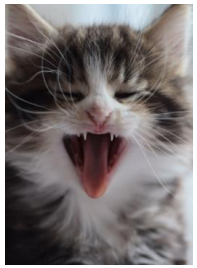


DOAG 2021 Datenbank mit Exaday
Online Konferenz vom 17.Mai bis 18.Mai 2021

Job Steuerung in der Oracle Datenbank

JOB STEUERUNG IN DER DATENBANK MIT DEM ORACLE SCHEDULER – AB 19C EIN MUSS

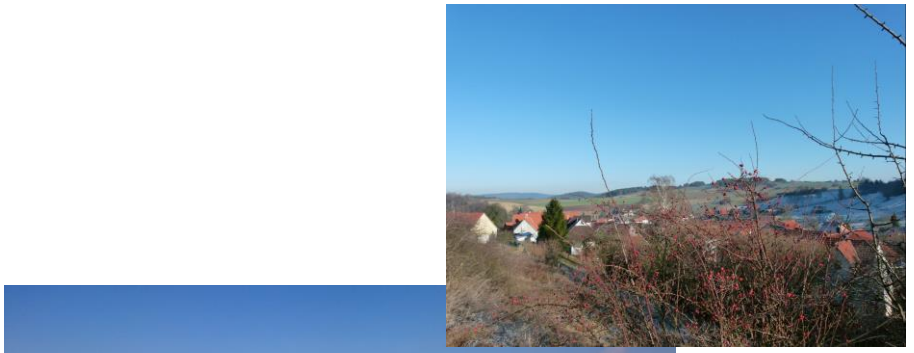
GPI Consult Gunther Pippèrr

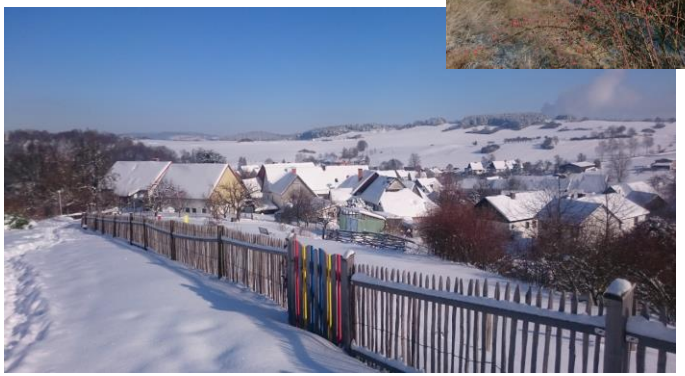
ORACLE ACE Associate

gunther@pipperr.de

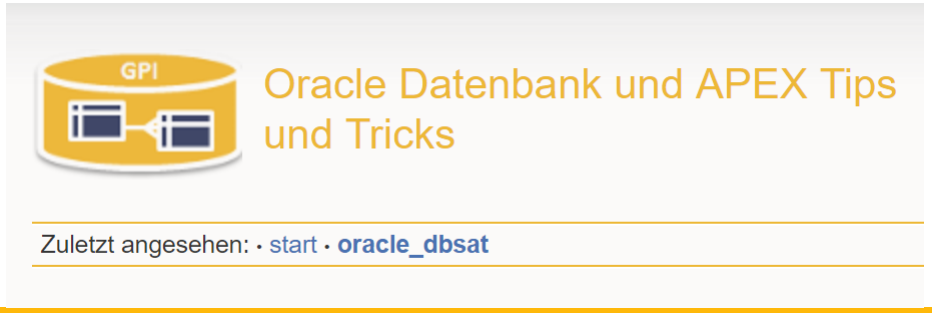
Mein Blog

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





Bergweg 14 - 37216 Witzenhausen/Roßbach



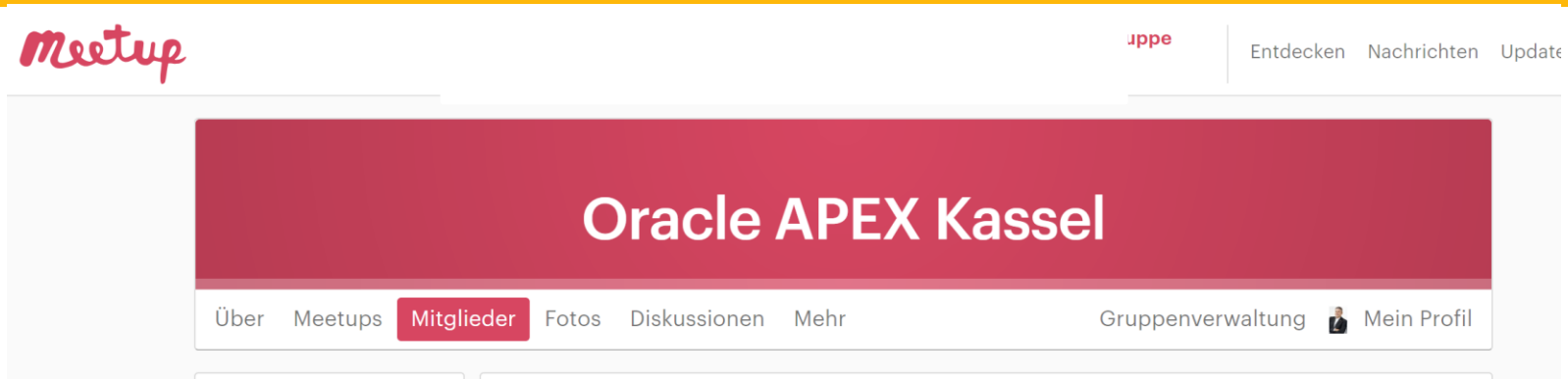
Freiberuflicher Oracle Datenbank Experte - Ich unterstütze Sie gerne in ihren Projekten.

APEX Meetup Gruppe Kassel-Göttingen

Raum für Veranstaltung in Kassel oder Göttingen gesucht, können Sie uns unterstützen?

Mitglieder gesucht!

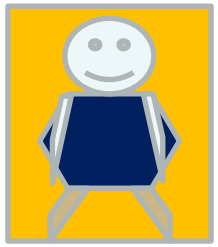
https://www.meetup.com/de-DE/Oracle-APEX-Kassel/



Agenda

1	**Die Ausgangslage / Intention / Szenarien**
2	Job Steuerung Architektur
3	Feature Set des Oracle Schedulers
4	APEX Automation
5	Fazit

Warum nicht Job Steuerung mit DBMS_JOB?

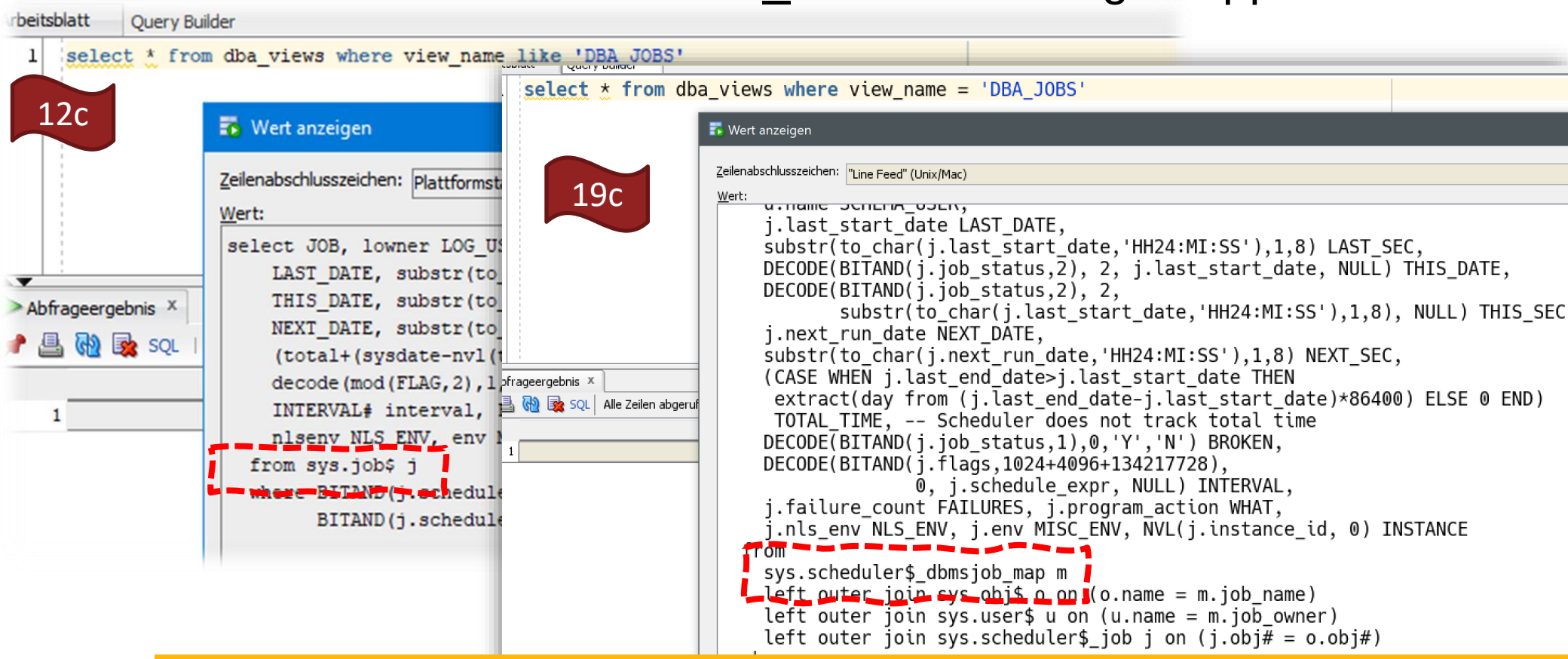


- Ist doch viel einfacher
- Warum was neues verwenden
- Das lief doch schon immer so

- Seit Version 12.2.0.1 „deprecated“
- Seit Version 19c gemappt auf DMBS_SCHEDULER

Die Ausgangslage – 19c und DBMS_JOB (1)

- Ab Oracle 19c keine DBMS_JOBS mehr!
  - Alles wird intern auf DBMS_SCHEDULER gemappt



Mapping ALT auf NEU über => **sys.scheduler$_dbmsjob_map**

Die Ausgangslage – 19c und DBMS_JOB (2)

- DBMS_JOB kann noch verwendet werden, führt aber zu einem Scheduler Job mit allen damit verbunden Abhängigkeiten und Funktionalitäten!

19c

```
1  BEGIN
2
3  DBMS_JOB.isubmit (
4      job       => 66
5      ,what     => 'begin null; /*doNix Job */ end;
6      ,next_date => SYSDATE
7      ,interval  => 'SYSDATE + 1/24 /* 1 Hour */'
8  );
9  COMMIT;
10
11 END;
12 /
```

Abfrageergebnis | Skriptausgabe

Aufgabe abgeschlossen in 0,241 Sekunden

```
PL/SQL-Prozedur erfolgreich abgeschlossen.
```

```
1  select * from dba_scheduler_jobs where job_name like 'DBMS%'
```

Single Record View

OWNER	SYS
JOB_NAME	DBMS_JOB$_66
JOB_SUBNAME	
JOB_STYLE	REGULAR
JOB_CREATOR	SYS
CLIENT_ID	
GLOBAL_UID	
PROGRAM_OWNER	
PROGRAM_NAME	
JOB_TYPE	PLSQL_BLOCK
JOB_ACTION	begin null; /*doNix Job */ end;
NUMBER_OF_ARGUMENTS	0
SCHEDULE_OWNER	
SCHEDULE_NAME	
SCHEDULE_TYPE	PLSQL
START_DATE	13.04.21 12:31:59,000000000 EUROPE/BERLIN
REPEAT_INTERVAL	SYSDATE + 1/24 /* 1 Hour */
EVENT_QUEUE_OWNER	

Im Vergleich

DBMS_JOB

- Kein automatisches Commit beim Anlegen des Jobs
- Einfacher in der Definition
- Für komplizierte “Next Job” Zeiten oft umständlich
- Nur Aufruf von PL/SQL

DBMS_SCHEDULER

- Automatisches Commit beim Anlegen des Jobs
- Komplexer in der Definition
- Log vergangener Job Läufe
- Kalender für Schedules
- Event Basierender Aufruf
- OS Jobs / Remote Jobs und mehr

Fallstricke bei der Umstellung – NLS (1)

- Beim Anlegen eines Jobs (DBMS_SCHEDULER oder DBMS_JOB) werden die NLS Settings der anlegenden Umgebung mit in die Definition des Jobs übernommen!
  - NLS_ENV Spalte von DBA_SCHEDULER_JOBS oder DBA_JOBS
    - The Priority of NLS Parameters Explained (Where To Define NLS Parameters) (Doc ID 241047.1)„

- Die NLS Einstellungen können sich bei der automatischen Migration verändern
  - Da die NLS Settings der Migrations-Session bei der Neu-Anlage des Jobs verwendet werden!

Fallstricke bei der Umstellung – NLS (2)

- Auf einem Scheduler Job lassen sich die NLS Einstellungen expliziert einstellen
  - **dbms_scheduler.set_attribute**

```
BEGIN
DBMS_SCHEDULER.SET_ATTRIBUTE (
    name           => 'GPI.DB_TRANS_JOB'
   ,attribute      => 'NLS_ENV'
   ,VALUE           => q'[
      NLS_LANGUAGE='GERMAN'
          NLS_TERRITORY='GERMANY'
          NLS_CURRENCY='€'
          NLS_ISO_CURRENCY='GERMANY'
          NLS_NUMERIC_CHARACTERS=',.'
          NLS_DATE_FORMAT='DD.MM.YYY'
          NLS_DATE_LANGUAGE='GERMAN'
          NLS_SORT='XGERMAN'
   ]'
);
END;
```

Lizenz?

- In der Datenbank
  - In allen DB Editionen enthalten
  - Oracle Scheduler ab Oracle 10g

- Oracle Remote Scheduler Agent
  - „Oracle Database installed on Remote Host“ => enthalten
  - Keine Oracle Installation vorhanden => **Unklar ….**

Szenarien

Wo bei kann uns der Scheduler so alles helfen?

ETL Szenarien umsetzen

- Für welche Szenarien können wir die erweiterten Möglichkeiten von Oracle Scheduler Jobs verwenden?

- **Ziel :**
  Alles über die Datenbank zentral steuern und konfigurieren

- Wie:
  - Dateien bei Bedarf verarbeiten und Einlesen
  - Auf einem Event in der Datenbank reagieren und Business Prozesse anstoßen
  - Auf Fehler reagieren und Job Abhängigkeiten beachten

ETL Szenario mit Scheduler Job Feature (1)

- Quelle liefert “gelegentlich” auf Server in die DMZ, diese müssen zuerst in Richtung Datenbank “versandt” werden
  - Wenn die Daten eintreffen müssen die Daten gleich verarbeitet werden

Filewatcher Funktionalität / External Job Fuktionalität

- Prüfe alle x Minuten ob die Datei da ist
- Falls ja, starte den Lade Job

https://www.pipperr.de/dokuwiki/doku.php?id=dba:oracle_scheduler_file_watcher_19c

ETL Szenario mit Scheduler Job Feature (2)

- Daten werden als CSV geliefert und müssen angepasst werden
  - Betriebssystem Programm muss aber die Daten zuvor formatieren

External Skript Funktionalität

Scheduler ruft im OS das Programm auf und formatiert die Daten

https://www.pipperr.de/dokuwiki/doku.php?id=dba:oracle_scheduler_12c_external_scripts

ETL Szenario mit Scheduler Job Feature (3)

- Daten müssen in die Datenbank eingelesen werden
  - Stage Tabelle muss gefüllt und Daten verteilt werden

Job Chain

War der Schritt zuvor erfolgreich, kann über “External Table” die Datei eingelesen werden

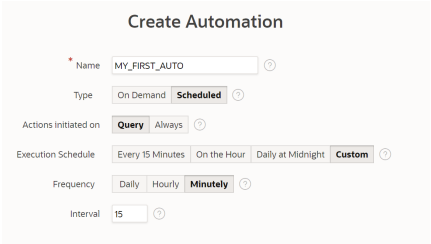
ETL Szenario mit Scheduler Job Feature (4)

- Nach dem Einlesen der Daten kann die Tagesverarbeitung starten
  - Liegen Daten vor, kann der Job A für die Tagesverarbeitung gestartet werden

Event Basierend

Liegen die Daten korrekt vor, startet der Tagesverarbeitungsjob

APEX Automations



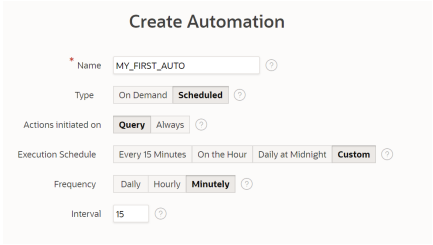
ETL Szenario mit Scheduler Job Feature (5)

- Fachabteilung soll per Mail informiert werden

Job Mail Feature

Information über die eigentliche Verarbeitung versenden

APEX Automations



Weitere Szenario für Jobs

- Parallelisierung
  - Starten von Light Wight Jobs um parallele Session für die Verarbeitung von Daten zu erstellen

Sicherheit

Rechte in der Datenbank

Rechte - Wer darf Scheduler Jobs definieren?

- Welche Rollen sind notwendig

  – Für mein Schema einen Job anlegen
    - Create Job

  – Job für andere Schemas aufrufen
    - Create any Job
    - Execute any program
    - Execute any class

  – Scheduler verwalten
    - Manager Scheduler

Rechte für Externe Ressourcen

- Betriebssystem Zugriff auf Filesystem / Scripts
  - Wird über ein “Credential Object” vergeben

```
-- Credentials anlegen
BEGIN
  DBMS_CREDENTIAL.CREATE_CREDENTIAL('OS_USER_JOB_CONTROL','job_control','my_secret_pwd');
END;    -- DD abfragen
/
        SELECT  credential_name
               ,username
           FROM   user_credentials
        ORDER BY credential_name
        /
```

Wo landen die Daten?
sys.scheduler$_credential
Vor 11.2.0.4 auslesen mit
DBMS_ISCHED.GET_CREDENTIAL_PASSWORD als sys

- Mail Versandt
  - ACL zum Zugriff auf dem Mail Server
  - Wallet bei Verschlüsselung

Siehe im Detail => https://www.pipperr.de/dokuwiki/doku.php?id=prog:plsql_send_mail_tls

Sicherheitesüberlegungen

- Kann der Oracle Scheduler ein Sicherheits-Risiko darstellen?
  - Vorsichtig mit „Any“ Rechten!
  - Vorsicht mit Betriebssystem Zugriff!
  - Auf aktuelle Hinweise achten wie:
    - Oracle Critical Patch Update Advisory - January 2021
      - **CVE-2021-2035 mit 8.8!**

CVE#	Component	Package and/or Privilege Required	Protocol	Remote Exploit without Auth.?	CVSS VERSION 3.1 RISK (see Risk Matrix Definitions)									Supported Versions Affected	Notes
					Base Score	Attack Vector	Attack Complex	Privs Req'd	User Interact	Scope	Confid-entiality	Inte-grity	Avail-ability		
CVE-2021-2035	RDBMS Scheduler	Export Full Database	Oracle Net	No	8.8	Network	Low	Low	None	Un-changed	High	High	High	12.1.0.2, 12.2.0.1, 18c, 19c	

Aktuelle Bugs

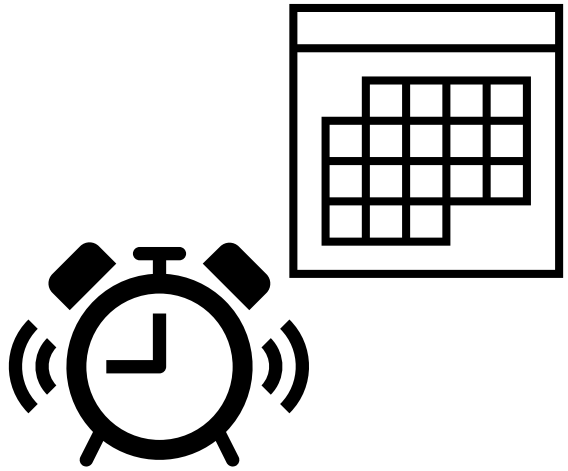
- Bug 28805242 - scheduler job suddenly stops running (Doc ID 28805242.8)

- https://oracle-base.com/blog/2019/08/29/q-is-the-create-job-privilege-required-in-19c-a-no/
  - SR 3-20860955641 : Jobs can be created without the CREATE JOB privilege. Bug 30357828.

- https://mikedietrichde.com/2020/05/21/dbms_job-one-off-patch-needed-for-oracle-19-3-0-19-7-0/
  - Support Note: 2645984.1 – Delete On SYS.SCHEDULER$_DBMSJOB_MAP Causing Row Locks .

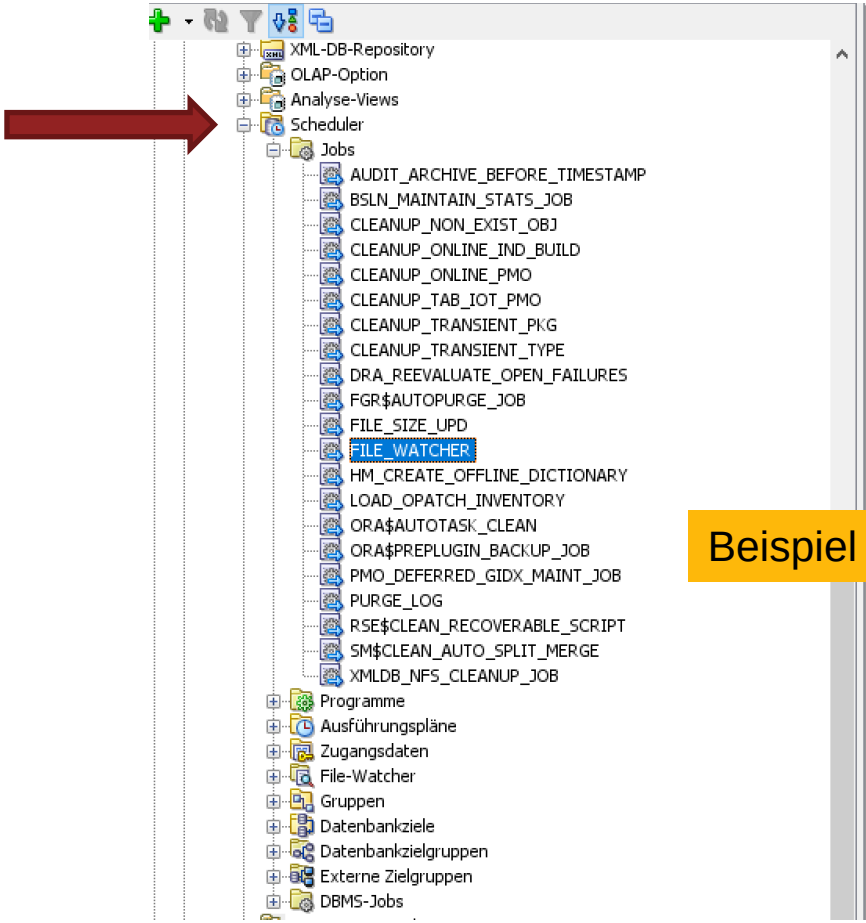
Datenbank Jobs im Detail

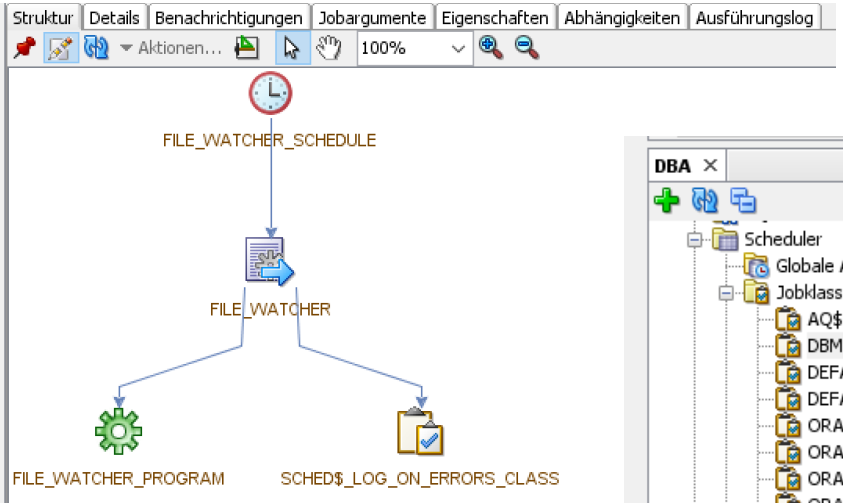
Architektur und Funktionalität von Datenbank Jobs



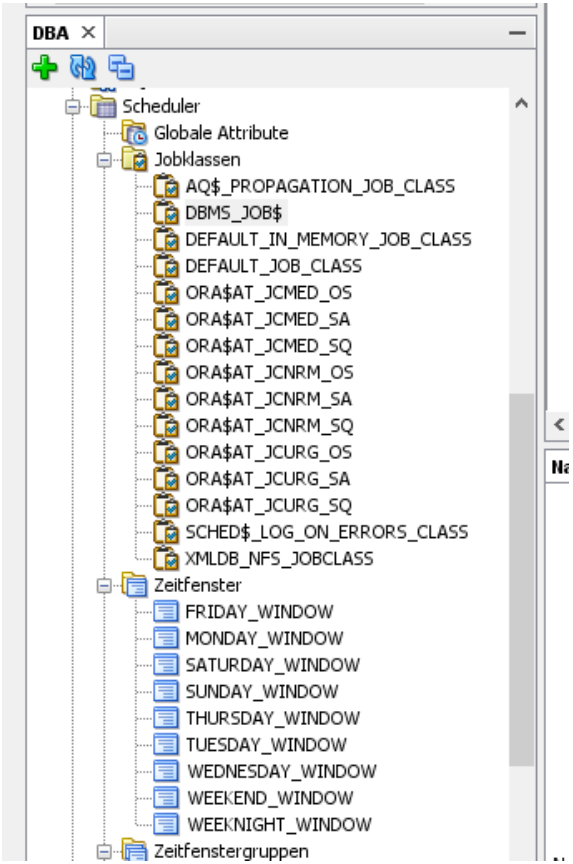
Graphische Oberfläche

- SQL Developer – Jeder User sieht nur seine Jobs!

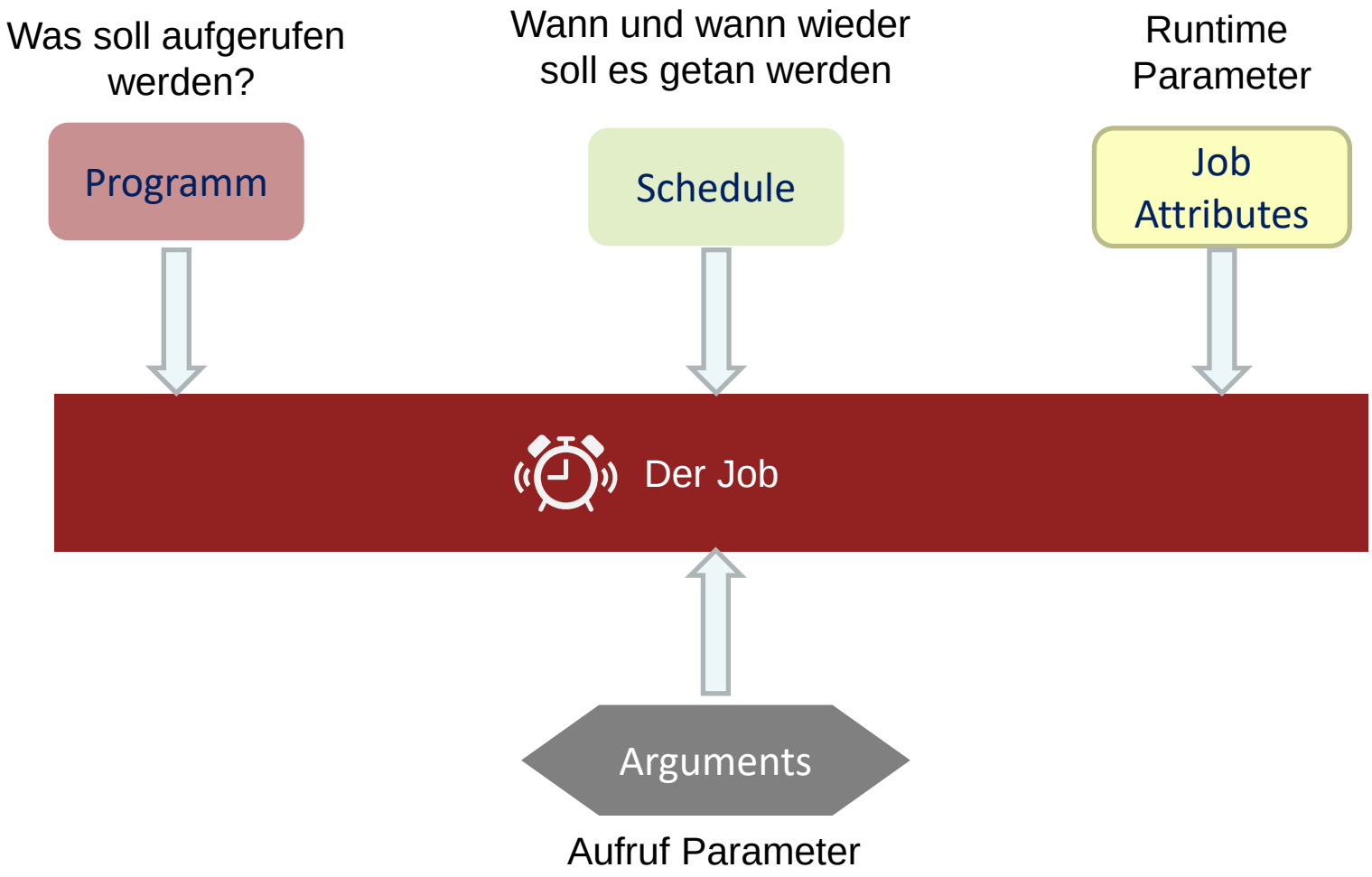




Beispiel für den SYS File Watcher Job

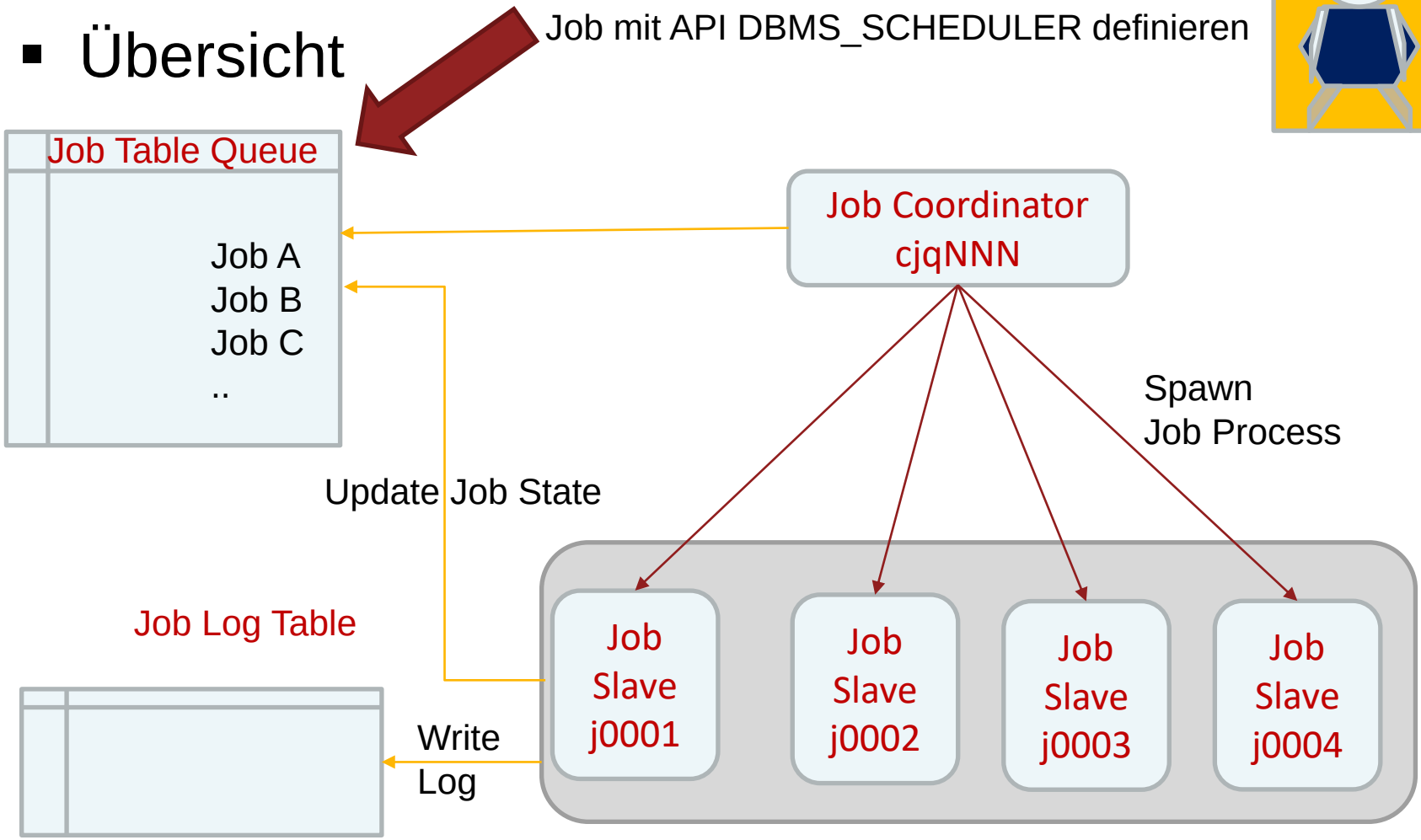


Aus was besteht ein Datenbank Job?



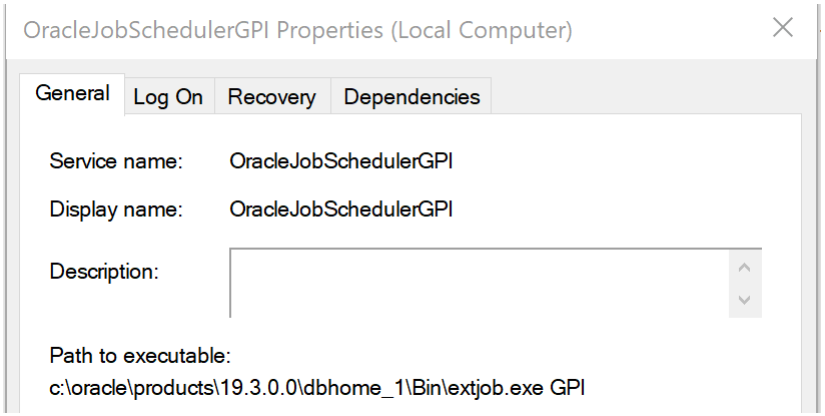
Architektur des Oracle Schedulers

- Übersicht



Oracle Scheduler Remote Agent

- Programm Aufruf kann auch auf einer anderen Maschine über den „Scheduler Remote Agent“ erfolgen
  - z.B. Windows:



  - Eine Oracle Datenbank muss auf dieser Maschine nicht installiert sein!

Siehe dazu:
https://oracle-base.com/articles/11g/scheduler-agent-installation-11gr2

Wichtige Begriffe zu DBMS Scheduler Jobs (1)

- **Job** – Der eigentliche Job
  - Kann mit Parametern aufgerufen werden

- **Programm** – Abstrakte Definition / Logik was in einem Job ausgeführt werden soll
  - Wie eine PL/SQL Procedure
  - Kann mit Parametern aufgerufen werden
  - Vorteil: PL/SQL Code anpassen ohne den Job neu anlegen zu müssen!

- **Schedule** – Wann und wann wieder soll der Job starten

Wichtige Begriffe zu DBMS Scheduler Jobs (2)

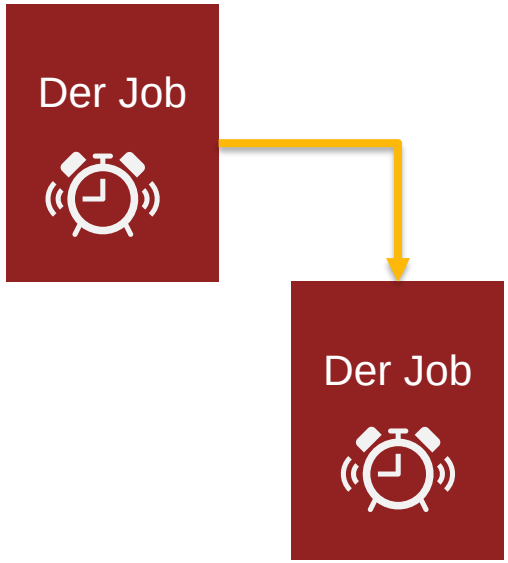
- **Job Class** – Mit welchen Eigenschaften soll der Job verknüpft werden
  - Jeder Job besitze immer eine Job Klasse - meist den Default „DEFAULT_JOB_CLASS“
  - Definiert die für den laufenden Job verwendete Ressource Consumer Group
  - Ist eine Ressource Consumer Group UND ein Service definiert, wird die Ressource Consumer Group über diese Einstellung auf der Job Class festgelegt und NICHT mehr über dem Mapping auf dem Service verändert!
  - Definiert den verwendeten DB Service - Ist dieser Service down oder nicht vorhanden wird KEIN Fehler geworfen ⇒ der Job mit der Klasse läuft einfach nicht!
  - Definiert den Log Level

Wichtige Begriffe zu DBMS Scheduler Jobs (3)

- **Window** – Zeitfenster in dem der Job laufen darf
  - Ein definiertes Zeitfenster, ist das Ende des Zeitfenster erreicht, kann je nach Bedarf der Job abgebrochen werden
  - Über den dann aktiven Ressource Plan und der Consumer Group + Service Angabe in der Job Class wird der Job auf das Fenster gebunden
- **Resource Plan** – Was darf der Job an Ressourcen in der DB beanspruchen

Wichtige Begriffe zu DBMS Scheduler Jobs (4)

- **Chain** – Kette von verknüpften Jobs
  - Zwei Jobs miteinander verknüpfen



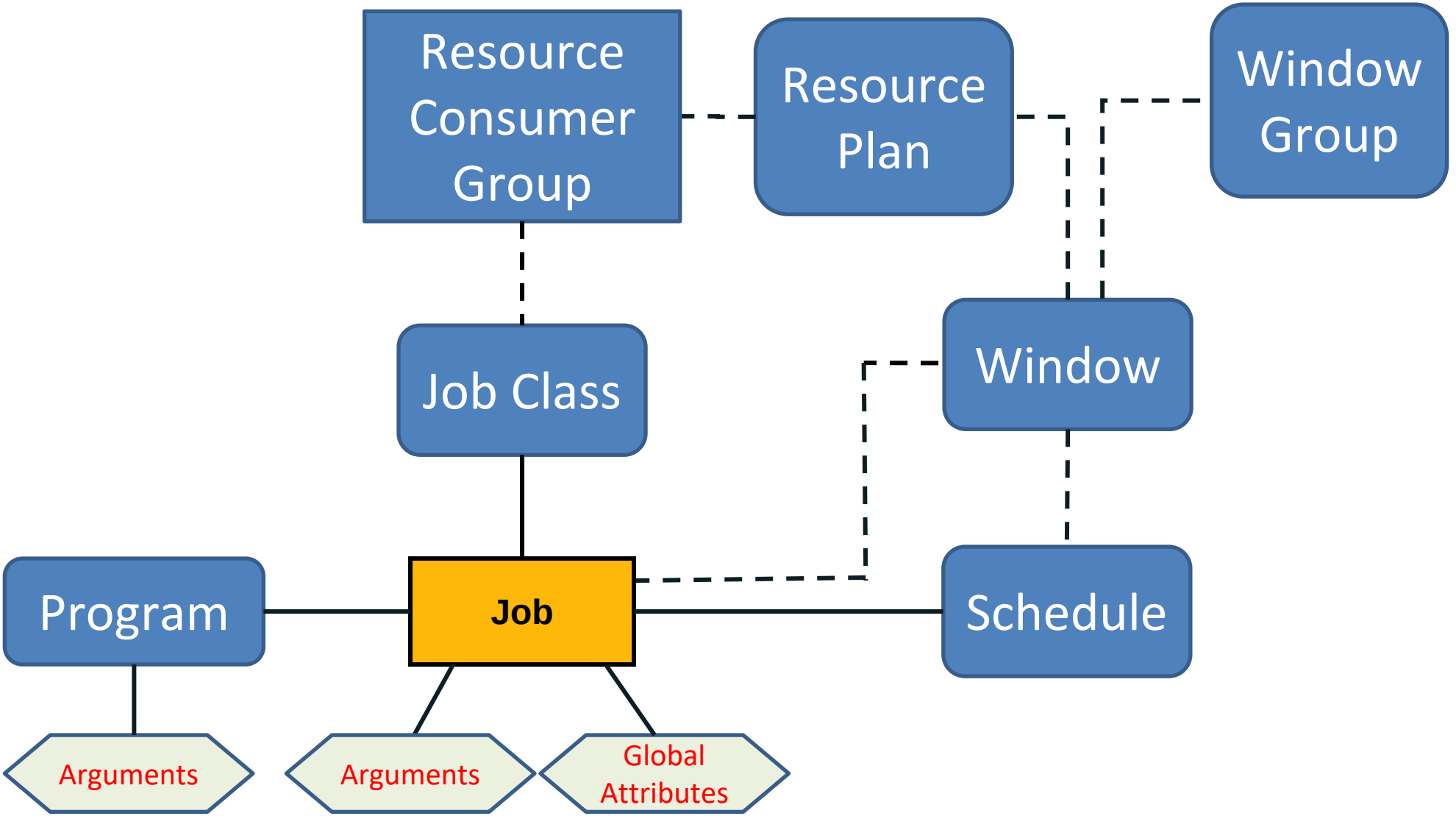
Global Attribute

- Globale Attribute für die gesamte Job Umgebung wie die aktuelle Zeitzone
- Wie:
  - DEFAULT_TIMEZONE - Zeitzone für den Job
  - EMAIL_SENDER
  - EMAIL_SERVER
  - EMAIL_SERVER_CREDENTIAL
  - EMAIL_SERVER_ENCRYPTION
  - EVENT_EXPIRY_TIME
  - FILE_WATCHER_COUNT
  - LAST_OBSERVED_EVENT
  - LOG_HISTORY
  - MAX_JOB_SLAVE_PROCESSES

Scheduler Job Typen

Normal	Lightweight Job	In Memory job
▪ Standard ▪ Full Feature Set ▪ job_style job attribute '**REGULAR**',	▪ Für “short-duration jobs”, die häufig starten ▪ Kein Schema Objekt ▪ Performanter und “platz sparender” in der Anlage ,da nicht so viele Daten im Data Dictionary hinterlegt werden müssen ▪ job_style job attribute '**LIGHTWEIGHT**‘.	▪ **In-memory runtime** Basiert auf den Ligthweight Jobs; Können ein „repeat interval“ haben und mehrfach starten ▪ Kein Logging da DEFAULT_IN_MEMORY_JOB_CLASS, auf logging level NONE. ▪ Existieren nur im Cache, nicht persistent. ▪ Nur auf der Instance auf der dieser Job angelegt wurde

Übersicht

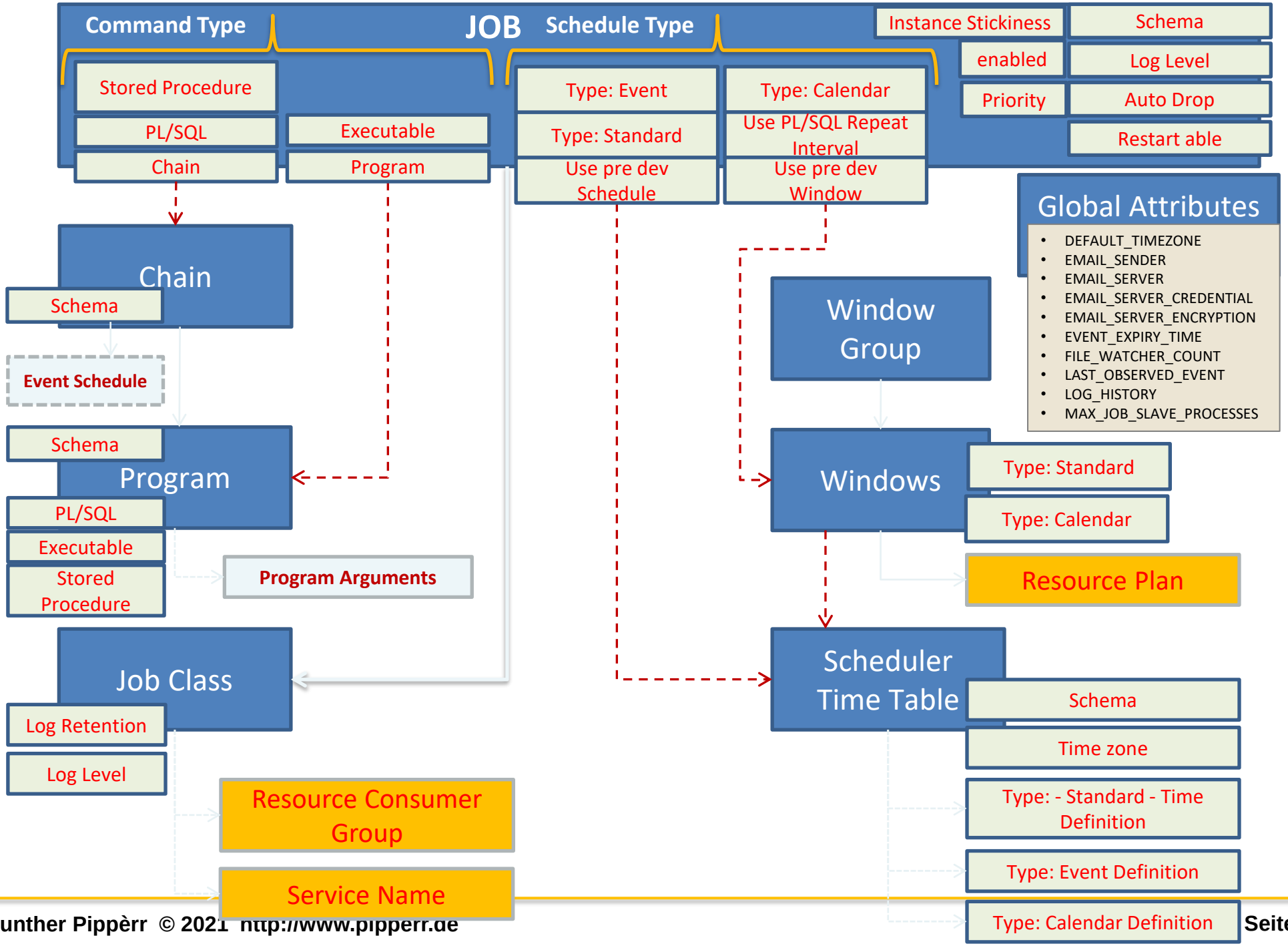


Aus https://download.oracle.com/owsf_2003/40615_goorah.ppt

Die API - DBMS_SCHEDULER

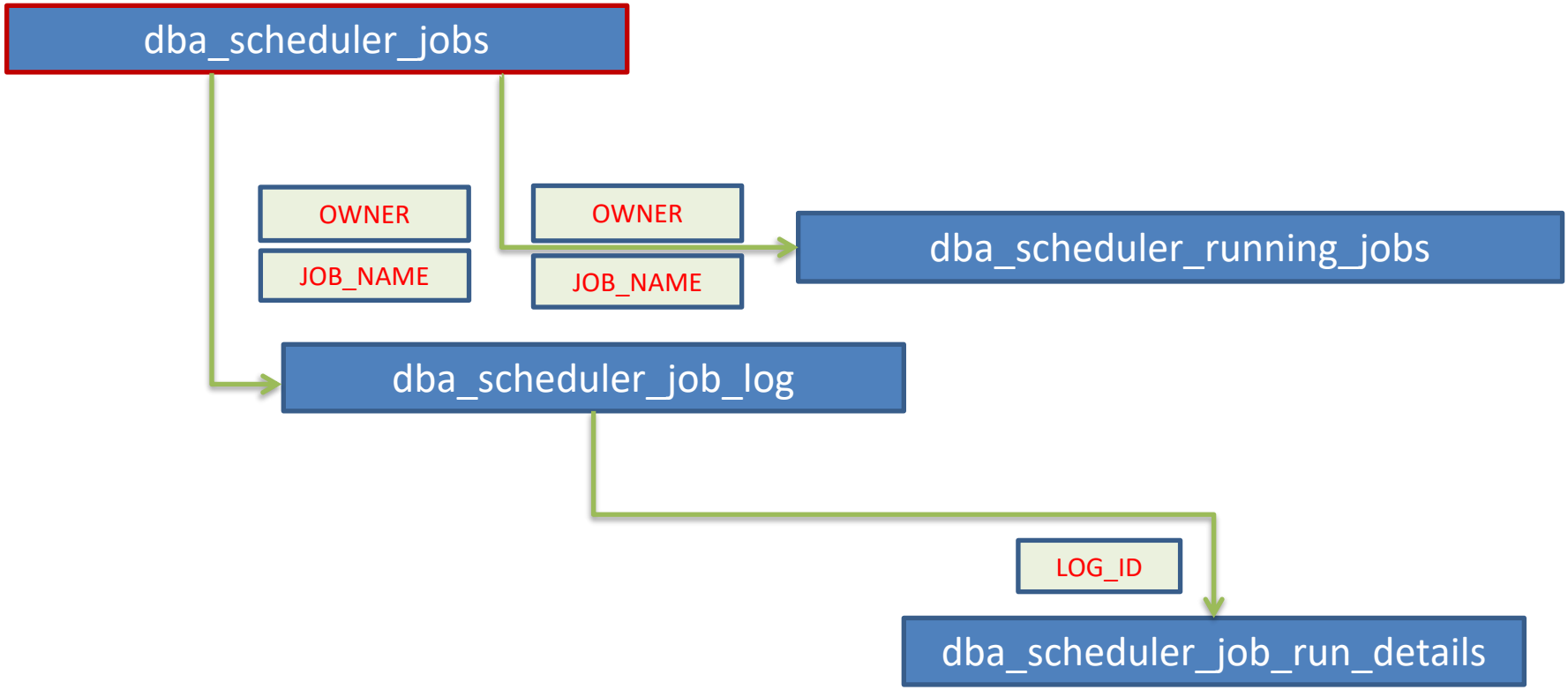
- Jobs erzeugen
  - Einfachster Aufruf

```
BEGIN
     DBMS_SCHEDULER.create_job (  job_name        => 'CLEAN_SQL_ERROR_LOG_TABLE‘
                            , job_type        => 'PLSQL_BLOCK‘
                            , job_action      => 'BEGIN system.deleteOraErrorTrigTab(15); END;‘
                            , start_date      => SYSTIMESTAMP
                            , repeat_interval => 'freq=daily; byhour=13; byminute=0‘
                            , end_date        => NULL
                            , enabled         => TRUE
                            , comments        => 'Job to clean all lean Error Log older then xx days‘);
END;
```

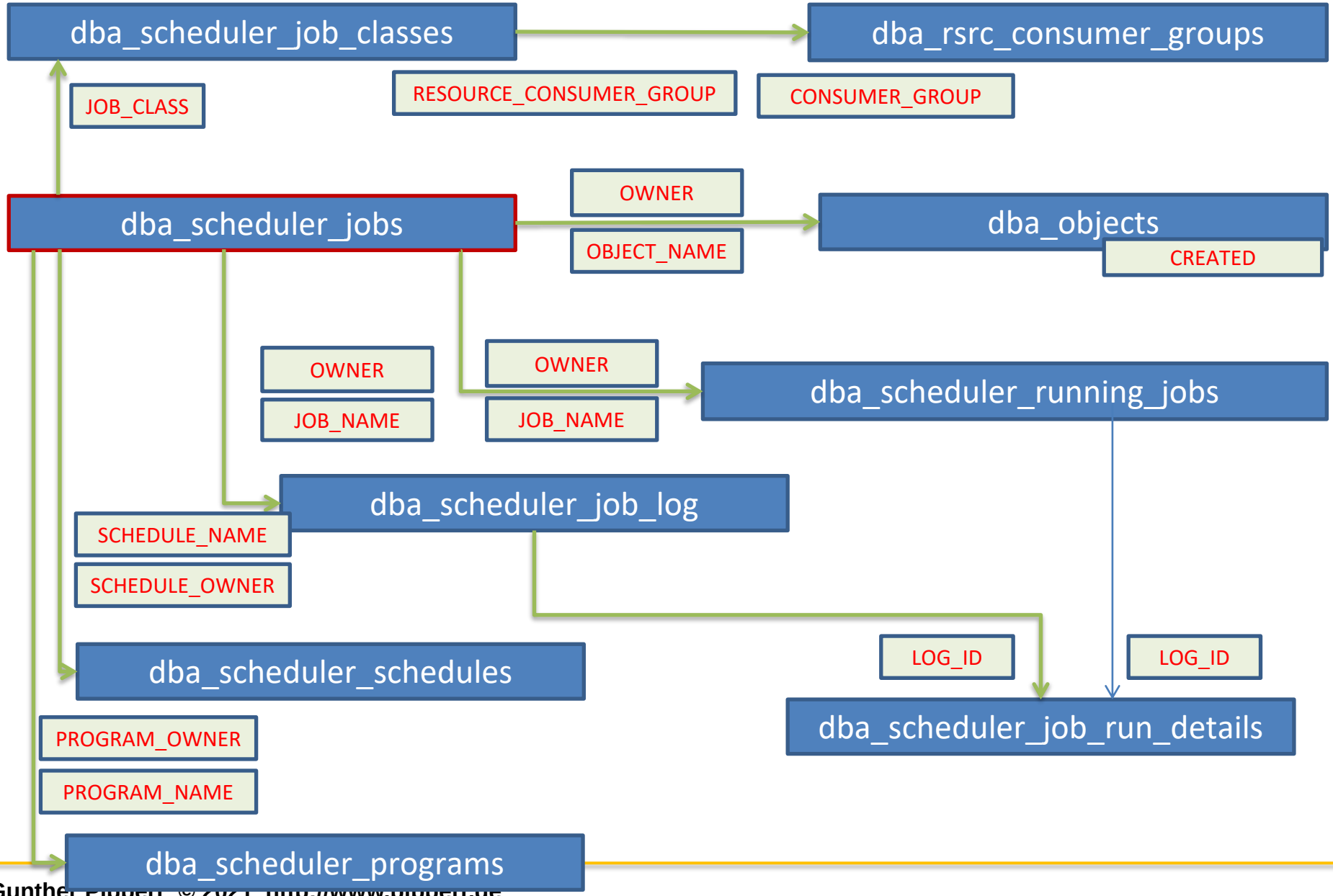

Command Type
JOB
Schedule Type
Instance Stickiness
Schema
enabled
Log Level
Priority
Auto Drop
Restart able
Stored Procedure
PL/SQL
Chain
Executable
Program
Type: Event
Type: Standard
Use pre dev Schedule
Type: Calendar
Use PL/SQL Repeat Interval
Use pre dev Window
Global Attributes
DEFAULT_TIMEZONE
EMAIL_SENDER
EMAIL_SERVER
EMAIL_SERVER_CREDENTIAL
EMAIL_SERVER_ENCRYPTION
EVENT_EXPIRY_TIME
FILE_WATCHER_COUNT
LAST_OBSERVED_EVENT
LOG_HISTORY
MAX_JOB_SLAVE_PROCESSES
Chain
Schema
Event Schedule
Window Group
Schema
Program
PL/SQL
Executable
Stored Procedure
Program Arguments
Windows
Type: Standard
Type: Calendar
Resource Plan
Job Class
Log Retention
Log Level
Resource Consumer Group
Service Name
Scheduler Time Table
Schema
Time zone
Type: - Standard - Time Definition
Type: Event Definition
Type: Calendar Definition

Scheduler Meta Data – Zentrale Views

- Was wurde wie definiert – die wichtigsten Views:



Scheduler Meta Data – Weitere Views



Scheduler Meta Data – Zentrale Views (1)

```
select js.owner
       ,  js.job_name
       ,  decode (js.state,  'SCHEDULED', 'SHUD',  'DISABLED', 'DIS',
'RUNNING', 'RUN',  js.state) as state
       ,  js.JOB_CLASS
       ,  js.program_name
       ,  js.job_action
       ,  to_char (o.CREATED, 'dd.mm.yyyy hh24:mi') as CREATED
       ,  JOB_STYLE
       ,  js.run_count
       ,  js.failure_count
       ,  to_char (js.last_start_date, 'dd.mm.yyyy hh24:mi') as
last_start_date
       ,  to_char (js.next_run_date, 'dd.mm.yyyy hh24:mi') as
next_run_date
    from dba_scheduler_jobs js, dba_objects o
   where      js.owner = o.owner(+)
         and js.job_name = o.OBJECT_NAME(+)
```

Scheduler Meta Data – Zentrale Views (2)

```
 select d.owner||'.'||d.job_name as job_name
     ,l.job_class
     ,d.log_date
     ,d.status
     ,d.error# error_number
                   ,d.errors
     ,d.actual_start_date
     ,extract (second from d.cpu_used)
                       + (  extract (minute from d.cpu_used)
                          * 60)
                       + (  extract (hour from d.cpu_used)
                          * 60
                          * 60)  as timeused
     ,l.additional_info
  from dba_scheduler_job_run_details d
   right join dba_scheduler_job_log l on (d.log_id = l.log_id)
;
```

Job Log File Pflege

- Wie voll ist das Log? Wie alt sind die Daten?

```
SELECT COUNT(*)       AS count_log
      , MIN(log_date) AS first_log
      , MAX(log_date) AS last_log
      , trunc(MAX(log_date)) -trunc(MIN(log_date))  AS  log_tage
   FROM dba_scheduler_job_log
/

   COUNT_LOG  FIRST_LOG   LAST_LOG    LOG_TAGE
------------  ---------   ----------  -----------
       67538  04-NOV-16   01-SEP-20   1397
```

- Einstellen mit:

```
SYS@GPI> EXEC DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE('log_history','10');
```

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

Job Log File Pflege – Problem – Lösch Job prüfen

- Job “PURGE_LOG” prüfen!

```
  SELECT js.owner
       ,  js.job_name
       ,  decode (js.state,  'SCHEDULED', 'SHUD',  'DISABLED', 'DIS',  'RUNNING', 'RUN',  js.state) AS state
       ,  js.JOB_CLASS
       ,  js.program_name
       ,  js.job_action
       ,  to_char (o.CREATED, 'dd.mm.yyyy hh24:mi') AS CREATED
       ,  js.run_count
       ,  js.failure_count
       ,  to_char (js.last_start_date, 'dd.mm hh24:mi') AS last_start_date
       ,  to_char (js.next_run_date, 'dd.mm hh24:mi') AS next_run_date
    FROM dba_scheduler_jobs js, dba_objects o
   WHERE      js.owner = o.owner(+)
         AND js.job_name = o.OBJECT_NAME(+)
         AND js.job_name LIKE 'PURGE_LOG'
ORDER BY owner, job_name
/
```

Job Log File Pflege – Problem

- Globale Einstellung wird von Job Class überschrieben

```
SELECT owner
     , job_class_name
     , logging_level
     , log_history
     , comments
 FROM DBA_SCHEDULER_JOB_CLASSES
ORDER BY  log_history
/
```

  - => Default Wert oft 10.000!
    - die Daten von heute werden dann am 30.07.4758 gelöscht; Wohl etwas spät …

- Einstellen mit

```
EXEC DBMS_SCHEDULER.SET_ATTRIBUTE('ORA$AT_JCNRM_OS','log_history','30');
```

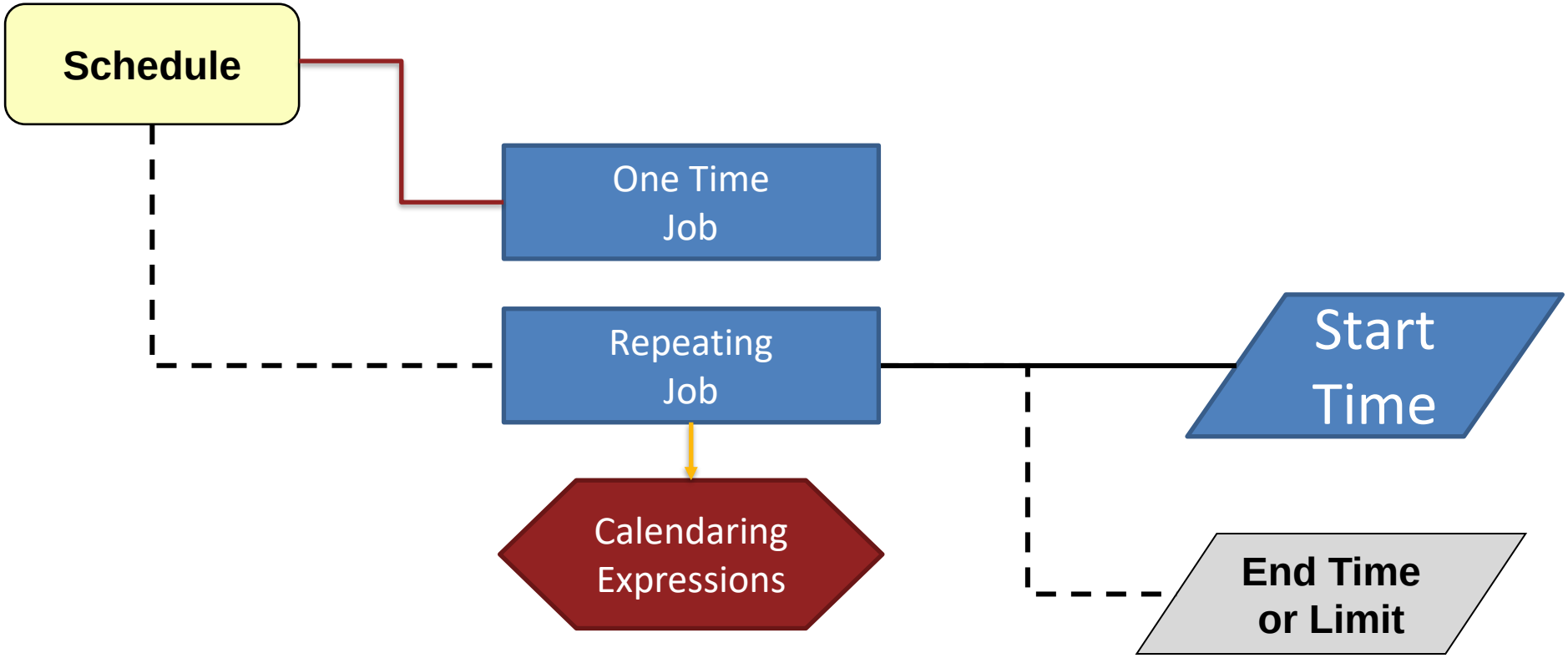
Das Problem mit dem Job Log

- Nur ein “lebender” Job kann noch etwas ins Log schreiben!
  - In vielen Situationen keine Log Information!
    - Wie ORA-600 / DB Instance gestoppt / Job vom DBA abgeschoßen etc.
  - Auf die Job Klasse achten ob Loging überhaupt aktiviert ist!

- Lösung: Zweiter Job überwacht ersten Job und alarmiert!

Ein Schedule definieren (1)

- **Schedule** - Festlegen wann der Job laufen soll und wann er erneut starten muss

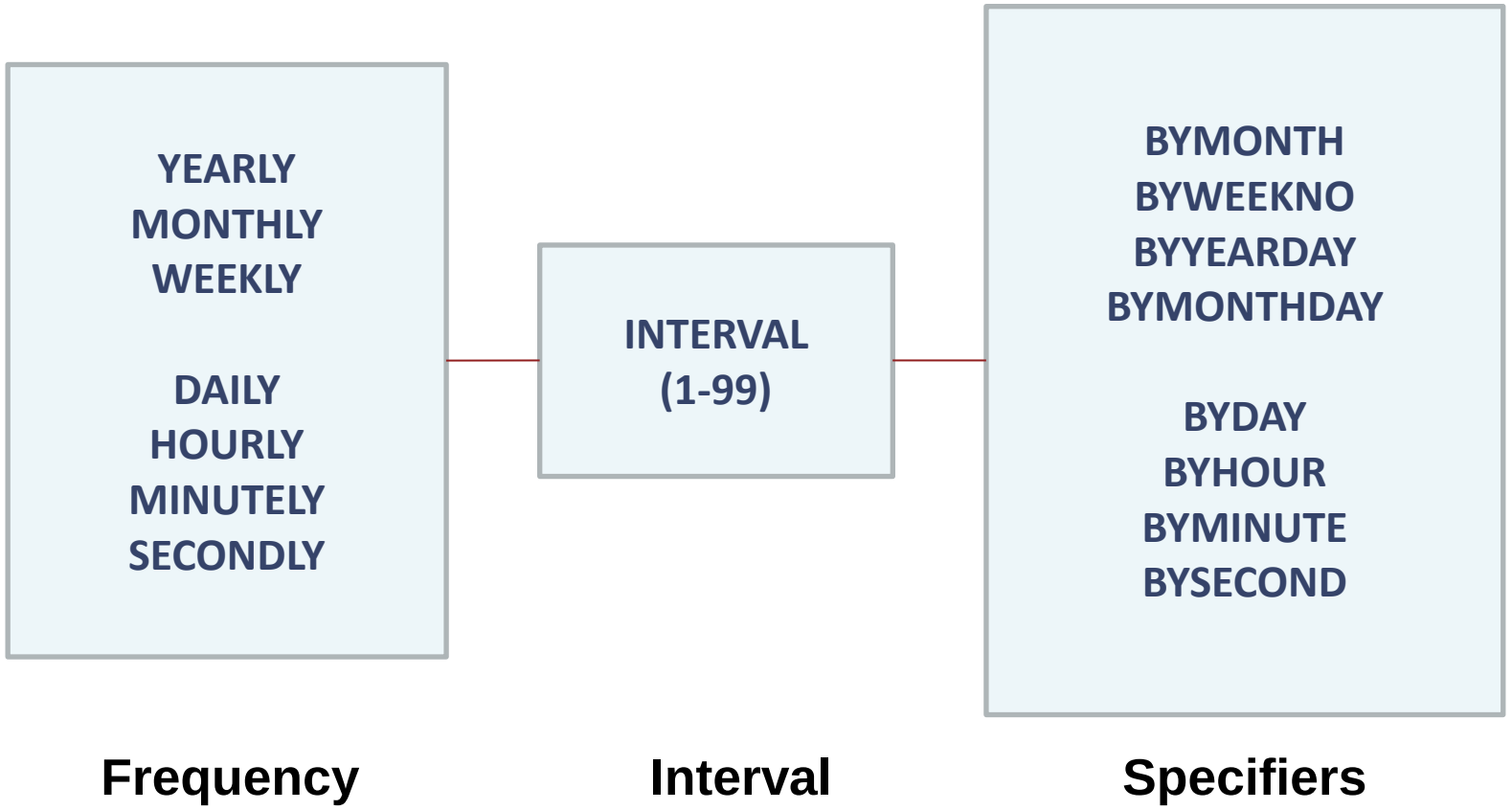


https://www.pipperr.de/dokuwiki/doku.php?id=dba:oracle_scheduler_repeat_interval

Ein Schedule definieren (2)

Calendaring Expressions

- Calendaring Expressions



Ein Schedule definieren (3)

- Calendaring Expressions – Beispiele

  - Every hour: FREQ=HOURLY; INTERVAL=1
  - Every two weeks: FREQ=WEEKLY; INTERVAL=2
  - Every 30 days: FREQ=DAILY; INTERVAL=30
  - Every 5 minutes: FREQ=MINUTELY; INTERVAL=5
  - Every second: FREQ=SECONDLY

  - Jeden 15th im Monat: FREQ=MONTHLY; BYMONTHDAY=15
  - Jede 4. Woche am Montag: FREQ=YEARLY; BYWEEKNO=4,8,12,16,20,24,28,32,36,40,44,48,52; BYDAY=MON
  - 14:30 jeden Dienstag: FREQ=WEEKLY; BYDAY=TUE; BYHOUR=14; BYMINUTE=30

(Aus der Oracle Dokumentation)

Eine "Calendaring Expressions" testen

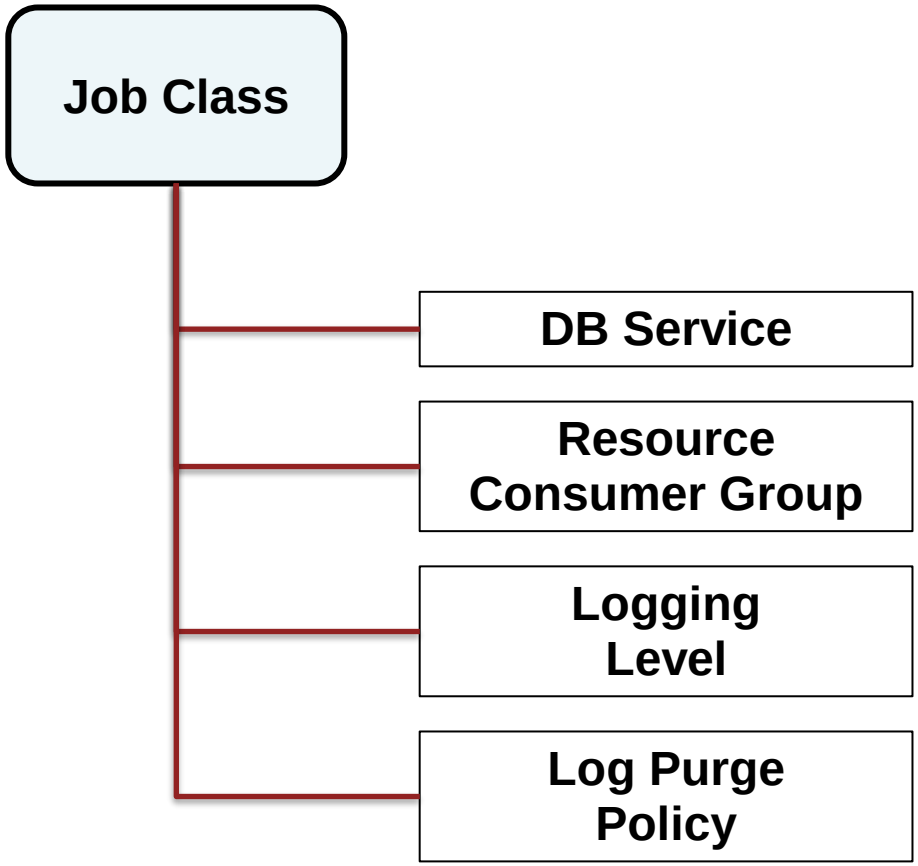
- DBMS_SCHEDULER EVALUATE_CALENDAR_STRING

```
DECLARE
        v_next_run_date TIMESTAMP;
        v_start_date    TIMESTAMP:=systimestamp;
        v_return_date_after TIMESTAMP
BEGIN
  FOR i IN 1 ..10
  loop
  DBMS_SCHEDULER.EVALUATE_CALENDAR_STRING(
                 calendar_string   => 'FREQ=MINUTELY; INTERVAL=15'
               , start_date        => v_start_date
               , return_date_after => v_return_date_after
               , next_run_date     => v_next_run_date);

        DBMS_OUTPUT.PUT_LINE('-- Info actual date :: '||to_char(v_start_date,'dd.mm.yyyy hh24:mi')||
                             '   --> next_run_date:: '||to_char(v_next_run_date,'dd.mm.yyyy hh24:mi')
        );
         v_return_date_after := v_next_run_date;
  END loop;
END;
/
```

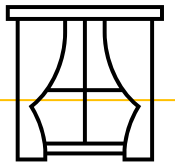
Job Class

- Die Elemente der Job Class

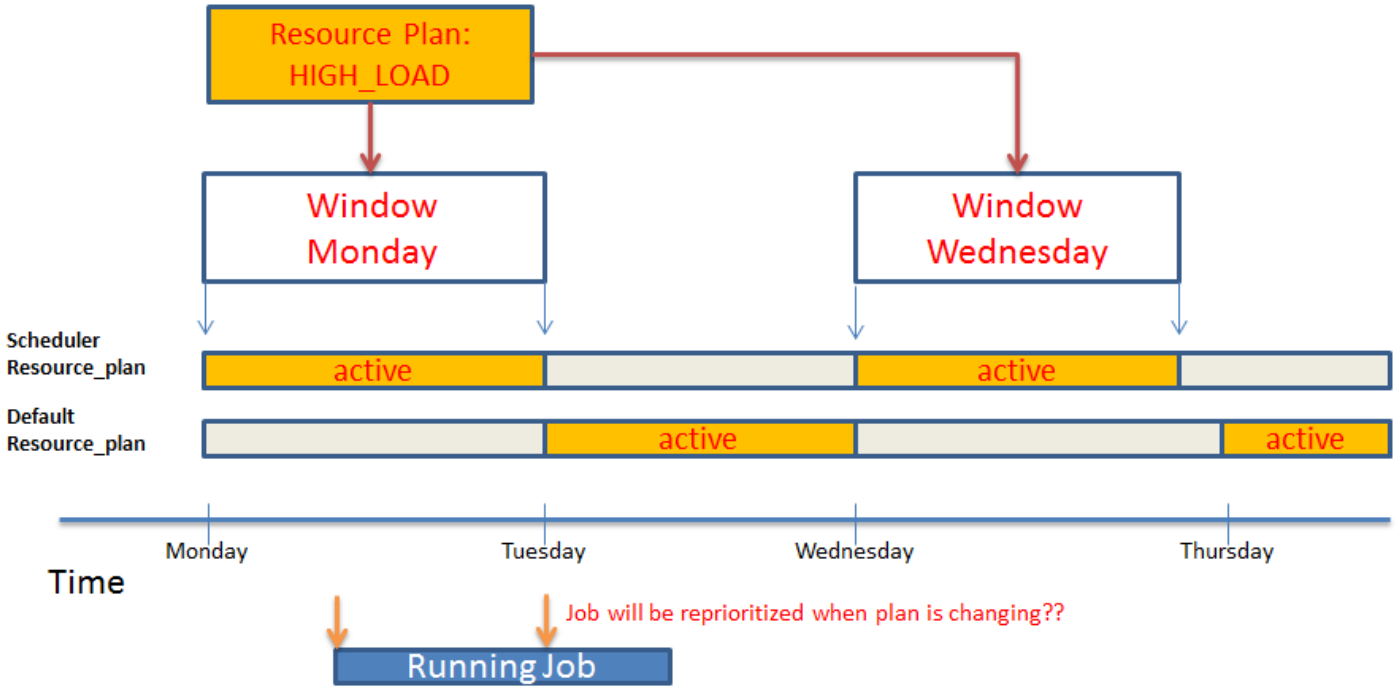



```
begin
DBMS_SCHEDULER.CREATE_JOB_CLASS(
      job_class_name          => 'ADMIN_JOBS'
    , resource_consumer_group => 'ADMIN_JOBS'
    , service                 => 'ADMIN_SRV'
);
End;
/
```

Die Window Funktionalität



- Job Lauf auf Zeiträume einschränken



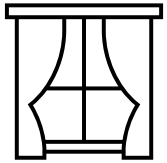
Job priority/resource usage will be defined by:

A) Over the Consumer Group Mappings Priority - only if NO Consumer Group is defined:

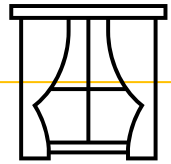
- **DB Service**
- **DB User to primary consumer group relation**

B) The settings of the Resource Consumer Group

Meta Data - Windows



Plan zu Job Fenster



- Zu welchem Plan ist ein Job Fenster zugeordnet:

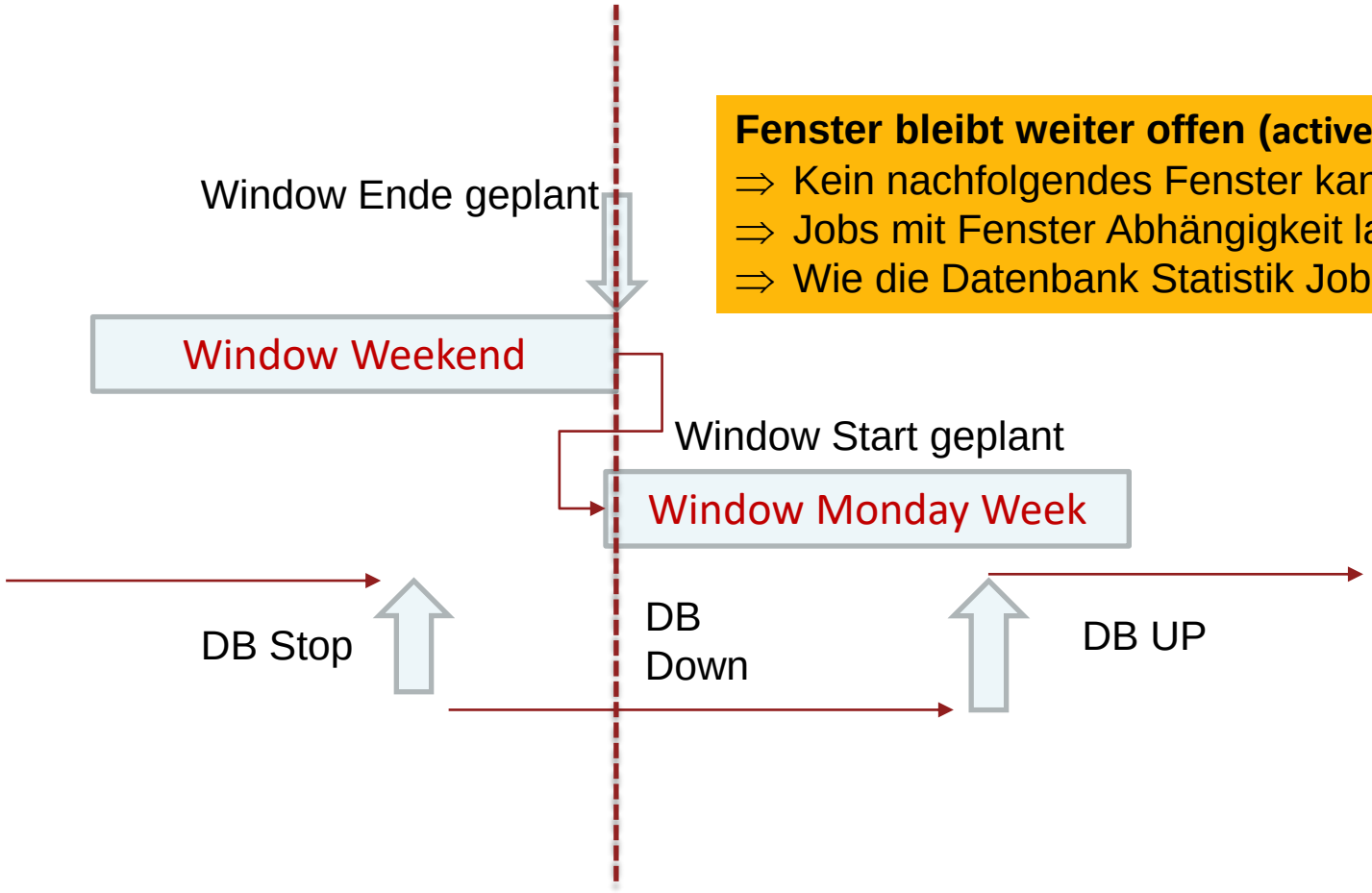
```
COLUMN job_class_name    format a25
COLUMN job_class_service format a25
COLUMN resource_plan     format a28
COLUMN consumer_group    format a25
COLUMN job_window        format a20


SELECT jc.job_class_name
     , jc.service AS job_class_service
     , cg.consumer_group
     , pd.plan     AS resource_plan
     , sw.window_name AS job_window
  FROM dba_scheduler_job_classes jc
      , dba_rsrc_consumer_groups cg
      , dba_rsrc_plan_directives pd
      , dba_scheduler_windows sw
      , DBA_RSRC_GROUP_MAPPINGS gm
  WHERE jc.resource_consumer_group = cg.consumer_group
    AND cg.consumer_group=pd.group_or_subplan (+)
    AND sw.resource_plan(+)=pd.group_or_subplan
 ORDER BY 1
/
```

Über den dann aktiven Ressource Plan und der Consumer Group + Service Angabe in der Job Class wird der Job auf das Fenster gebunden

Was passiert wenn ein Window offen bleibt?

- Szenario: Datenbank wird gestoppt und dann ein paar Stunden später wieder gestartet



Fenster bleibt weiter offen (active)
⇒ Kein nachfolgendes Fenster kann gestartet werden
⇒ Jobs mit Fenster Abhängigkeit laufen nun nicht mehr!
⇒ Wie die Datenbank Statistik Jobs!

Prüfen mit:

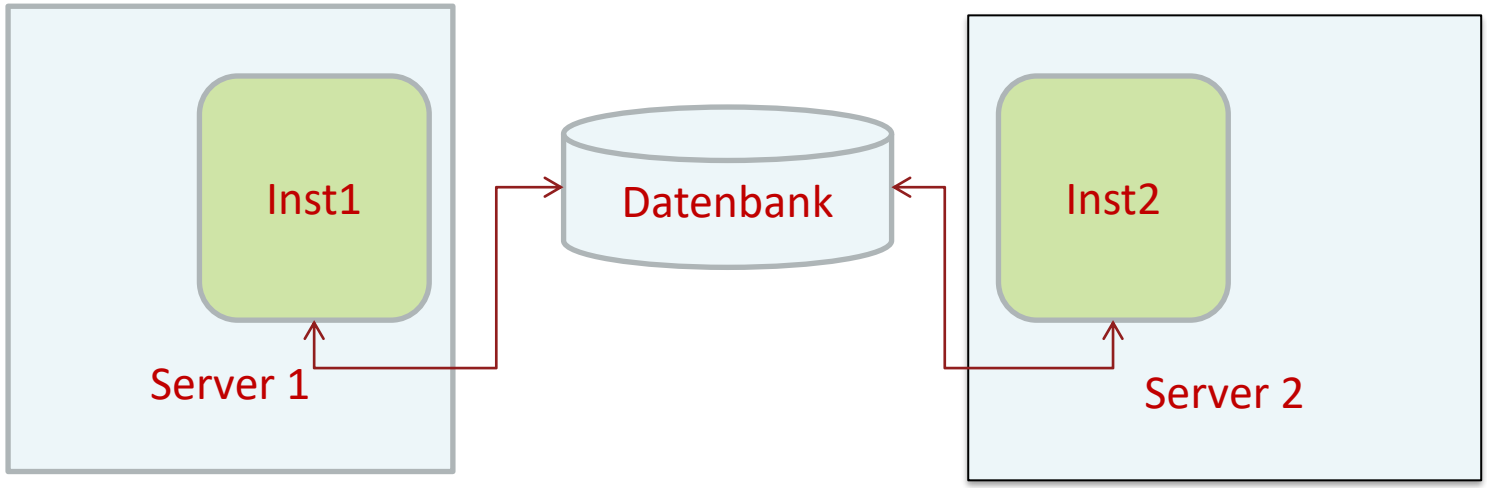
- Nach längeren Downtimes prüfen:

```
column check_active format a10    heading 'Check|if ok’

select window_name
    ,  to_char (last_start_date, 'DD.MM.YYYY HH24:MI') as last_start_date
    ,  enabled
    ,  active
    ,  decode (active, 'TRUE', '<==CHECK IF POSSIBLE', '-') as check_active
 from dba_scheduler_windows
order by last_start_date
/
```

Real Applikation Cluster und Jobs

- Wo läuft der Job am Ende?



In einer Cluster Umgebung kann definiert werden auf welchem Knoten der Job laufen soll.

1. Fest auf eine Instance binden – Instance ID setzen
2. Letzte Instance verwenden auf dem der Job mal lief - Stickiness => True
3. Job Klasse und Service verwenden (am saubersten!)

RAC – Jobausführung per Service steuern

- **Job Klasse und Service**
  - Einen Service im Cluster anlegen der NUR auf der gewünschten Instance läuft
  - Eine Job Klasse anlegen, die diesen Service verwendet
  - Dem Job diese Job Klasse zuordnen

RAC - Jobs – Auf Instance binden

- **Job auf eine Instance binden**
  - Nachteil: Falls die Instance down ist, läuft der Job aber auch nicht!
  - Attribut setzen mit:

```
begin
 dbms_scheduler.set_attribute(  name        => 'GPI.DO_MY_JOB‘
                              , attribute   => 'INSTANCE_ID‘
                              , value       => 3 );
end;
/
```

RAC - Jobs – Instance Stickiness

- **Schalter Instance Stickiness => True**
  - Wenn „Instance Stickiness“ auf „TRUE“
  - dann versucht die DB den Job auf der Instance wieder zu starten, auf der zuletzt der Job lief
  - ABER nur wenn diese Instance auch die Instance mit der aktuell geringsten Last ist!
  - D.h. In einem System mit stark wechselnder Last kann man sich nicht darauf verlassen, dass der Job wirklich jedes Mal auf derselben Instance startet!

File Watcher Job - Übersicht

- SYS - Scheduler Credential anlegen
  - Execute granten
- SYS - File Watcher anlegen
  - Execute granten an USER
- USER - Scheduler Programm für den eigentlichen Task anlegen
- SYS - Event Based Job für den User mit Referenz auf den File Watcher anlegen
- Alles aktiveren
- Alle (default) 10 Minuten prüft die Datenbank das Verzeichnis

Beispiel: https://www.pipperr.de/dokuwiki/doku.php?id=dba:oracle_scheduler_file_watcher_19c

File Watcher - Credential

- Credential anlegen

```
BEGIN
  DBMS_SCHEDULER.CREATE_CREDENTIAL(
        FILE_WATCH_CRED’
      , 'os_user'
      , 'xxxgeheimxxxx');
END;
/
```

- Granten an das Schema das den Job ausführt

```
GRANT EXECUTE ON WATCH_CREDENTIAL to job_control;
```

File Watcher - File Watcher anlegen

- Watcher anlegen (auf das richtige Verzeichnis achten!)

```
BEGIN
  DBMS_SCHEDULER.CREATE_FILE_WATCHER(
          FILE_WATCHER_NAME  => 'IMPORT_FILE_WATCHER'
        ,  DIRECTORY_PATH     => '/srv/job_control'
        ,  FILE_NAME          => 'import*.csv'
        ,  CREDENTIAL_NAME    => 'FILE_WATCH_CRED'
        ,  DESTINATION        => NULL
        ,  ENABLED            => FALSE
  );
END;
/
```

- Granten an das Schema das den Job ausführt

```
GRANT EXECUTE ON IMPORT_FILE_WATCHER to job_control;
```

Anlage abfragen über **dba_scheduler_file_watchers**

Parameterisieren

- ? = im Pfad => Oracle Home als Basis Pfad verwenden
- MIN_FILE_SIZE = (Bytes) erst reagieren, wenn die Datei eine bestimmte größe erreicht hat
- STEADY_STATE_DURATION = (Datentyp Zeit „INTERVAL DAY TO SECOND“ , max. 1h, min 10s) in der die Daten unverändert sein müssen
  - Das Intervall für den DB File Watcher Job muss in Relation dazu definiert werden, damit überhaupt das richtig ermittelt werden kann! D.h. etwas kleiner als STEADY_STATE_DURATION!

File Watcher – Eigentlichen Task anlegen (1)

- Programm anlegen

```
BEGIN
  DBMS_SCHEDULER.create_program(
    program_name        => 'CSV_INTERFACE_FILE_IMPORT_PROG',
    program_type        => 'STORED_PROCEDURE',
    program_action      => 'CSV_INTERFACE_FILE_IMPORT',
    number_of_arguments => 1,
    enabled             => FALSE);
END;
/
```

- Metadata Argument für “event_message” anlegen

```
BEGIN
  DBMS_SCHEDULER.define_metadata_argument(
    program_name       => 'CSV_INTERFACE_FILE_IMPORT_PROG',
    metadata_attribute => 'event_message',
    argument_position  => 1);
EN
```

File Watcher – Eigentlichen Task anlegen (2)

- Prozedur für die eigentliche Aufgabe anlegen

```
CREATE OR REPLACE PROCEDURE CSV_INTERFACE_FILE_IMPORT (p_file_watch_result
SYS.SCHEDULER_FILEWATCHER_RESULT)
 AS
  v_message CSV_IMPORT_FILE_WATCH_LOG.message%TYPE;
BEGIN
  v_message :=   'Pfad  :' || p_file_watch_result.directory_path
               ||'-Name  :'|| p_file_watch_result.actual_file_name
                           ||'-Size  :'|| p_file_watch_result.file_size
                           ||'-Date  :'|| p_file_watch_result.file_timestamp
                           ||'-Epoch :'|| p_file_watch_result.ts_ms_from_epoch;

  INSERT INTO CSV_IMPORT_FILE_WATCH_LOG (message)
  VALUES (v_message);
  COMMIT;
END;
/
```

File Watcher – Job anlegen

- Der eigentliche Job

```
BEGIN
  DBMS_SCHEDULER.create_job(
    job_name        => 'job_control.CSV_INTERFACE_FILE_IMPORT_JOB',
    program_name    => 'job_control.CSV_INTERFACE_FILE_IMPORT_PROG',
    event_condition => NULL,
    queue_spec      => 'IMPORT_FILE_WATCHER',
    auto_drop       => FALSE,
    enabled         => FALSE);
END;
/
```

- Bei Bedarf mehrfach ausführen

```
BEGIN
 DBMS_SCHEDULER.set_attribute('CSV_INTERFACE_FILE_IMPORT_JOB‘
                              ,'PARALLEL_INSTANCES',TRUE);
END;
/
```

File Watcher - Aktivieren und kontrollieren

- Einschalten/aktivieren

```
EXEC DBMS_SCHEDULER.enable('IMPORT_FILE_WATCHER');
EXEC DBMS_SCHEDULER.enable('CSV_INTERFACE_FILE_IMPORT_PROG');
EXEC DBMS_SCHEDULER.enable('CSV_INTERFACE_FILE_IMPORT_JOB');
```

- Kontrollieren:

```
SELECT file_watcher_name
      , destination
      , directory_path
      , file_name
      , credential_name
 FROM dba_scheduler_file_watchers;
```

See => https://aychin.wordpress.com/2014/08/21/oracle-file_watcher-and-its-troubleshooting/

Filewatcher Debug (1)

- Queue testen

```
select owner, queue_table, qid, enqueue_enabled, dequeue_enabled from dba_queues where
name='SCHEDULER_FILEWATCHER_Q';
```

- Basis Routine aufrufen und Trace File auf Auffälligkeiten prüfen

```
oradebug setmypid
oradebug tracefile_name

alter system set events '27401 trace name context forever, level 262144';

begin
 dbms_ischedfw.file_watch_job;
end;
/
```

```
file_watcher:: 2021-05-16 21:52:10.680: LastChkTime for directory /srv/job_control changed to 1621194686000
file_watcher:: 2021-05-16 21:52:10.682: All checking done
file_watcher:: 2021-05-16 21:52:10.683: Setting up results information
file_watcher:: 2021-05-16 21:52:10.684: Directory: /srv/job_control
file_watcher:: 2021-05-16 21:52:10.685: Result file name: import23.csv
file_watcher:: 2021-05-16 21:52:10.687: Request pattern: import*.csv, Regex: import.*.csv
file_watcher:: 2021-05-16 21:52:10.688: Match found, adding to list of matching requests
file_watcher:: 2021-05-16 21:52:10.690: File watching done
file_watcher:: 2021-05-16 21:52:10.709: Updating history
file_watcher:: 2021-05-16 21:52:10.710: Iteration 1
file_watcher:: 2021-05-16 21:52:10.710: Updating history entry
file_watcher:: 2021-05-16 21:52:10.710:   Dir Path: /srv/job_control
file_watcher:: 2021-05-16 21:52:10.710:   Last Chk Time: 16-MAY-21 07.51.26.000000 PM +00:00
file_watcher:: 2021-05-16 21:52:10.710: Processing results
file_watcher:: 2021-05-16 21:52:10.711: Iteration 1
file_watcher:: 2021-05-16 21:52:10.711:   Dir Path: /srv/job_control
file_watcher:: 2021-05-16 21:52:10.711:   File Name: import23.csv
file_watcher:: 2021-05-16 21:52:10.711:   File Size: 22258
file_watcher:: 2021-05-16 21:52:10.711:   File Tstamp: 16-MAY-21 07.51.26.000000 PM +00:00
file_watcher:: 2021-05-16 21:52:10.711:   Matching Requests:
file_watcher:: 2021-05-16 21:52:10.711:     Request: SYS.IMPORT_FILE_WATCHER
SCHED 05-16 21:52:10.713 1 00 51129 ora 0(sjssuWriteJssuFlags):Using Flags 9
```

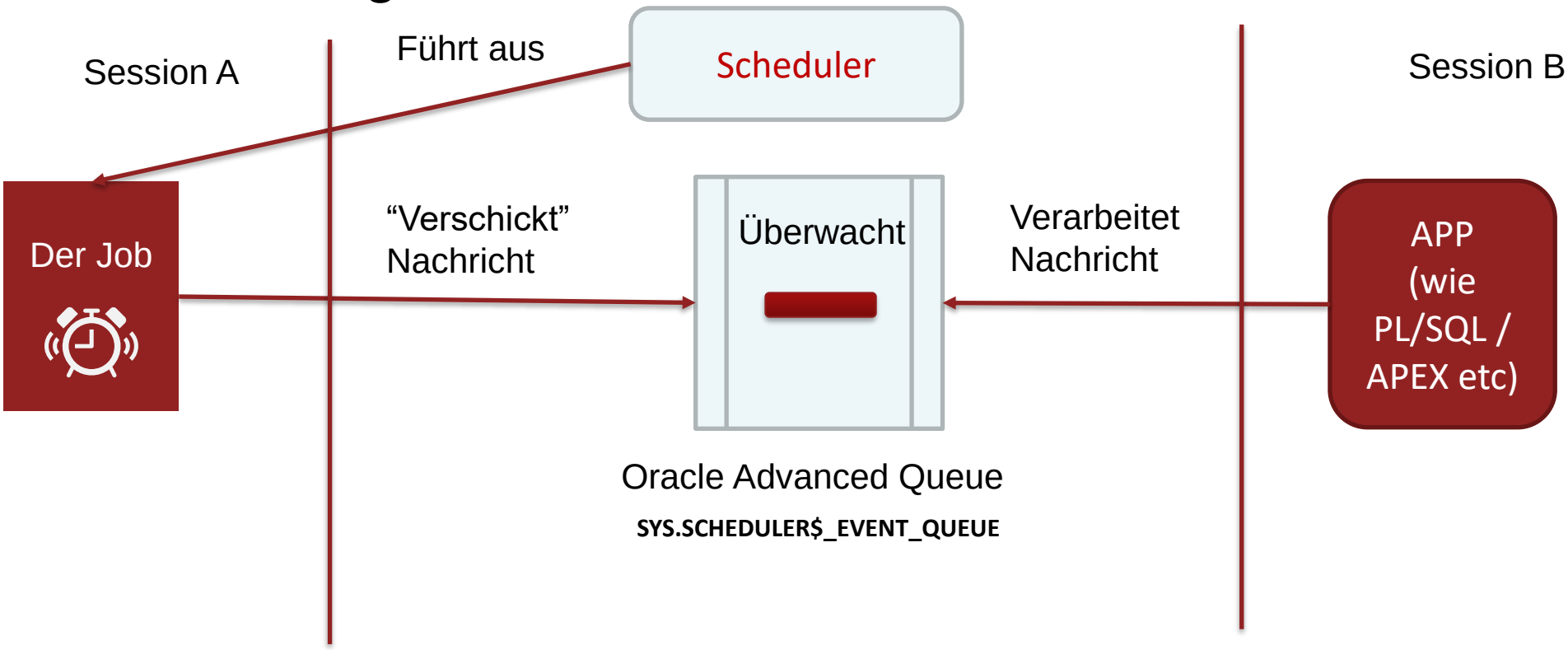
Filewatcher Debug (2)

- Letzter Aufruf

```
SELECT DIRECTORY_PATH
     , LAST_CHECK_TIME
  FROM SYS.SCHEDULER$_FILEWATCHER_HISTORY;
```

"Event Based Job" – Job sendet Nachricht

- Job erzeugt Events

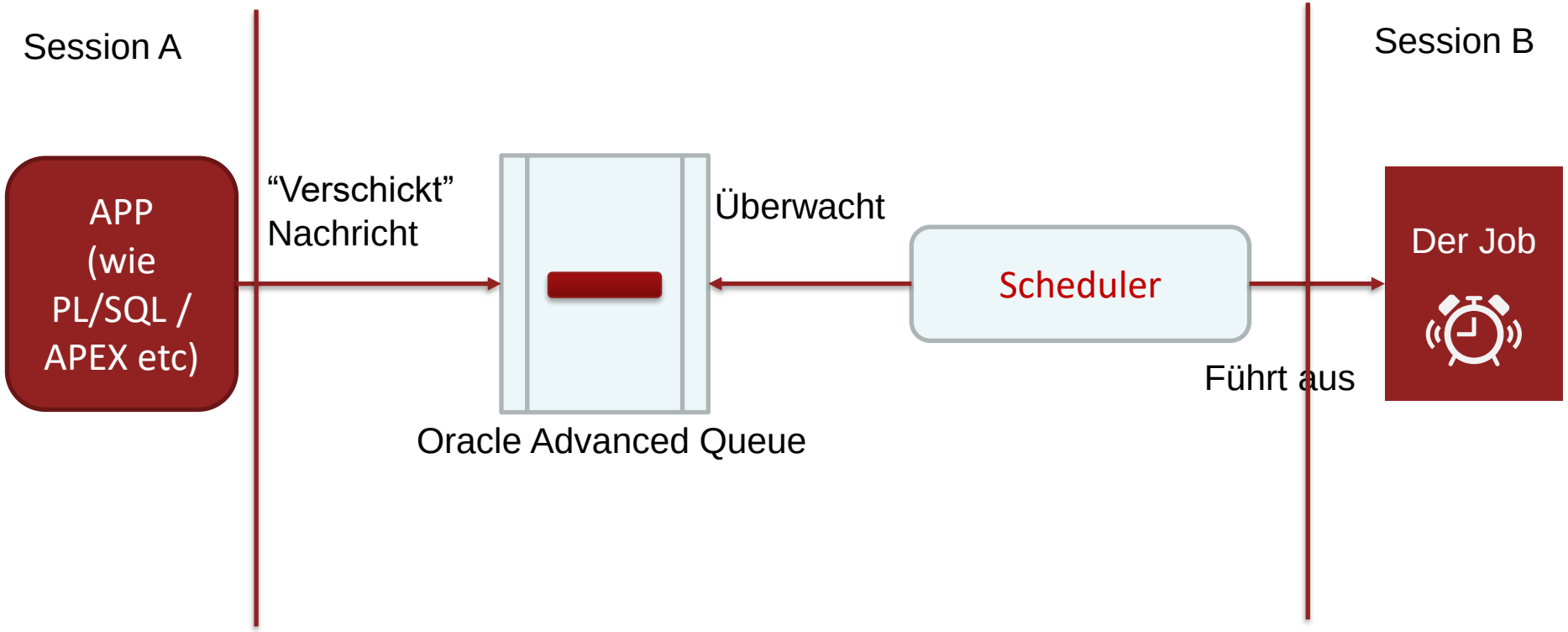


Szenario: Job benachrichtigt Application über Fehler

Seit Oracle 10g Database Release 2

"Event Based Job" – Job wird gestartet

- Aus der Applikation über “Events” einen Job triggern



Seit Oracle 10g Database Release 2

Szenario: Parallel zur "Applikations Session Thread" eine weitere Session starten

How to Build up an Event Based Job - An Example (Doc ID 1431664.1)

E-Mail Benachrichtigung aktiveren

- Eine Änderung des Job Status triggert die Versendung der Mail
  - Zuvor das Versenden von Mails mit ACL etc. aktivieren
  - Parameter definieren

```
BEGIN
  DBMS_SCHEDULER.set_scheduler_attribute('email_server', 'postfix.pipperr.local:25');
  DBMS_SCHEDULER.set_scheduler_attribute('email_sender', 'database@pipperr.de');
END;
/
```

  - An bestehenden Job “anhängen”

```
BEGIN
 DBMS_SCHEDULER.add_job_email_notification (
  job_name         =>  ‘DATA_IMPORT_RUN',
  recipients       =>  ‘gunther@pipperr.de',
  events           =>  'job_failed',
  filter_condition => ':event.error_code=600');
END;
/
```

Event wie.:job_started, job_succeeded , job_failed

Zusätzlich filtern

  - Überwachen über DBA_SCHEDULER_NOTIFICATIONS

Summary

Fazit „DB Jobs“

- Der Oracle Scheduler
  - Etwas komplexer in der Verwendung als DBA_JOBS
  - Aber sehr mächtig!
  - Vor einen Migration auf Oracle 19c bereits auf Scheduler Jobs umstellen
- APEX Automation
  - Jobs einfach in die eigene APP integrieren

Mehr

- Quellen
  - https://download.oracle.com/owsf_2003/40615_goorah.ppt

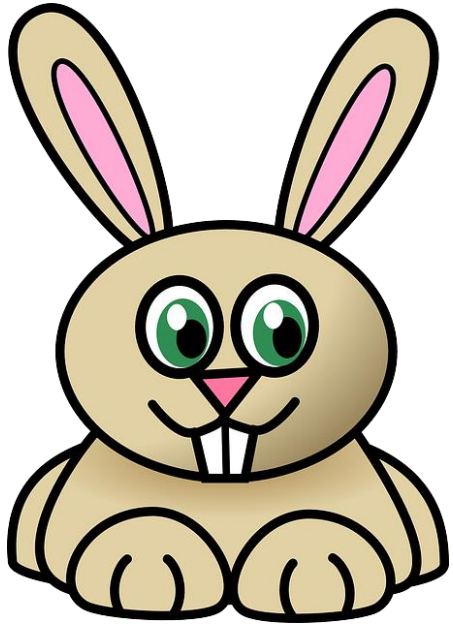
- Blog Gunther Pippèrr https://www.pipperr.de/dokuwiki/doku.php?id=dba:oracle_scheduler
- Wieder mal eine andere Skript Library
  - https://github.com/gpipperr/OraPowerShell

- Bildmaterial : https://pixabay.com

Source Code: https://github.com/gpipperr/APEX_CONNECT_2021_JOB_AUTOMATION

Diskussion

Fragen ?



F&A

Fragen

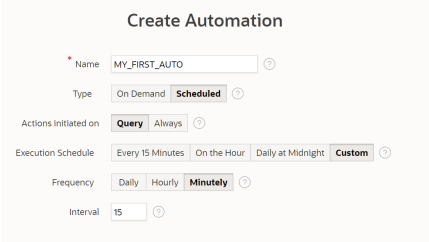
APEX Job Steuerung

APEX Automations Feature

APEX's eigener "einfacher" Job Scheduler

Declarative asynchronous processes

APEX Automations



https://www.pipperr.de/dokuwiki/doku.php?id=prog:apex_automations

APEX Automations Feature - Idee

- Logik im Hintergrund der Applikation bei Bedarf ausführen lassen

- Typische Business Cases
  - Workflows an starten, wenn bestimmte Daten in der Datenbank vorliegen
    - **Wie:** Sachbearbeiter A hat Vorgang bearbeitet und abgeschlossen, Kunde B erhält automatisch eine Mail, dass der Vorgang bearbeitet wurde
    - **Wie**: Bei Fehlern im Log Admin per Mail benachrichtigen
  - Regelmäßig im Hintergrund etwas starten
    - **Wie**: Logs aufräumen

APEX Automations Feature

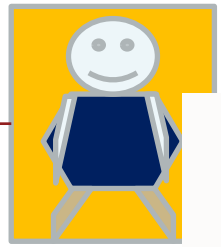
- Mit APEX Automations werden PL/SQL Aktionen definiert

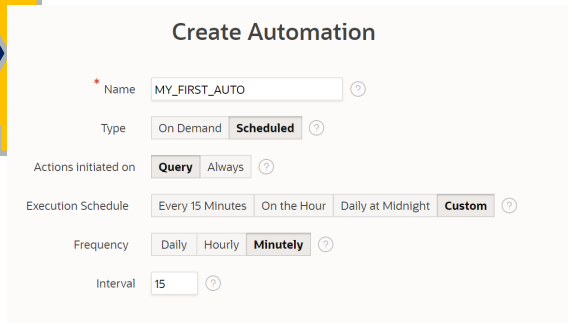
- Diese Aktion kann in Abhängigkeit von den Ergebnissen einer Query ausgeführt werden
  - Die Ergebnisse der Query können mit in der Aktion verarbeitet werden
  - SQL Query (Local Database or REST Enabled SQL)
    - Query kann auch mit einer „PL/SQL function returning SQL Query“ definiert werden
  - REST Data Source

- Eine vorab definierte Automation kann manuell über das APEX_AUTOMATION Package aufgerufen werden

Übersicht APEX Automations

Meta Daten über Oberfläche eintragen





WWV_FLOW_AUTOMATIONS

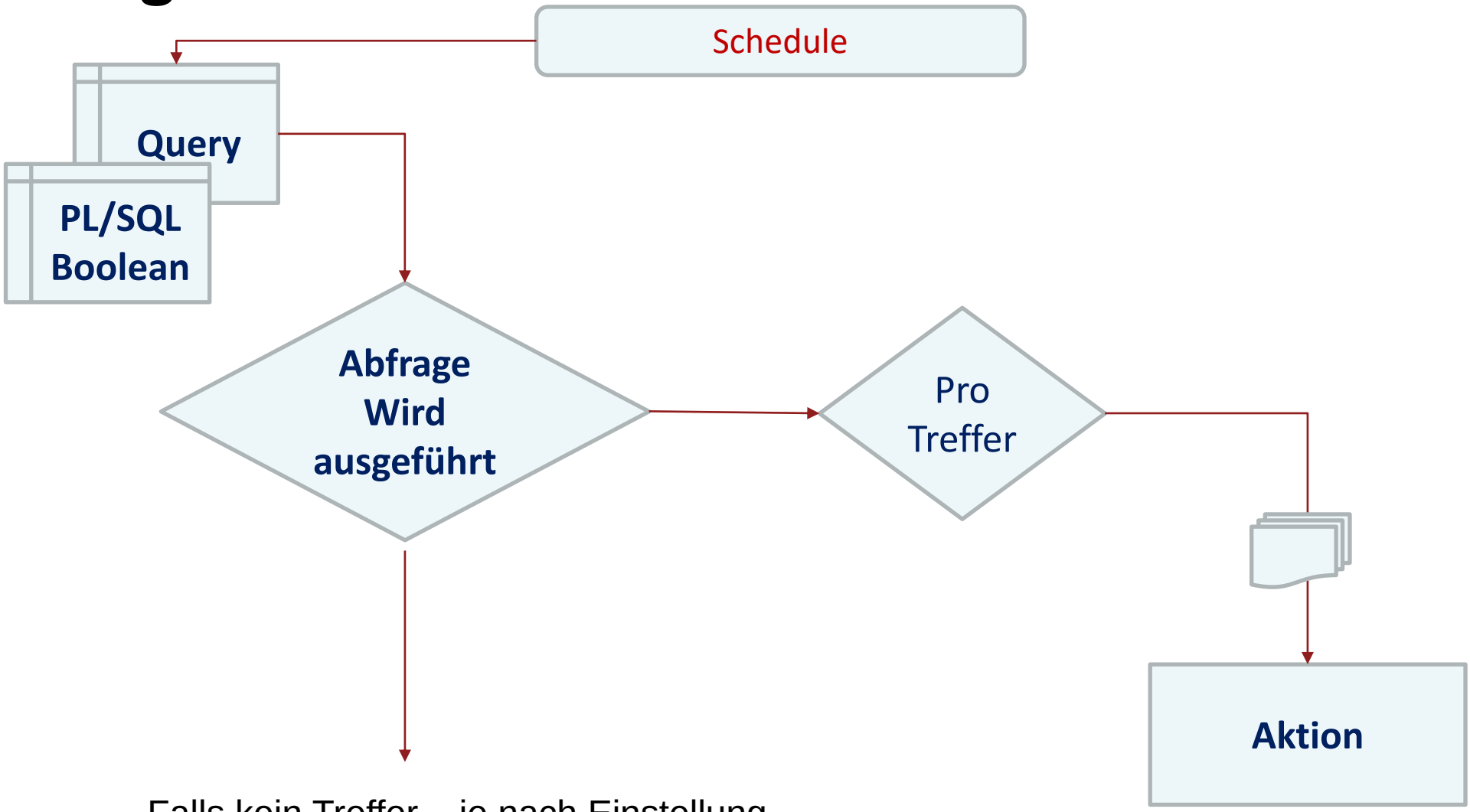
Prüft alle 2 Minuten auf zu startende Jobs

APEXxxxx DB Job
ORACLE_APEX_AUTOMATIONS

Startet über einen „One Time DB Scheduler Job“ im APP Schema
dieser führt die eigentliche Aufgabe im Scope des Schemas durch

DB JOB

Die eigentliche Job Action

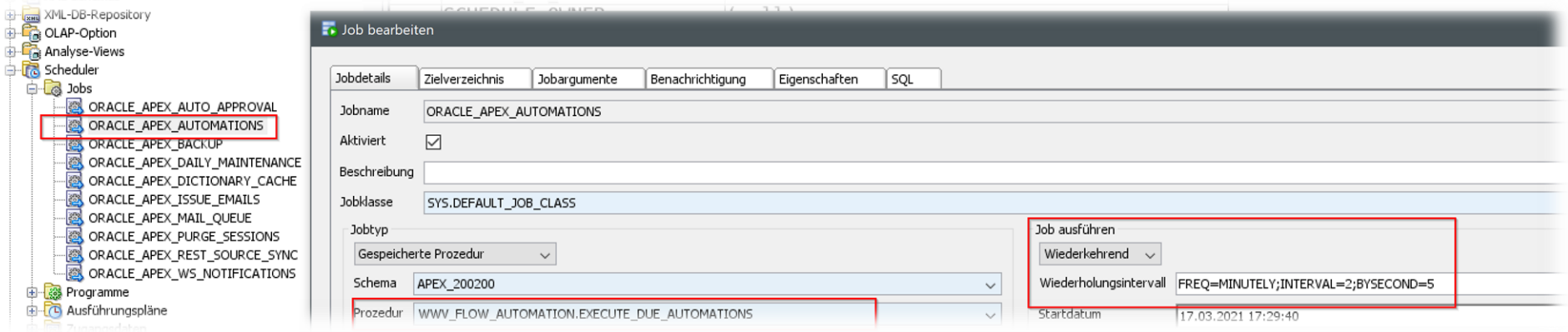


Falls kein Treffer – je nach Einstellung

Was wird erzeugt? – Wer ruft das dann auf? (1)

- Kein DB Job pro Automation!

- Ein zentraler APEX Job, per Default alle 2 Minuten



Was wird erzeugt? – Wer ruft das dann auf? (2)

- Der Job “EXECUTE_DUE_AUTOMATIONS” prüft die “WWV_FLOW_AUTOMATIONS” View

- Falls der eigentliche Task laufen soll, wird ein einmaliger Job über den Scheduler angelegt

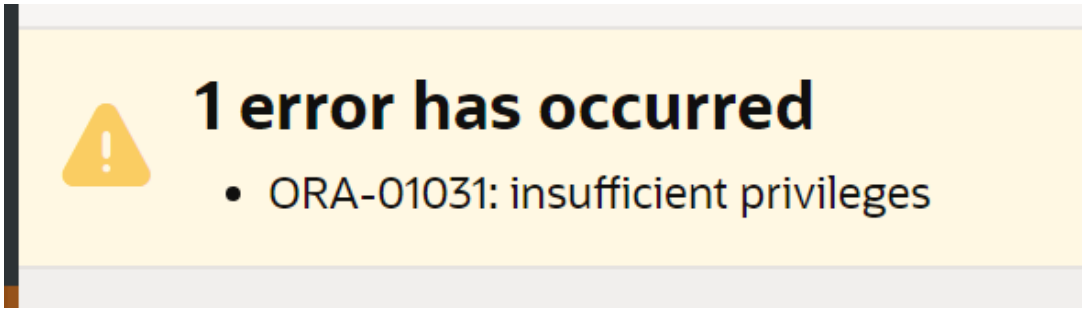
  – Mit der Methode “EXECUTE_AUTOMATION_ACTIONS”

  – Diese startet den eigentliche Task Code im richtigen Scope der jeweiligen APEX Applikation über WWV_FLOW_PLUGIN.EXECUTE_PROCESS

Vorraussetzung – Create Job Recht!

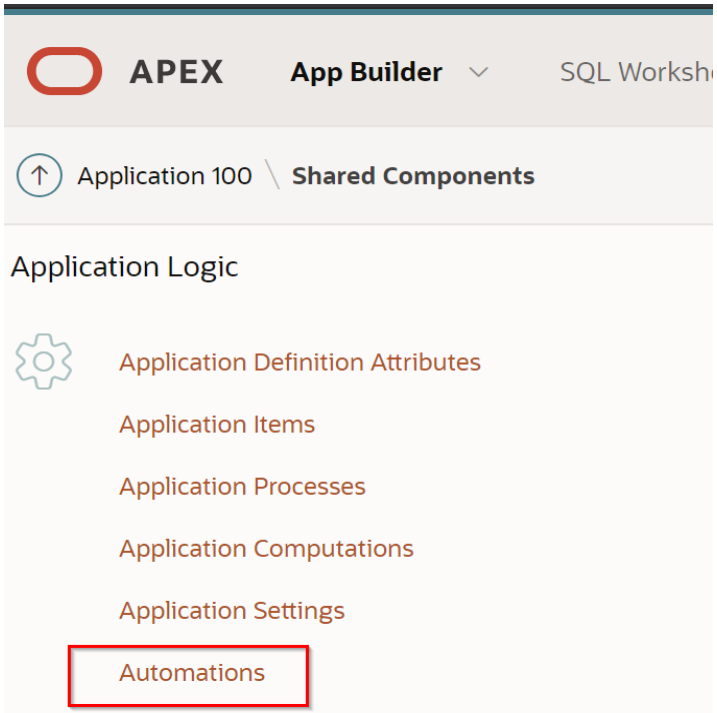
- **Das APP Schema muss das „create job“ Rechte besitzen**

- **Automation lässt sich anlegen, aber nicht aktivieren!**



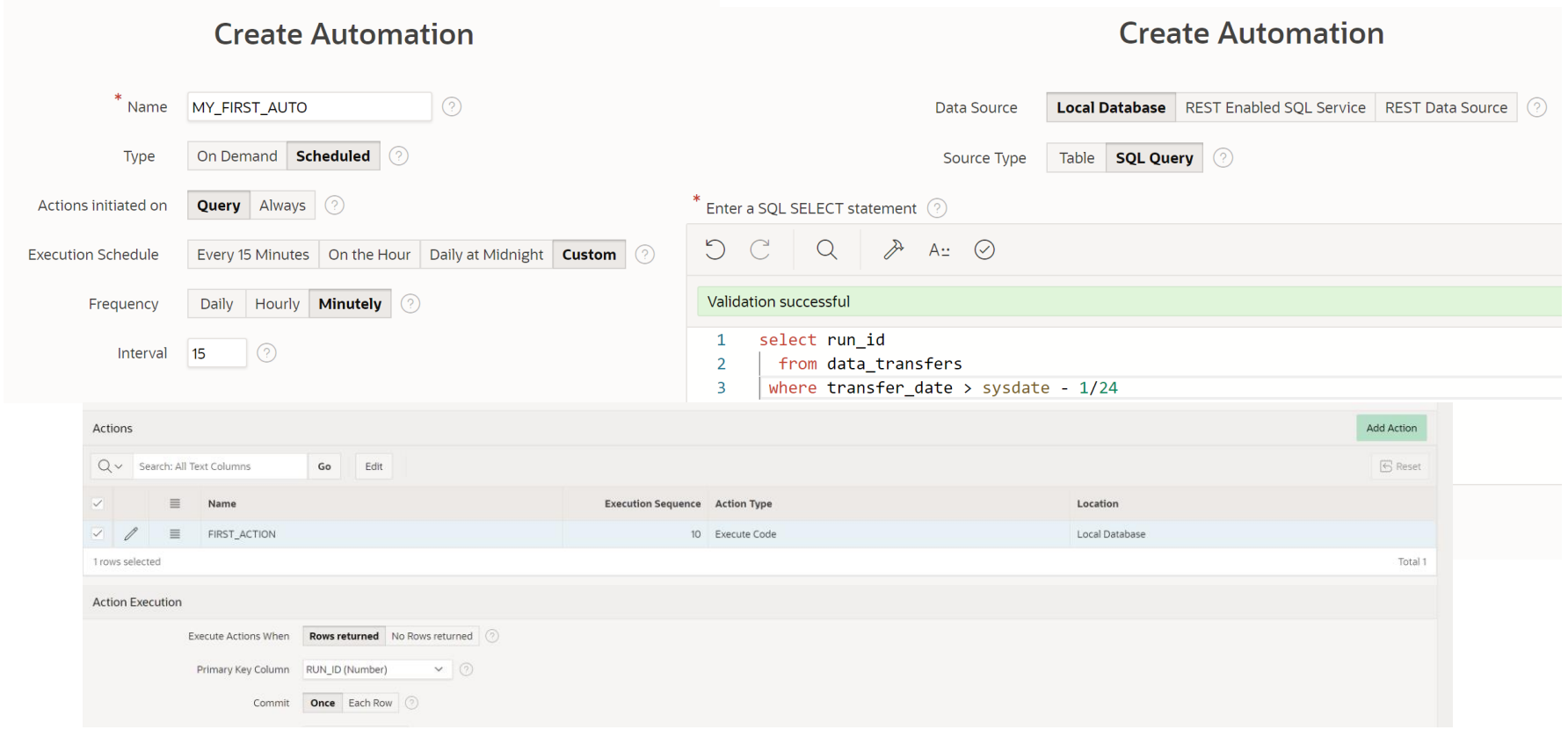
Anlegen (1)

- Unter “**Shared Components**”
  - ” Application Logic - **Automations**“



Anlegen (2)

- Automations anlegen



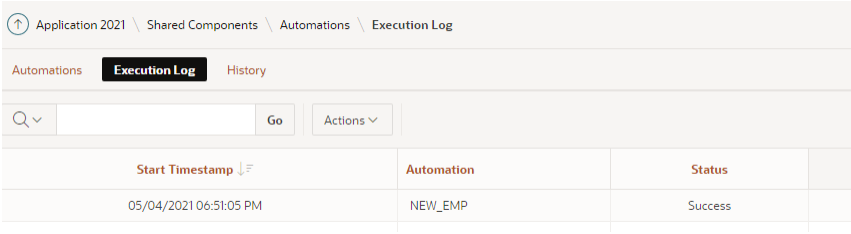
Auf die **STATIC ID** achten, über diese wird der Automation Task später in der API angesprochen!

Warum funktioniert es nicht?

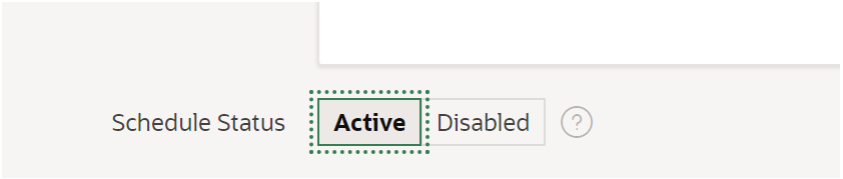
- Bei ersten Fehler wird der Job disabled!

Name	Type	Execute Actions When	Schedule Status	Last Run
NEW_EMP	Scheduled	Rows Returned	Disabled	05/04/2021 03:50:22 PM

- Wo finde ich den Fehler?
  - APEX_AUTOMATION_LOG
  - APEX_AUTOMATION_MSG_LOG



- Nach Behebung des Fehlers wieder aktiveren!



Meta Daten dazu abfragen

- Was wurde angelegt:
  - APEX_APPL_AUTOMATION_ACTIONS
  - APEX_APPL_AUTOMATIONS
- Monitoring
  - APEX_AUTOMATION_LOG
  - APEX_AUTOMATION_MSG_LOG

Vorteil

- In der APEX Welt integriert
- Es wird kein PL/SQL für das Anlegen von Jobs benötigt
- Nach einen Deployment sind die Jobs einfach “da” !
  - Jobs müssen nicht jedes Mal umständlich neu angelegt / angepasst werden