

Betrifft: Oracle-Scheduler, Eine Alternative zu "professionellen" Scheduling?
Autor: Sven Vetter, Trivadis AG, Bern – Schweiz
Quelle: Aus unserer Forschungs- und Beratungstätigkeit

Schlüsselworte:

Scheduler, Ablaufplanung, Jobs, Jobketten, Advanced Queuing, Chain

"Rückblick" auf Oracle10g Release 1

Mit dieser Version wurde ein neuer Scheduler eingeführt. Warum? Mit dem Package DBMS_JOB konnten doch schon seit langer Zeit Jobs gestartet werden!

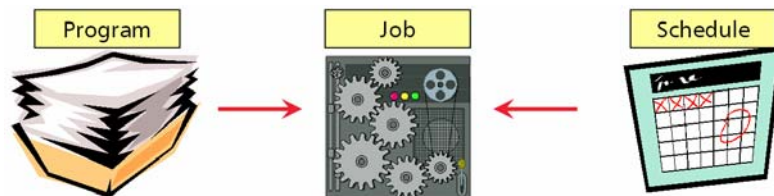
Aber ein paar Sachen fehlten doch, und bevor ich die Funktionalität des neuen Features testete, überlegte ich mir ein paar wichtige Kriterien, welche unbedingt erfüllt sein sollten:

- Grafische Oberfläche
- Jobabhängigkeiten
 - Job B darf nur starten, wenn Job A fertig
 - Job C darf gleichzeitig mit Job B, aber nicht mit Job A laufen
- Zeitfenster (Job darf nur von - bis laufen)
- Ressourcenverwaltung und Prioritätenverwaltung der Jobs
- Benachrichtigung des Administrators
- Einfacher Start von externen Programmen

Vielleicht am Anfang nochmals zur Klarstellung, ich schreibe von dem Scheduler, der neu in die Datenbank eingebaut ist und auf Basis des Packages DBMS_SCHEDULER implementiert ist! Da Oracle schon 5 Scheduler implementiert hat (DBMS_JOB, DBMS_SCHEDULER, Scheduler im Grid- und Database Control, Oracle Workflow), ist es manchmal nicht so einfach, seine Jobs wieder zu finden (vor allem, wenn man sich in einem grafischen Tool etwas zusammengeklickt hat – und nicht genau weiss, welcher Scheduler den Job ausführt...).

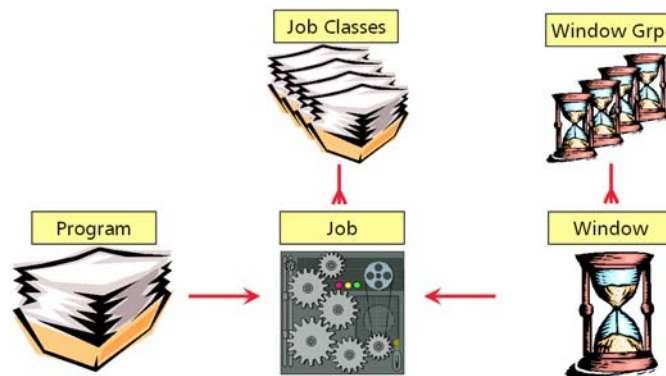
Konzept

Oracle hat den Scheduler sehr modular aufgebaut. Die Grundkomponenten sind:



- Das Program definiert Metadaten:
 - Welches "echte" Programm (PL/SQL Stored Procedure, Anonymous PL/SQL, Shell-Script) wird gestartet
 - Welche Parameter werden übergeben
- Der Schedule legt fest, wann und in welchem Wiederholintervall ein Programm gestartet wird
- Der Job ist die Kombination aus Program und Schedule – die lauffähige Instanz

Neben diesen 3 Grundkomponenten gibt es weitere, optionale Komponenten:



- Das Window erweitert den Schedule, indem eine maximale Laufzeit definiert wird
- Window Groups fassen mehrere Windows zusammen
- Über Job Classes können Ressourcen verwaltet werden

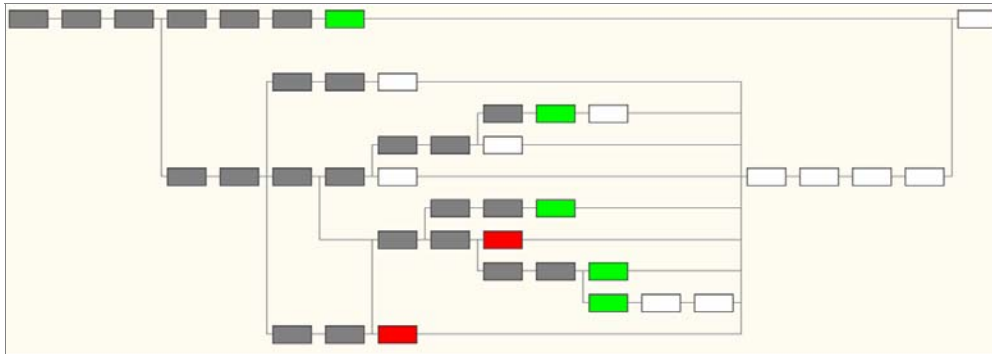
Doch zurück zu meinen Kriterien:

- Grafische Oberfläche: Ja, mit Enterprise Manager
- Jobabhängigkeiten: Fehlt komplett
- Zeitfenster (Job darf nur von - bis laufen)
Ja, über Windows, aber die Programme müssen so geschrieben sein, dass die Abbruchbedingung auch behandelt wird
- Ressourcenverwaltung und Prioritätenverwaltung der Jobs
Ressourcenverwaltung ja, Prioritätenverwaltung auch, die Prioritäten wurden aber nicht beachtet...
- Benachrichtigung des Administrators: Keine Möglichkeit
- Einfacher Start von externen Programmen
Ja, beliebige Shell-Scripts oder externe Programme können aufgerufen werden

Soweit zu 10g Release 1, aber was bringt uns Release 2?

Jobketten

Wieder zurück zu meinen Wünschen – was habe ich erwartet? Etwas in dieser Art:



Also ein grafisches Tool, mit dem ich die Abhängigkeiten zwischen den Jobs definieren kann (wann dürfen Jobs parallel laufen, oder bei welchem Ende-Status darf der nächste Job loslaufen, ...).

Die Realität:

View Chain: SYSTEM.ETL_CHAIN_1

Name ETL_CHAIN_1 Rule Set Name SCHED_RULESET\$1
Schema SYSTEM Rule Set Schema SYSTEM
Enabled TRUE User Rule Set FALSE
Description Test for etl

Steps

Step Name	Type	Object Name	Skip	Pause
STEP1	PROGRAM	SYSTEM.PROGRAM_1	FALSE	FALSE
STEP2	PROGRAM	SYSTEM.PROGRAM_2	FALSE	FALSE
STEP3	PROGRAM	SYSTEM.PROGRAM_3	FALSE	FALSE

Rules

Name	Condition	Action	Description
R1_START	1=1	START "STEP1"	Start Step1
R2_START_STEP2	step1 COMPLETED	START "STEP2"	Start Step2 (after Step1)
R3_START_STEP2	step2 FAILED	START "STEP3"	Start Step3 (on error of step2)
R4_CHAIN_END	step2 SUCCEEDED	END	Chain End
R4_CHAIN_ERROR	step3 COMPLETED	END STEP2.ERROR_CODE	Cleanup (run step3)

Also grafisch schon irgendwie, aber doch auch viel Text...

Und – dieses Fenster aus dem Enterprise Manager stimmt nicht mit dem obigen Ablauf überein, dies sind nur 3 Jobs innerhalb einer Jobkette.

Zurück zum Konzept:

Jobsteps

Eine Jobkette kann aus beliebig vielen Steps bestehen. Steps können vordefinierte Programme, andere Jobketten oder Event Schedules (siehe weiter hinten in diesem Artikel)

sein. Sie werden durch die PL/SQL Procedure DBMS_SCHEDULER oder den Enterprise Manager definiert.

Die Reihenfolge der Steps innerhalb der Job Kette wird nicht durch die Reihenfolge der Definition, sondern durch Regeln definiert.

Jobsteps können weitere Eigenschaften haben:

- PAUSE
Nach der Ausführung des Jobs erhält dieser nicht den Status COMPLETED, d.h., andere Steps, welche darauf abfragen, werden nicht ausgeführt. Das entspricht in anderen Schemata einem Breakpoint, an welchem durch manuellen Eingriff entschieden wird, ob weitergearbeitet werden soll -> siehe Überwachung
- SKIP
Job wird nicht ausgeführt, aber Status wird auf COMPLETED gesetzt
- RESTART_ON_RECOVERY
Wenn durch Instanz-Shutdown der Job gestoppt wurde, wird dieser nach STARTUP neu ausgeführt

Rules

Regeln definieren, wann solch ein Jobstep innerhalb einer Jobkette ausgeführt wird.

Eine Regel besteht aus Bedingung, Aktion, Kommentar (optional) und Regelname (optional, wird durch System vergeben, wenn leer).

Regeln werden erzeugt und bearbeitet durch den Enterprise Manager oder durch die PL/SQL-Prozedur DBMS_SCHEDULER.DEFINE_CHAIN_RULE:

The screenshot shows the 'Create Rule' dialog box. It has a title bar 'Create Rule' and two buttons: 'Cancel' and 'Continue'. The dialog contains the following fields:

- Name:** Two radio buttons are present: 'Use System-Defined Name' (unselected) and 'Specify Name' (selected). A text box next to 'Specify Name' contains the text 'Start_Chain'.
- Description:** A text box containing the text 'Start etl'.
- * Condition:** A text area containing the text 'TRUE'.
- * Action:** A text box containing the text 'START step1'.

```
SYS.dbms_scheduler.define_chain_rule
(chain_name => 'SYSTEM.ETL_CHAIN_1'
, condition => 'step1 COMPLETED'
, rule_name => 'R2_Start_Step2'
, comments => 'Start Step2 (after Step1)'
, action => 'START STEP2');
```

Rules -> Bedingung

Eine Bedingung ist ein logischer Ausdruck (in Scheduler- oder SQL-Syntax). Sobald das Ergebnis wahr ist, wird eine definierte Aktion ausgeführt.

Mindestens eine Regel pro Job Kette muss initial auf TRUE ausgewertet werden, damit die Jobkette gestartet wird. Typischerweise wird das Ergebnis anderer Steps überprüft.

Scheduler-Syntax:

- stepname SUCCEEDED | FAILED | STOPPED
- stepname COMPLETED (= succeeded, failed oder stopped)
- stepname ERROR_CODE <arithmetischer Ausdruck >
z.B. =, !=, <, >, <=, >=, IN, NOT IN

Durch SQL-Syntax kann z.B. getestet werden:

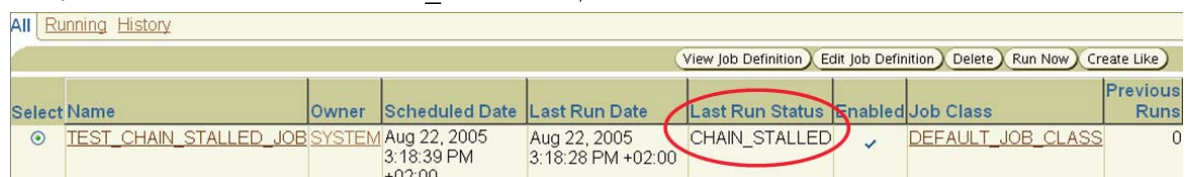
- Wann ein Step gestartet oder beendet wurde
(der Name des Steps wird dabei als Bind-Variable übergeben)
- z.B. in der letzten halben Stunde fertig:
' :step1.END_DATE > SYSDATE-(1/24/2)'
- oder Laufzeit der Steps, z.B. kürzer 30 min.:
' :step1.duration < interval "30" minute'

Rules -> Aktion

3 Arten von Aktionen sind möglich:

- '[AFTER interval] START step1, ...'
Starten von Job Steps (auch verzögert, d.h. nach einem Intervall)
- 'STOP step1, ...'
Beenden von laufenden Job Steps
- 'END [errorcode]'
Beenden der Jobkette

Jede Jobkette muss mit (mindestens) einer Aktion 'END' beendet werden, sondern bleibt die Jobkette im Status 'CHAIN_STALLED', d.h. sie läuft noch!

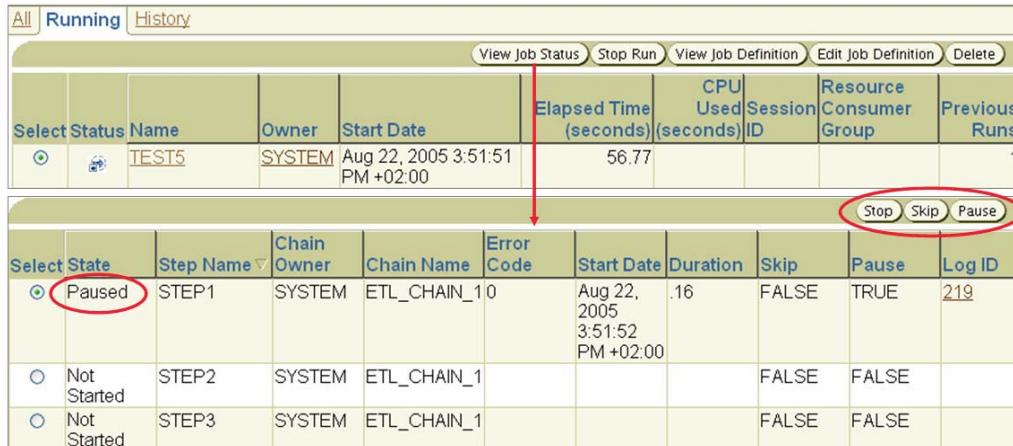


The screenshot shows a table of job definitions. The 'Last Run Status' column for the job 'TEST_CHAIN_STALLED_JOB' is circled in red and contains the value 'CHAIN_STALLED'. The job is owned by 'SYSTEM' and is currently enabled.

Select	Name	Owner	Scheduled Date	Last Run Date	Last Run Status	Enabled	Job Class	Previous Runs
<input checked="" type="radio"/>	TEST_CHAIN_STALLED_JOB	SYSTEM	Aug 22, 2005 3:18:39 PM +02:00	Aug 22, 2005 3:18:28 PM +02:00	CHAIN_STALLED	<input checked="" type="checkbox"/>	DEFAULT_JOB_CLASS	0

Überwachung

Die Überwachung von Jobs kann über die Views DBA | ALL | USER_SCHEDULER_RUNNING_JOBS oder mit dem Enterprise Manager durchgeführt werden:



Select	Status	Name	Owner	Start Date	Elapsed Time (seconds)	CPU Used (seconds)	Session ID	Resource Consumer Group	Previous Runs
<input checked="" type="radio"/>	Running	TEST5	SYSTEM	Aug 22, 2005 3:51:51 PM +02:00	56.77				1

Select	State	Step Name	Chain Owner	Chain Name	Error Code	Start Date	Duration	Skip	Pause	Log ID
<input checked="" type="radio"/>	Paused	STEP1	SYSTEM	ETL_CHAIN_1	0	Aug 22, 2005 3:51:52 PM +02:00	.16	FALSE	TRUE	219
<input type="radio"/>	Not Started	STEP2	SYSTEM	ETL_CHAIN_1				FALSE	FALSE	
<input type="radio"/>	Not Started	STEP3	SYSTEM	ETL_CHAIN_1				FALSE	FALSE	

Beeinflusst (STOP, PAUSE, SKIP, CANCEL) werden die Jobs durch den Enterprise Manager oder durch das Package DBMS_SCHEDULER. Mit folgender Syntax kann z.B. der Job, der im obigen Screenshot auf Pause steht, übersprungen werden:

```
BEGIN
  DBMS_SCHEDULER.ALTER_CHAIN (
    chain_name => 'ETL_CHAIN_1',
    step_name  => 'STEP1',
    attribute  => 'SKIP',
    value      => TRUE);
END;
/
```

Event Scheduling

Jobs können nicht nur zeitbasierend, sondern neu auch durch Ereignisse gestartet werden. Ereignisse können durch Applikationen oder durch Status-Änderungen eines Jobs (started, failed, completed, limit_reached, ...) erzeugt werden.

Anhand dieser Ereignisse können sowohl Jobs als auch Schedules definiert werden.

Diese Funktionalität erlaubt die einfache und flexible Integration des Schedulers in eigene Anwendungen!

Für Applikations Events muss Advanced Queuing benutzt werden. Vorgehen:

- TYPE anlegen, welcher Nutzdaten der Nachricht hält
- Queue-Table und Queue anlegen, Queue starten
- Enqueuer und Dequeuer (Job-Owner) berechtigen

- Job anlegen, optional Bedingung definieren:

General	Schedule	Options
Name JOB_SET_NEW_SAL	Queue Name HR_QUEUE_OWNER.QUEUE_SAL_CHANGES	Priority Medium
Owner SYSTEM	Agent Name	Schedule Limit (minutes)
Enabled TRUE	Condition tab.user_data.new_sal>100	Maximum Runs
Description Set new sal	Available to Start Aug 23, 2005 1:03:53 PM Europe/Berlin	Maximum Failures
Logging Log job runs only Level (RUNS)		Job Weight 1
Job Class DEFAULT_JOB_CLASS		Instance TRUE
Auto Drop FALSE		
Restartable FALSE		

Diese Jobs werden zeitnah gestartet (typischerweise innerhalb 1-2 sek.).

Sollen Daten an den Job übergeben werden, muss eine PL/SQL-Prozedur aufgerufen werden, welche als Inputparameter einen Wert vom Typ des AQ-Types übernimmt.

Eines ist aber dringend zu beachten: Wenn während der Laufzeits eines Jobs neue Events auftreten, werden diese ignoriert! Dadurch können Daten verloren gehen, wenn die Nutzdaten direkt an den Job übergeben werden!

Scheduler-Events werden erst durch eine Jobänderung erzeugt.

Beispiel:

```
BEGIN
  DBMS_SCHEDULER.SET_ATTRIBUTE (
    name      => 'JOB_SET_NEW_SAL',
    attribute => 'raise_events',
    value     => DBMS_SCHEDULER.JOB_FAILED);
END;
/
```

Die Events werden der Queue SYS.SCHEDULER\$_EVENT_QUEUE abgelegt und können von dort abonniert werden

Benutzerdefinierter Kalender

Seit Oracle10g ist die Definition des Wiederhol-Intervalls von Jobs sehr flexibel.

Z.B. ist es einfach zu definieren, dass ein Job am letzten Tag jedes Monats laufen soll:

```
FREQ=MONTHLY; BYMONTHDAY=-1;
```

Oder jeden vierten Montag:

```
FREQ=WEEKLY; INTERVAL=4; BYDAY=MON;
```

Durch die Schachtelung mehrerer Schedules kann dies seit Oracle10g Release 2 noch erweitert werden.

Beispiel: Ich habe einen Kalender, welcher meine Firmenfeiertage definiert:

```
BEGIN
sys.dbms_scheduler.create_schedule(
  repeat_interval => 'FREQ=WEEKLY;BYDAY=MON,TUE,WED,THU,FRI',
  start_date =>
    to_timestamp_tz('2005-08-08 Europe/Paris','YYYY-MM-DD TZR'),
  end_date =>
    to_timestamp_tz('2005-09-02 Europe/Paris','YYYY-MM-DD TZR'),
  schedule_name => 'COMPANY_HOLIDAYS');
END;
```

Dieser Kalender kann in einem Schedule benutzt werden. Im Beispiel wird ein Schedule definiert, der jeden Freitag 12:00 Uhr, aber nicht während der Betriebsferien, laufen soll:

```
BEGIN
sys.dbms_scheduler.create_schedule(
  repeat_interval =>
    'FREQ=YEARLY;BYDAY=FRI;BYHOUR=12;EXCLUDE=COMPANY_HOLIDAYS',
  start_date =>
    to_timestamp_tz('2005-08-23 +2:00', 'YYYY-MM-DD TZH:TZM'),
  schedule_name => 'WEEKLY_JOB');
END;
```

Die neue Syntax (INCLUDE/EXCLUDE) kann nur mittels PL/SQL erzeugt werden. Im Enterprise Manager kann aber die Definition kontrolliert werden, z.B. werden die nächsten fünf geplanten Durchführungen angezeigt (Aufgerufen im August 2005):

Next 5 Scheduled Dates	
	Refresh
Date	
Sep 9, 2005 12:20:00 PM	
Sep 16, 2005 12:20:00 PM	
Sep 23, 2005 12:20:00 PM	
Sep 30, 2005 12:20:00 PM	
Oct 7, 2005 12:20:00 PM	

Realität mit Oracle10g Release2

Kommen wir zurück zu meinen Wünschen.

Was ist besser geworden:

- Jobabhängigkeiten: Funktioniert jetzt gut
- Benachrichtigung des Administrators: Funktioniert jetzt besser. Es gibt noch keine automatische Benachrichtigungen, aber ich kann ein PL/SQL-Programm oder ein Shell-Script, welches die Benachrichtigung übernimmt, in eine Jobkette einbauen (und über Regeln definieren, wann eine Mail verschickt werden soll)

Fazit

Der für mich wichtigste Punkt funktioniert jetzt gut – die Jobabhängigkeiten. Aus meiner Sicht ist der Scheduler im Moment nur für kleinere bis mittlere Abläufe geeignet, da das Pflegen der Regeln sehr schnell unübersichtlich wird.

Oracle hat aber alle Funktionalität eingebaut, die eine richtige Jobsteuerung braucht (und das zum Teil flexibler als andere Anbieter, siehe z.B. SQL-Syntax in Regeln, Kombination mit Advanced Queuing, Benutzerdefinierte Kalender).

Deswegen hoffe ich, dass es irgendwann ein Tool geben wird, mit dem die Abhängigkeiten grafisch erstellt werden können – spätestens dann werde ich grosse Jobpläne wieder testen... - und wahrscheinlich wieder darüber schreiben ☺

Ein Tipp zum Schluss: Besuchen Sie die Schulungen der Firma Trivadis, dort lernen Sie mehr über dieses und andere Features – und werden auf dem Laufenden gehalten, wann der Einsatz einer bestimmten Funktionalität sinnvoll ist.

Kontaktadresse:

Sven Vetter

Trivadis AG

Papiermühlestrasse 73

CH-3014 Bern

Tel.: +41-31-928 09 60

Fax: +41-31-928 09 64

E-Mail Sven.Vetter@trivadis.com

Internet: <http://www.trivadis.com>