role myAPPRole'
```

Prüfen welche Rollen in der Session aktiv sind => **select * from session_roles**;

Rolle läßt sich nur über den Aufruf des PL/SQL aktivieren!

- Rolle aufrufen

```
exec secadmin.checkMyAppRules;
```

- Test mit:

```
select * from session_roles;

…
myAPPRole
```

Daran denken das ein SET Role alle anderen Rollen deaktiviert!

Falls die Default Rollen des Users weiterhin gültig sein sollen, diese mit in den set Befehl als komplette Liste aufnehmen!

Der User muss die entsprechenden Execute Rechte auf die PL/SQL Procedure besitzen!

Login Trigger verwenden um die Rolle zu aktivieren

Rollen in der Datenbank kontrollieren

- VIEW DBA_APPLICATION_ROLES

Set Role Verhalten beachten!

- Ein "Set Role" setzt alle bestehenden Rollenzuordnungen zurück und enabled nur die angebene Liste von Rollen.
- Ein "Set Role ALL" enabled alle zugewiesenen Rollen, allerdings nicht die "Secure Application Roles" !

- D.h. wenn die bestehenden Rollen beibehalten werden sollen, muss in das SET Role Kommando die Liste der gerade aktvierten Rollen mit aufgenommen werden!

Einführen einer Lese Rolle pro technisches Schema

- Für jedes technische Schema wird eine Lese Rolle angelegt
  - ROLE_<SCHEMA>_READ

Profile verwenden um User Klassen zu unterscheiden

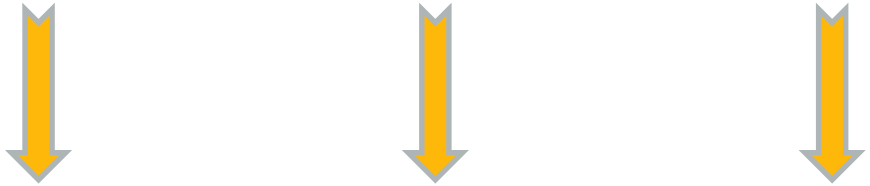
- Über die Verwendung eines Profils kann auch die Verwendung von Ressourcen eingeschränkt werden
  - Die Eigenschaften des Passworts können definiert werden (z.B.: wann es abläuft)
  - Ein default Profil für alle Benutzer
  - Ein definiertes Profil pro Benutzer möglich, damit die User zum Unterscheiden „flaggen2
  - Profile in der DB abfragen mit:

```
sys:gpi>select * from dba_profiles;
PROFILE    ROURCE_NAME                          RESOURCE LIMIT
---------- ------------------------------------ -------- ----------
DEFAULT    COMPOSITE_LIMIT                      KERNEL   UNLIMITED
DEFAULT    FAILED_LOGIN_ATTEMPTS                PASSWORD UNLIMITED
.....
DEFAULT    CONNECT_TIME                         KERNEL   UNLIMITED
DEFAULT    PRIVATE_SGA                          KERNEL   UNLIMITED
```

Oracle Proxy User Feature

- Mit seinem eigenen persönlichen User an einem fremden Schema anmelden
  - Ein persönlicher User hat seinen eigenen Account in der DB
  - Dem User wird das Recht vergeben sich an einem anderen Schema mit seinem Passwort anzumelden
  - Einfache [] Syntax

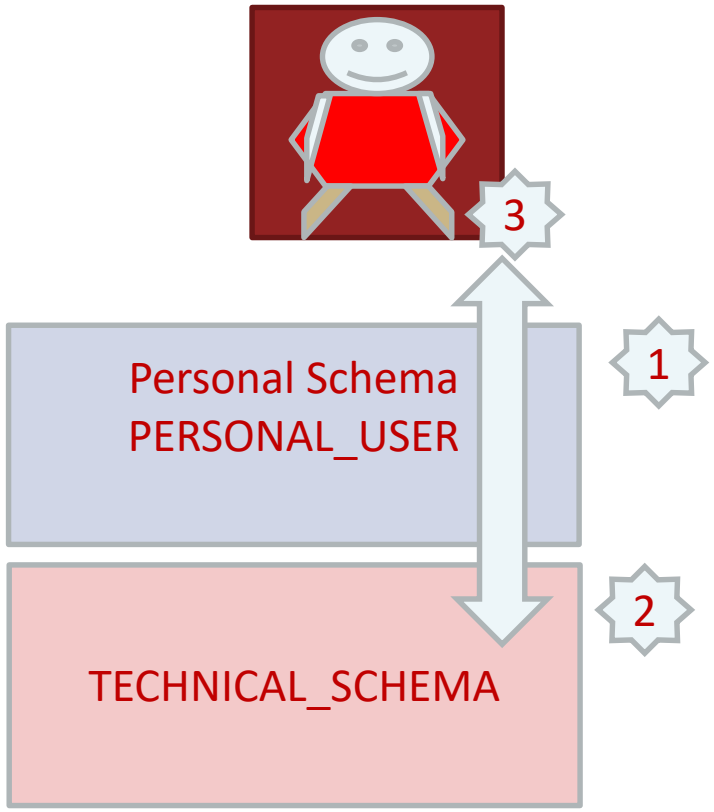
Your Login String - <YOUR_USER_NAME>[TECHNICAL_USER]/<YOUR_PASSWORD>



```
Sql>connect P_PIPPERR[APP_OWNER]/passw0rd
```

Proxy Feature

- Use Oracle Proxy



1- Anlegen des persönlichen Users

```
Create user PERSONAL_USER ….
```

2- Auf dem TECHNICAL_SCHEMA die Proxy Rechte vergeben

```
Alter user TECHNICAL_SCHEMA grant connect through PERSONAL_USER;
```

3 - Connect als Proxy

```
sqlplus PERSONAL_USER [TECHNICAL_SCHEMA]@local_db
```

Kann auch unter Java einfach implementiert werden

Den Proxy User in der DB nachweisen

- SYS_CONTEXT

```
SQL> connect gpi[hr]/gpi
```

```
SELECT     SYS_CONTEXT ('USERENV', 'CURRENT_USER')    CURRENT_USER
         , SYS_CONTEXT ('USERENV', 'CURRENT_USERID') CURRENT_USERID
         , SYS_CONTEXT ('USERENV', 'PROXY_USER')      PROXY_USER_ID
         , SYS_CONTEXT ('USERENV', 'PROXY_USERID')    PROXY_USER
         , SYS_CONTEXT ('USERENV', 'AUTHENTICATED_IDENTITY') AUTH_AS
FROM DUAL
/
```

```
CURRENT          CURRENT          PROXY            PROXY            AUTH
USER             USERID           USER             USERID           IDENT
-------------- -------------- -------------- -------------- --------------
HR               102              GPI              121              HR
```

SYS_CONTEXT
=> https://docs.oracle.com/database/121/SQLRF/functions199.htm#SQLRF06117

Den Proxy User überwachen

- DB Auditing
  - Der Proxy User und die eigentliche Session müssen gejoint werden

```
SELECT   a.username connecting_user
       , a.action_name
       , a.timestamp
       , a.comment_text
       , b.username proxy_username
       , b.action_name proxy_action_name
       , b.comment_text proxy_comment_text
 FROM dba_audit_trail a INNER JOIN dba_audit_trail b ON (a.proxy_sessionid=b.sessionid)
ORDER BY a.timestamp
/
```

Proxy DD Views

- Wie können die Proxy Rechte in der DB überwacht werden

  - DBA_PROXIES - Get all information from proxy users
  - USER_PROXIES – To which schemas I can connect
  - PROXY_USERS – Who can use which Proxy connect

- Wer kann sich an einem Schema anmelden

```
SELECT  proxy
        , client
        , authentication
        , authorization_constraint
        , ROLE
        , proxy_authority
 FROM dba_proxies
WHERE client LIKE UPPER('&&SCHEMA_NAME.')
ORDER BY 1
/
```

```
                Client                Auth
Proxy           User            Auth  Const                                 Role            Proxy
              ------------- ----- ------------------------------------- -------------- ----------
GPI             HR              NO    PROXY MAY ACTIVATE ALL CLIENT ROLES                   DATABASE
```

Testcase – Was passiert in den verschiedenen User Status

- Vergleich verschiedener User Status

Account Status	Technical User	Personal user	Login
Open	**OPEN**	**OPEN**	**YES**
Locked	**Not locked**	**LOCKED**	**NO**
Expired	**Not expired**	**EXPIERED**	**NO**
Locked	**LOCKED**	**Not locked**	**NO**
Expired	**EXPIERD**	**Not expired**	**YES**

As Proxy User ist der Connect an ein Schema mit abgelaufenen Password möglich

Proxy Feature und Rolen

- Mit dem Proxy Login kann auch gleich die richtige Rolle gesetzt werden!

- Use Role to set the rights of this proxy session!
  - For Example
  - TECHNICAL_SCHEMA has only as default role „connect“
  - To use a Role like RESOUCE you need a Password
  - If Proxy User connect he get a role only with the right he need for his work

```
ALTER USER TECHNICAL_SCHEMA GRANT CONNECT THROUGH PERSONAL_USER WITH ROLES hr_user_role;
```

Wartbarkeit über ein PL/SQL Hilfsprogramm erweitern

- In jeden Schema liegt eine PL/SQL Routine mit “AutID Definier”
- Beim Aufrufen der Procedure wird die Lese Rolle mit den Select Rechten auf Tabellen und Views gefüllt.
- Ein Admin Schema hat entsprechende Rechte und kann damit die Rollen mit Zugordnungen versorgen OHNE eigene Rechte in den jeweiligen Schemas

Oracle Listener härten

- Listener Security auf IP Adressen
  - Den Oracle Listener auf IP Adressen einschränken
  - Die sqlnet.ora des Listeners anpassen

```
TCP.VALIDNODE_CHECKING = YES

# Ausschließen
TCP.EXCLUDED_NODES= (10.10.*)

#Einladen
TCP.INVITED_NODES=(localhost,10.10.10.*)
```

Invited hat die höhere Priorität

Alternativ DB Firewall für Arme – CMAN der Connection Manager
https://www.pipperr.de/dokuwiki/doku.php?id=dba:sqlnet_cman_connection_manager

Projekt Resümee

- Ein sehr langer Weg …..
- Mehr als 2 Jahre, bis alle Passwörter sicher geändert werden konnten

- Mehr als 2 Jahre zum Einführen einer Lese Rolle

- Nur die einfachsten Basics konnten umgesetzt werden

Ursache: Unwissen, Kostenstellen denken, unwillige Hoster , unklare Zuständigkeiten, lange Subunternehmer Ketten

F&A

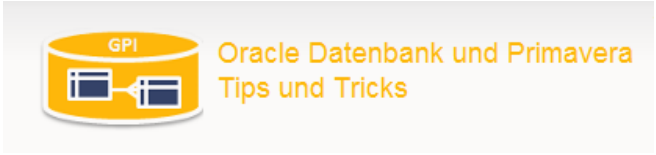
Fragen

Questions?

Quellen

- Oracle Dokumentation und Support Portal

- https://www.pipperr.de/dokuwiki/doku.php



- Wieder mal eine andere Script Library
  - https://github.com/gpipperr/OraPowerShell

- Bildmaterial : https://pixabay.com und https://publicdomainvectors.org