

# **Oracle 11g DBA Fundamentals Overview**

Lesson 12: Automating Tasks  
with the Scheduler

## Objectives

- After completing this lesson, you should be able to:
  - Simplify management tasks by using the Scheduler
  - Create a job, program, and schedule
  - Monitor job execution
  - Use a time-based or event-based schedule for executing Scheduler jobs

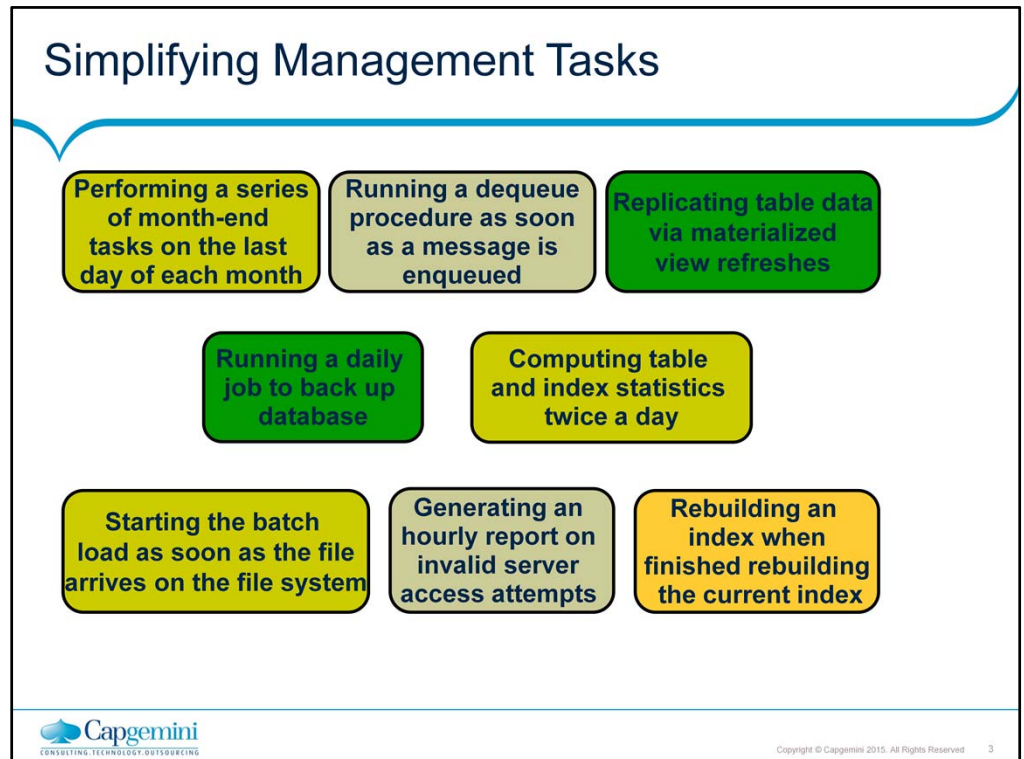


Copyright © Capgemini 2015. All Rights Reserved. 2

### Objectives

For information about the various Scheduler components and their interaction, see the Oracle Database Administrator's Guide.

For detailed information about the DBMS\_SCHEDULER package, see the Oracle Database PL/SQL Packages and Types Reference.



### Simplifying Management Tasks

Many tasks in the Oracle environment need job-scheduling capabilities. Routine database maintenance and application logic require jobs to be scheduled and run periodically. Business-to-business (B2B) applications require scheduling for their business events. DBAs need to schedule regular maintenance jobs in specified time windows.

Oracle Database 11g provides advanced scheduling capabilities through the database Scheduler, which is a collection of functions and procedures in the DBMS\_SCHEDULER package. The Scheduler can be invoked in any SQL environment, or through Enterprise Manager.

The Scheduler enables database administrators and application developers to control when and where various tasks take place in the database environment. These tasks can be time consuming and complicated; using the Scheduler, you can manage and plan these tasks.

Scheduler jobs can be started based on time or when a specified event occurs, and the Scheduler can raise events when a job's state changes (for example, from RUNNING to COMPLETE). You can also use a named series of programs that are linked together for a combined objective.

## A Simple Job

Database: test@11g-oracle.com > Scheduler Jobs > Create Job Logged in As HR

**WHEN**

Create Job Show SQL Cancel OK

**General** Schedule Options

Name: CREATE\_LOG\_TABLE\_JOB

Owner: HR

Enabled: ☒ Yes ☐ No

Description: Create the SESSION\_HISTORY table

Logging Level: Log job runs only (RUNS)   
 Specify logging requirements for the job

Job Class: DEFAULT\_JOB\_CLASS Create Job Class

Auto Drop: FALSE   
 Specify whether the job should be dropped after completion

Restartable: FALSE   
 Specify whether the job can be restarted manually or in the event of failure

**WHAT**

Command

Select the command type for the job, then enter the command requirements.

Command Type: PL/SQL Block Change Command Type

```
BEGIN
  execute immediate
    ('create table session_history(
      snap_time TIMESTAMP WITH LOCAL TIME ZONE,
      num_sessions NUMBER);'); end;';
start_date => systimestamp at time zone 'America/New_York',
job_class => 'DEFAULT_JOB_CLASS',
comments => 'Create the SESSION_HISTORY table',
auto_drop => FALSE, enabled => TRUE);
END;
```

Capgemini  
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved. 4

### A Simple Job

A job has two key components: action, “what” needs to be done and schedule, “when” action occurs. The “what” is expressed in the Command region of the screenshot shown in the slide and the job\_type and job\_action parameters.

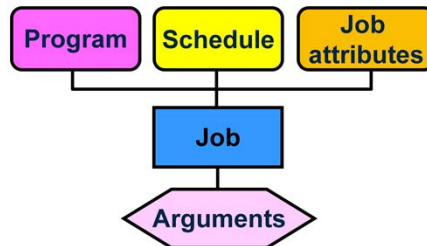
```
BEGIN
  sys.dbms_scheduler.create_job(
    job_name => 'HR'. 'CREATE_LOG_TABLE_JOB',
    job_type => 'PLSQL_BLOCK', job_action => 'begin
      execute immediate ('create table session_history(
        snap_time TIMESTAMP WITH LOCAL TIME ZONE,
        num_sessions NUMBER)); end;';
    start_date => systimestamp at time zone 'America/New_York',
    job_class => 'DEFAULT_JOB_CLASS',
    comments => 'Create the SESSION_HISTORY table',
    auto_drop => FALSE, enabled => TRUE);
END;
```

A job is also defined by “when” the desired action needs to take place. The “when” is expressed in a “schedule,” which can be based on time (see the start\_date parameter) or events, or be dependent on the outcome of other jobs. These options are discussed in this lesson.

## Key Components and Steps

■ To simplify management tasks with the Scheduler, perform the following steps:

1. Create a program.
2. Create and use a schedule.
3. Create and submit a job.
4. Monitor a job.



### Key Components and Key Steps

The Scheduler offers a modularized approach for managing tasks within the Oracle database. By breaking down a task into its components, such as time, location, and database object, the Scheduler offers you an easier way to manage your database environment. The Scheduler uses three basic components:

A job specifies what needs to be executed and when. For example, the “what” could be a PL/SQL procedure, a native binary executable, a Java application, or a shell script. You can specify the program (what) and schedule (when) as part of the job definition, or you can use an existing program or schedule instead. You can use arguments for a job to customize its run-time behavior.

A schedule specifies when and how many times a job is executed.

A schedule can be based on time or an event. You can define a schedule for a job by using a series of dates, an event, or a combination of the two, along with additional specifications to denote repeating intervals. You can store the schedule for a job separately and then use the same schedule for multiple jobs.

A program is a collection of metadata about a particular executable, script, or procedure. An automated job executes some task. Using a program enables you to modify the job task, or the “what,” without modifying the job itself. You can define arguments for a program, enabling users to modify the run-time behavior of the task.

## 1. Creating a Program

The screenshot shows the Oracle Enterprise Manager 10g Database Scheduler interface. On the left, a navigation tree includes 'Database Scheduler', 'Jobs', 'Chains', 'Schedulers', 'Programs', 'Job Classes', 'Window Groups', and 'Global Attributes'. The 'Programs' link is highlighted. The main pane shows the 'Scheduler Programs' page with a table of existing programs and a 'Create' button. Below this, the 'Create Program' form is displayed with fields for Name, Schema (set to 'HR'), Enabled (radio buttons for Yes/No), Description, Type (set to 'PLSQL\_BLOCK'), and Source. A text box on the right contains the SQL code for creating the program.

```

BEGIN
DBMS_SCHEDULER.CREATE_PROGRAM(
  program_name => 'CALC_STATS2',
  program_action =>
'HR.UPDATE_HR_SCHEMA_STATS',
  program_type => 'STORED_PROCEDURE',
  enabled => TRUE);
END;
/

```

### 1. Creating a Program

Use the `CREATE_PROGRAM` procedure to create a program. Creating a program is an optional part of using the Scheduler. You can also encode the action to be performed within an anonymous PL/SQL block in the `CREATE_JOB` procedure. By creating the program separately, you can define the action once, and then reuse this action within multiple jobs. This enables you to change the schedule for a job without having to re-create the PL/SQL block. You can also customize the job by specifying argument values.

To create a program in your own schema, you need the `CREATE JOB` privilege. A user with the `CREATE ANY JOB` privilege can create a program in any schema.

A program is created in a disabled state by default (unless the `enabled` parameter is set to `TRUE`). A disabled program cannot be executed by a job until it is enabled. You can specify that a program should be created in the enabled state by specifying a value of `TRUE` for `enabled`.

The program action is a string specifying a procedure, executable name, or a PL/SQL anonymous block, depending on what `program_type` is set to.

If you have a procedure called `UPDATE_HR_SCHEMA_STATS` that collects the statistics for the `hr` schema, then you can create a program to call this procedure.

In Enterprise Manager, select **Administration > Programs** and click the **Create** button.

## 2. Creating and Using Schedules

Database Instance: orcl > Scheduler Schedules > Create Schedule  
 Logged in As SYS

Create Schedule

Name:   
 Owner: HR  
 Description:   
 Time Zone: America/Los\_Angeles  
 Schedule Type: Standard  
 Repeating: By Hours, Interval (Hours): 1  
 Available to Start: ☒ Immediately  
 Not Available After: ☒ No End Date

```

BEGIN
  DBMS_SCHEDULER.CREATE_SCHEDULE(
    schedule_name => 'stats_schedule',
    start_date => SYSTIMESTAMP,
    end_date => SYSTIMESTAMP + 30,
    repeat_interval =>
      'FREQ=HOURLY;INTERVAL=1',
    comments => 'Every hour');
END;
  
```

Capgemini  
 CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved. 7

### 2. Creating and Using Schedules

By using a schedule (instead of specifying the execution times for a job within the job definition), you can manage the scheduled execution of multiple jobs without having to update multiple job definitions. If a schedule is modified, then each job that uses that schedule automatically uses the new schedule.

Use the `CREATE_SCHEDULE` procedure in the `DBMS_SCHEDULER` PL/SQL package to create a schedule.

The `start_date` represents the date on which the schedule becomes active. The schedule cannot refer to any dates before this date. The schedule is not valid after the `end_date`.

You can schedule repeated executions by supplying a calendaring expression for `repeat_interval`. This calendaring expression is used to generate the next date of the schedule. Dates falling after the `end_date` time are not included in the schedule.

In the example shown in the slide, a schedule named `STATS_SCHEDULE` is created, specifying a repeat interval of every four hours, starting now, and continuing for 30 days.

You can use Enterprise Manager to create schedules as shown in the slide.

### 3. Creating and Running a Job

**Create Job**

General Schedule Options

Name: LOG\_SESSIONS\_JOB

Owner: HR

Enabled: ☒ Yes ☐ No

Description: Count sessions with HR.LOG\_SESS\_COUNT\_PRGM

Logging Level: Log everything (FULL)

Job Class: DEFAULT\_JOB\_CLASS

Auto Drop: FALSE

Restartable: FALSE

Command

Command Type: Program

Program Name: HR.LOG\_SESS\_COUNT\_PRGM

#### 3. Creating and Running a Job

A job is a combination of a schedule and a description of what to do, along with any additional arguments that are required by the job.

The program or "Command" can be a precreated PL/SQL or Java program, an anonymous PL/SQL block, or an executable that is run from the operating system's command line.

The schedule for a job can be a predefined schedule (created with the DBMS\_SCHEDULER.CREATE\_SCHEDULE procedure) or defined as part of the job creation. The schedule specifies attributes about when the job is run, such as:

- A start time, which defines when the job is picked for execution and an end time, which specifies the time after which the job is no longer valid and is not scheduled any more

- An expression specifying a repeating interval for the job

- A complex schedule created by combining existing schedules

- A condition or change in state, called an event, that must be met before the job is started

There are many attributes that you can set for a job. Attributes control how the job executes.

To run a job in Enterprise Manager, select Administration > Jobs.



## 4. Monitoring a Job

```
SELECT job_name, status, error#, run_duration
FROM USER_SCHEDULER_JOB_RUN_DETAILS;

JOB_NAME      STATUS ERROR# RUN_DURATION
-----
GATHER_STATS_JOB SUCCESS    0 +000 00:08:20
PART_EXCHANGE_JOB FAILURE 6576 +000 00:00:00
```

**Scheduler Jobs**

Page Refreshed Sep 20, 2005 9:43:59 AM [Refresh](#) [Create](#)


[All](#) [Running](#) [History](#)

[Purge All Logs](#)

[View Job Status](#) [Purge Log](#) [View Job Definition](#)

Previous 25 51-75 of 3647 Next 25

Select	Status	Name	Owner	Completion Date	Run Duration (minutes)
<input checked="" type="radio"/>	✓	LOG_SESSIONS_JOB	HR	Sep 19, 2005 11:22:00 AM -07:00	0.0
<input type="radio"/>	✓	RLMSCHDNAGACTION	EXFSYS	Sep 19, 2005 11:21:05 AM -07:00	0.0
<input type="radio"/>	✓	LOG_SESSIONS_JOB	HR	Sep 19, 2005 11:19:00 AM -07:00	0.0
<input type="radio"/>	✓	LOG_SESSIONS_JOB	HR	Sep 19, 2005 11:16:00 AM -07:00	0.0
<input type="radio"/>	✓	LOG_SESSIONS_JOB	HR	Sep 19, 2005 11:13:00 AM -07:00	0.0


Copyright © Capgemini 2015. All Rights Reserved. 9

### 4. Monitoring a Job

The DBA\_SCHEDULER\_JOB\_RUN\_DETAILS view has a row for each job instance. Each row contains information about the job execution for that instance.

[DBA|ALL]\_SCHEDULER\_JOB\_RUN\_DETAILS views have the following columns:

- LOG\_ID: The unique identifier of the log entry
- LOG\_DATE: The time stamp of the log entry
- OWNER: The job owner
- JOB\_NAME: The name of the job
- STATUS: The status of the job execution
- ERROR#: The number of the first error encountered
- REQ\_START\_DATE: The time at which the job was scheduled to start
- ACTUAL\_START\_DATE: The time at which the job was actually started
- RUN\_DURATION: The duration of execution of the job
- INSTANCE\_ID: The instance upon which the job ran
- SESSION\_ID: The session the job ran within
- SLAVE\_PID: The process ID of the slave process used to perform the job execution
- CPU\_USED: The amount of CPU used for the job run
- ADDITIONAL\_INFO: Additional information about the job run

## Using a Time-Based or Event-Based Schedule

**Key Comp. & Steps**  
 > Schedules  
 Job Chains  
 Adv.Concepts

```

graph TD
    Schedule[Schedule] -.- Time[Time]
    Schedule -.- Event[Event]
      
```

Copyright © Capgemini 2015. All Rights Reserved. 10

### Using a Time-Based or Event-Based Schedule

To specify a time-based schedule for a job, you can specify either a calendaring expression or a datetime expression

When using a calendaring expression, the next start time for a job is calculated using the repeat interval and the start date of the job. When using datetime expressions, the specified expression determines the next time that the job should run. If no repeat interval is specified, the job runs only once on the specified start date.

If a job uses an event-based schedule, the job runs when the event is raised. At a high level, an event can be viewed as a change in state. An event occurs when a Boolean condition changes its state from FALSE to TRUE, or TRUE to FALSE.

The Scheduler uses Oracle Streams Advanced Queuing (AQ) to raise and consume events.

Note: The Scheduler does not guarantee that a job executes on the exact time because the system may be overloaded and thus resources may be unavailable.

## Creating a Time-Based Job

### ■ Example:

- Create a job that calls a backup script every night at 11:00, starting tonight.

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB(
  job_name=>'HR.DO_BACKUP',
  job_type => 'EXECUTABLE',
  job_action => '/home/usr/dba/rman/nightly_incr.sh',
  start_date=> SYSDATE,
  repeat_interval=>'FREQ=DAILY;BYHOUR=23',
                  /* next night at 11:00 PM */
  comments => 'Nightly incremental backups');
END;
/
```



Copyright © Capgemini 2015. All Rights Reserved 11

### Creating a Time-Based Job

Use the `CREATE_JOB` procedure of the `DBMS_SCHEDULER` package to create a job. Jobs are created disabled by default and they become active and scheduled only when they are explicitly enabled. All job names are of the form: `[schema.]name`.

You should use `SYSTIMESTAMP` and specify a time zone so that when the time changes because of daylight saving time, your job adjusts its execution time automatically.

By default, a job is created in the current schema. You can create a job in another schema

by specifying the name of the schema, as shown in the example in the slide. The job owner is the user in whose schema the job is created, whereas the job creator is the user who created the job. Jobs are executed with the privileges of the job owner. The national language support (NLS) environment of the job when it runs is the same as that which was present at the time the job was created.

The `job_type` parameter indicates the type of task to be performed by the job. The possible values are:

`PLSQL_BLOCK`: An anonymous PL/SQL block

`STORED_PROCEDURE`: A named PL/SQL, Java, or external procedure

`EXECUTABLE`: A command that can be executed from the operating system command line

## Creating a Time-Based Job Full Notes Page



Copyright © Capgemini 2015. All Rights Reserved 12

### Creating a Time-Based Job (continued)

The `job_action` parameter can be the name of the procedure to run, the name of a script or operating system command, or an anonymous PL/SQL code block, depending on the value of the `job_type` parameter.

In the example in the slide, `job_type` is specified as `EXECUTABLE` and `job_action` is the full OS-dependent path of the desired external executable plus optionally any command-line arguments.

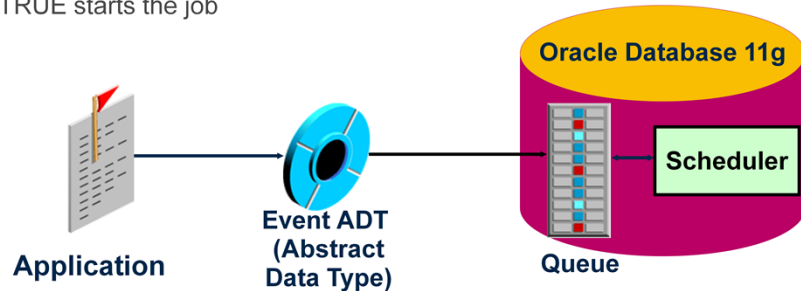
An external job is a job that runs outside the database. All external jobs run as a low-privileged guest user, as has been determined by the database administrator while configuring external job support. Because the executable is run as a low-privileged guest account, you should verify that it has access to necessary files and resources. Most, but not all, platforms support external jobs. For platforms that do not support external jobs, creating or setting the attribute of a job or a program to type `EXECUTABLE` returns an error.

Refer to your Oracle database platform-specific documentation for more information about configuring the environment to run external programs with the Scheduler. For example, you may need to reference one or more of the following books:

- Oracle Database Platform Guide 11g for Windows
- Oracle Database Installation Guide for UNIX Systems
- Oracle Database Release Notes 11g for AIX-Based Systems
- Oracle Database Release Notes 11g for hp HP-UX PA-RISC (64-bit)

## Creating an Event-Based Schedule

- To create an event-based job, you must set:
  - A queue specification (where your application enqueues messages to start a job)
  - An event condition (same syntax as an Oracle Streams AQ rule condition) that if TRUE starts the job



### Creating an Event-Based Schedule

Jobs can be triggered based on events. An application can notify the Scheduler to start a job by enqueueing a message onto an Oracle Streams AQ queue. A job started in this way is referred to as an event-based job. To create an event-based job, you must set the following two additional attributes with the `CREATE_JOB` procedure:

**queue\_spec:** A queue specification that includes the name of the queue where your application enqueues messages to raise job start events, or in the case of a secure queue, the `<queue_name>`, `<agent_name>` pair

**event\_condition:** A conditional expression based on message properties that must evaluate to TRUE for the message to start the job. The expression must use the same syntax as an Oracle Streams AQ rule condition. You can include user data properties in the expression, provided that the message payload is a user-defined object type, and that you prefix object attributes in the expression with `tab.user_data`.

You can either specify `queue_spec` and `event_condition` as in-line job attributes, or create an event-based schedule with these attributes and then create a job that references this schedule.

## Creating Event-Based Schedules with Enterprise Manager

The screenshot shows the 'Schedule' configuration page in Oracle Enterprise Manager. It includes the following fields and options:

- Schedule**
  - Time Zone:  (with a calendar icon)
  - Schedule Type:  (dropdown menu)
- Event Parameters**
  - Queue Name:  (with a 'Change Queue' button)
  - Agent Name:  (with a calendar icon)
  - Condition:



Copyright © Capgemini 2015. All Rights Reserved 14

### Creating Event-Based Schedules with Enterprise Manager

The Create Schedule page enables you to choose between a standard, time-based schedule and an event-based schedule. If you choose an event-based schedule, then the interface changes and you can specify the queue name, agent name, and event condition, in addition to the other schedule attributes.

#### Note

The Scheduler runs the event-based job for each occurrence of an event that matches event\_condition. However, events that occur while the job is already running are ignored; the event gets consumed, but does not trigger another run of the job.

#### References:

See the Oracle Streams Advanced Queuing User's Guide and Reference for information about how to create queues and enqueue messages.

For more information about Oracle Streams AQ rules and event conditions, see the DBMS\_AQADM.ADD\_SUBSCRIBER procedure in the Oracle Database PL/SQL Packages and Types Reference 11g Release 2 manual.

## Creating an Event-Based Job

- Example: Create a job that runs if a batch load data file arrives on the file system before 9:00 a.m.

```
BEGIN
DBMS_SCHEDULER.CREATE_JOB(
  job_name=>'ADMIN.PERFORM_DATA_LOAD',
  job_type => 'EXECUTABLE',
  job_action => '/home/usr/dba/rman/report_failure.sh',
  start_date => SYSTIMESTAMP,
  event_condition => 'tab.user_data.object_owner =
"HR" and tab.user_data.object_name = "DATA.TXT"
and tab.user_data.event_type = "FILE_ARRIVAL"
and tab.user_data.event_timestamp < 9 ',
  queue_spec => 'HR.LOAD_JOB_EVENT_Q');
END;
```



Copyright © Capgemini 2015. All Rights Reserved 15

### Creating an Event-Based Job

To specify event information as job attributes, you use an alternate syntax of `CREATE_JOB` that includes the `queue_spec` and `event_condition` attributes. The job can include event information in-line as job attributes or can specify event information by pointing to an event schedule. The example shown in the slide uses an in-line event-based schedule.

The example in the slide shows a job that is started when a file arrives on the operating system, as long as the file arrives before 9:00 a.m. Assume that the message payload is an object with four attributes named `object_owner`, `object_name`, `event_type`, and `event_timestamp`.

The example uses a user-defined event. Therefore, before this job can be started, when the file arrives on the file system, a program or procedure must enqueue the event object type with the proper information into the specified event queue. The `HR.LOAD_JOB_EVENT_Q` queue must be of the same type as the event object type used for notifying the Scheduler of an event occurrence. That is, the `HR.LOAD_JOB_EVENT_Q` queue must be a typed queue where the type has four attributes named `object_owner`, `object_name`, `event_type`, and `event_timestamp`.

For more information about how to create queues and enqueue messages, refer to the Oracle Streams Advanced Queuing User's Guide and Reference documentation.



## Event-Based Scheduling

- Event types:

- User- or application-generated events
- Scheduler-generated events

- Events raised by Scheduler jobs:

- JOB\_START
  - JOB\_SCH\_LIM\_REACHED
- JOB\_SUCCEEDED
- JOB\_DISABLED
- JOB\_FAILED
  - JOB\_CHAIN\_STALLED
- JOB\_BROKEN
  - JOB\_ALL\_EVENTS
- JOB\_COMPLETED  
JOB\_RUN\_COMPLETED
- JOB\_STOPPED

```
DBMS_SCHEDULER.SET_ATTRIBUTE('hr.do_backup',
'raise_events', DBMS_SCHEDULER.JOB_FAILED);
```

- Example of raising an event:



Copyright © Capgemini 2015. All Rights Reserved 16

### Event-Based Scheduling

You can create a job that directly references an event as the means to start the job, instead of assigning a schedule to the job. There are two types of events:

**User- or application-generated events:** An application can raise an event to be consumed by the Scheduler. The Scheduler reacts to the event by starting a job. An example of such events: a running job completes; a file arrives on the file system; an account within the database is locked; and the inventory reaches a low threshold.

**Scheduler-generated events:** The Scheduler can raise an event to indicate state changes that occur within the Scheduler itself. For example, the Scheduler can raise an event when a job starts, when a job completes, when a job exceeds its allotted run time, and so on. The consumer of the event is an application that performs some action in response to the event.

You can configure a job so that the Scheduler raises an event when the job's state changes. You do this by setting the `raise_events` job attribute. By default, a job does not raise any state change events until you alter the `raise_events` attribute for a job. To alter this attribute, you must first create the job by using the `CREATE_JOB` procedure and then use the `SET_ATTRIBUTE` procedure to modify the attribute's default value. The example shows that the `hr.do_backup` job is altered, so that it raises an event if the job fails.



## Event-Based Scheduling (continued)

After you enable job state change events for a job, the Scheduler raises these events by enqueueing messages onto the default event queue SYS.SCHEDULER\$\_EVENT\_QUEUE.

The default Scheduler event queue is a secure queue. Depending on your application, you may have to configure the queue to enable certain users to perform operations on it. See the *Oracle Streams Concepts and Administration* documentation for information about secure queues.

The default Scheduler event queue is intended primarily for Scheduler-generated events. Oracle does not recommend the use of this queue for user applications, or user-defined events.

Event Type	Description
JOB_START	The job is started.
JOB_SUCCEEDED	The job is successfully completed.
JOB_FAILED	The job failed, either by raising an error or by abnormally terminating.
JOB_BROKEN	The job is disabled and changed to the BROKEN state, because it exceeded the number of failures defined by the MAX_FAILURES job attribute.
JOB_COMPLETED	The job is completed, because it reached the values set by the MAX_RUNS or END_DATE job attributes.
JOB_STOPPED	The job is stopped by a call to the STOP_JOB procedure.
JOB_SCH_LIM_REACHED	The job's schedule limit is reached. The job is not started, because the delay in starting the job exceeded the value of the SCHEDULE_LIMIT job attribute.
JOB_DISABLED	The job is disabled by the scheduler or by a call to the SET_ATTRIBUTE procedure.
JOB_CHAIN_STALLED	A job running a chain is put into the CHAIN_STALLED state. A running chain becomes stalled if there are no steps running or scheduled to run and the chain EVALUATION_INTERVAL is set to NULL. The chain waits for manual intervention.
JOB_ALL_EVENTS	JOB_ALL_EVENTS is not an event, but a constant, that provides an easy way for you to enable all events.
JOB_RUN_COMPLETED	A job run is completed. It either failed, succeeded, or is stopped.

## Summary

- In this lesson, you should have learned how to:
  - Simplify management tasks by using the Scheduler
  - Create a job, program, and schedule
  - Monitor job execution
  - Use a time-based or event-based schedule for executing Scheduler jobs

