

UNIT 8

Automate Tasks with the Scheduler

(4 Hours) — Database Administration (CACS405)

*Note: These notes are prepared for study purposes and **may contain errors or omissions**. Syntax, file paths and screen names differ between Oracle versions and operating systems (examples here follow Oracle 19c on Linux). Please verify important points with your textbook, class notes and the official Oracle documentation, and test commands on a practice database, never on a production system.*

8.1 Introduction to the Scheduler

The **Oracle Scheduler** is a built-in job scheduling service, accessed through the `DBMS_SCHEDULER` package. It lets a DBA run tasks automatically at a chosen time, repeatedly, or when an event occurs. Typical tasks: nightly statistics gathering, RMAN backups, purging old data, refreshing materialized views and running batch reports.

It replaces the older `DBMS_JOB` package (still present for compatibility) and offers many more features:

- Jobs can run **PL/SQL blocks**, **stored procedures**, or **external programs / OS scripts**.
- Time-based schedules with a powerful calendar syntax, and **event-based** schedules.
- **Job classes** and **windows** tie jobs to Resource Manager, so batch work can be limited during busy hours.
- **Job chains** (steps with dependencies), logging of every run, e-mail notifications, and remote job execution.

Note: Oracle's automated maintenance tasks (optimizer statistics, segment advisor) are also run by the Scheduler in maintenance windows.

8.2 Access Rights (Privileges)

Users need suitable privileges before creating or running scheduler objects.

Privilege / role	Allows
CREATE JOB	Create jobs, schedules, programs and chains in one's own schema.
CREATE ANY JOB	Create jobs and related objects in any schema (very powerful — same as being able to run code as that user).

Privilege / role	Allows
CREATE EXTERNAL JOB	Create jobs that run OS executables or scripts outside the database.
EXECUTE ANY PROGRAM, EXECUTE ANY CLASS	Use programs or job classes owned by other users.
MANAGE SCHEDULER	Create job classes, windows and window groups; set scheduler attributes; stop any job.
SCHEDULER_ADMIN role	Includes the above privileges (with admin option); grant to trusted administrators only.
Object privileges	EXECUTE, ALTER, ALL on a specific job, program, schedule, chain or class.

```

GRANT CREATE JOB TO hr;
GRANT CREATE EXTERNAL JOB TO hr;
GRANT MANAGE SCHEDULER TO batch_admin;
GRANT EXECUTE ON sys.batch_class TO hr;           -- use a job class
GRANT ALTER ON hr.purge_old_logs_job TO ops_user; -- change one job

```

Listing 8.1: Granting scheduler privileges

Jobs run with the privileges of the job **owner**. Roles are not active inside jobs, so the owner needs direct grants on the objects the job uses. Views to check: DBA_SCHEDULER_JOBS, USER_SCHEDULER_JOBS, DBA_SYS_PRIVS.

8.3 Scheduler Components and Workflow

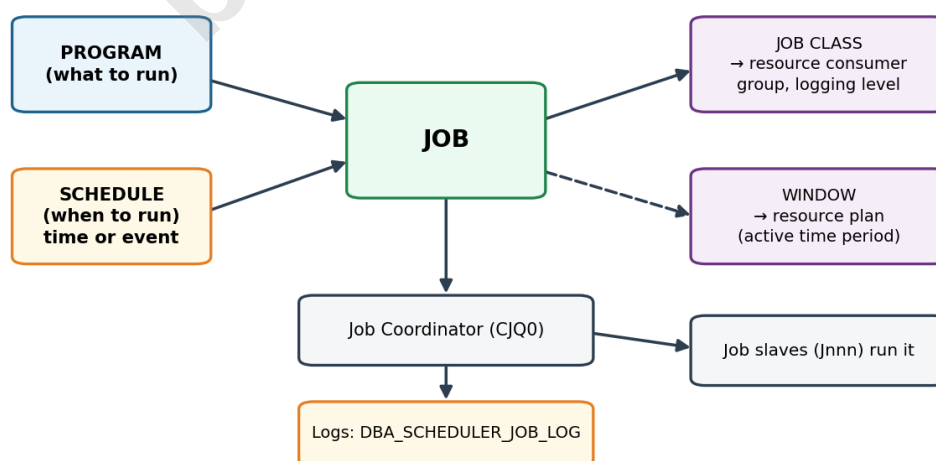


Figure 8.1: Main scheduler components and how a job is run

Component	Description
Job	The unit of work: <i>what</i> to run and <i>when</i> . Can be defined completely inside the job or by referencing a program and a schedule.
Program	A reusable definition of <i>what</i> is executed: a PL/SQL block, stored procedure or executable, with its arguments.
Schedule	A reusable definition of <i>when</i> : start date, end date and repeat interval (or an event).
Job class	A group of jobs sharing attributes: a Resource Manager consumer group, logging level, log retention and service.
Window	A time interval during which a specific Resource Manager plan becomes active (for example a nightly batch window).
Window group	A named list of windows.
Chain	A set of steps (programs) with rules about the order and conditions in which they run.
Event / queue	A message that starts a job; can be raised by an application, by the Scheduler itself, or by a file arrival.
Credential	Stores an OS user name and password used to run external jobs.

8.3.1 Workflow

- **1. Create a **program** (what) and a **schedule** (when), or define both directly in the job.**
- **2. Create a **job** and **enable** it** (jobs created with `enabled => TRUE`). Optionally assign a job class.
- **3. The **job coordinator** (process CJQ0) checks the job table, and when a job is due it starts a **job slave** process (Jnnn), which runs the job's action.**
- **4. The result is written to the log tables (DBA_SCHEDULER_JOB_LOG, DBA_SCHEDULER_JOB_RUN_DETAILS). The job's state is updated: for a repeating job it is rescheduled; a one-time job is completed (and may be dropped automatically).**

The parameter `JOB_QUEUE_PROCESSES` sets the maximum number of job slaves that can run at once (setting it to 0 stops jobs from running).

8.4 Creating a Job

The `DBMS_SCHEDULER.CREATE_JOB` procedure creates a job. Important parameters:

Parameter	Meaning
<code>job_name</code>	Name of the job (in the form <code>schema.name</code> if not your own schema).

Parameter	Meaning
job_type	PLSQL_BLOCK, STORED_PROCEDURE, EXECUTABLE, or CHAIN (also SQL_SCRIPT, BACKUP_SCRIPT in newer versions).
job_action	The PL/SQL code, procedure name or OS command path.
start_date, end_date	When the job may first / last run.
repeat_interval	Calendar expression that says how often to repeat. NULL means run once.
enabled	TRUE to make the job runnable immediately (default is FALSE).
auto_drop	TRUE (default) drops a job after it completes its schedule.
comments	Description.

```

BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
    job_name      => 'PURGE_OLD_LOGS_JOB',
    job_type      => 'PLSQL_BLOCK',
    job_action    => 'BEGIN DELETE FROM app_log WHERE log_date < SYSDATE - 30;
                     COMMIT; END;',
    start_date    => SYSTIMESTAMP,
    repeat_interval => 'FREQ=DAILY; BYHOUR=2; BYMINUTE=0; BYSECOND=0',
    enabled       => TRUE,
    comments      => 'Delete log rows older than 30 days, every night at 2:00';
  END;
/

```

Listing 8.2: A PL/SQL job that runs every night at 02:00

```

-- OS script job (needs CREATE EXTERNAL JOB; a credential is usual)
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
    job_name      => 'BACKUP_SCRIPT_JOB',
    job_type      => 'EXECUTABLE',
    job_action    => '/home/oracle/scripts/backup.sh',
    repeat_interval => 'FREQ=WEEKLY; BYDAY=SUN; BYHOUR=1',
    enabled       => TRUE);
  END;
/

```

Listing 8.3: An external job

8.4.1 Repeat Interval Calendar Syntax

Expression	Meaning
FREQ=DAILY; BYHOUR=2	Every day at 02:00
FREQ=HOURLY; INTERVAL=4	Every 4 hours

Expression	Meaning
FREQ=WEEKLY; BYDAY=MON,WED,FRI; BYHOUR=9	Monday, Wednesday and Friday at 09:00
FREQ=MONTHLY; BYMONTHDAY=1	First day of each month
FREQ=MONTHLY; BYMONTHDAY=-1	Last day of each month
FREQ=MINUTELY; INTERVAL=15	Every 15 minutes
FREQ=YEARLY; BYMONTH=JAN; BYMONTHDAY=1	Every 1 January

FREQ can be YEARLY, MONTHLY, WEEKLY, DAILY, HOURLY, MINUTELY or SECONDLY. Other keywords are INTERVAL, BYDAY, BYMONTHDAY, BYHOUR, BYMINUTE and BYSECOND.

8.4.2 Managing Jobs

```
EXEC DBMS_SCHEDULER.RUN_JOB('PURGE_OLD_LOGS_JOB');           -- run now
EXEC DBMS_SCHEDULER.DISABLE('PURGE_OLD_LOGS_JOB');
EXEC DBMS_SCHEDULER.ENABLE('PURGE_OLD_LOGS_JOB');
EXEC DBMS_SCHEDULER.STOP_JOB('PURGE_OLD_LOGS_JOB');          -- stop a running job
EXEC DBMS_SCHEDULER.SET_ATTRIBUTE('PURGE_OLD_LOGS_JOB',
    'repeat_interval','FREQ=DAILY;BYHOUR=3');
EXEC DBMS_SCHEDULER.DROP_JOB('PURGE_OLD_LOGS_JOB');

SELECT job_name, state, enabled, last_start_date, next_run_date
FROM   user_scheduler_jobs;
SELECT job_name, status, actual_start_date, run_duration, errors
FROM   user_scheduler_job_run_details ORDER BY log_date DESC;
```

Listing 8.4: Running, changing and monitoring jobs

Example 8.1

Schedule a stored procedure `hr.refresh_summary` to run at 6:00 AM every Monday to Friday.

Solution

```
job_type => 'STORED_PROCEDURE', job_action => 'HR.REFRESH_SUMMARY',
repeat_interval => 'FREQ=WEEKLY; BYDAY=MON,TUE,WED,THU,FRI; BYHOUR=6;
BYMINUTE=0; BYSECOND=0', enabled => TRUE.
```

8.5 Job Classes

A **job class** groups jobs that have similar requirements. It links jobs to a Resource Manager **consumer group** so that CPU is shared according to the active resource plan, and controls logging for its jobs. All jobs belong to a class; if none is specified, the job is in `DEFAULT_JOB_CLASS`. Creating a class requires the `MANAGE_SCHEDULER` privilege; the class is owned by `SYS`.

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB_CLASS (
    job_class_name      => 'BATCH_CLASS',
    resource_consumer_group => 'LOW_GROUP',      -- an existing consumer group
    logging_level       => DBMS_SCHEDULER.LOGGING_FULL,
    log_history         => 30,                  -- days to keep log entries
    comments            => 'Low priority batch jobs');
END;
/
-- assign a job to the class
EXEC DBMS_SCHEDULER.SET_ATTRIBUTE('PURGE_OLD_LOGS_JOB', 'job_class', 'BATCH_CLASS');

GRANT EXECUTE ON BATCH_CLASS TO hr;
SELECT job_class_name, resource_consumer_group, logging_level
FROM   dba_scheduler_job_classes;
```

Listing 8.5: Creating and using a job class

Logging level	What is recorded
LOGGING_OFF	No log entries.
LOGGING_RUNS	Detailed entry for each run of a job (default level).
LOGGING_FULL	Every operation on the job (create, enable, run, change, drop) and each run.

8.6 Time-Based and Event-Based Schedules

8.6.1 Time-Based Schedules

A **schedule** stored as an object can be reused by many jobs. Changing the schedule changes all the jobs that use it.

```

BEGIN
  DBMS_SCHEDULER.CREATE_SCHEDULE (
    schedule_name    => 'NIGHTLY_2AM',
    start_date       => SYSTIMESTAMP,
    repeat_interval  => 'FREQ=DAILY; BYHOUR=2; BYMINUTE=0; BYSECOND=0',
    comments         => 'Every night at 02:00');

  DBMS_SCHEDULER.CREATE_PROGRAM(
    program_name     => 'GATHER_STATS_PROG',
    program_type     => 'STORED_PROCEDURE',
    program_action    => 'DBMS_STATS.GATHER_SCHEMA_STATS',
    number_of_arguments => 1,
    enabled          => FALSE);
  DBMS_SCHEDULER.DEFINE_PROGRAM_ARGUMENT ('GATHER_STATS_PROG', 1,
                                          'ownname', 'VARCHAR2', 'HR');
  DBMS_SCHEDULER.ENABLE ('GATHER_STATS_PROG');

  DBMS_SCHEDULER.CREATE_JOB (
    job_name         => 'GATHER_HR_STATS_JOB',
    program_name     => 'GATHER_STATS_PROG',
    schedule_name    => 'NIGHTLY_2AM',
    enabled          => TRUE);
END;
/

```

Listing 8.6: Job built from a program and a schedule

A **window** is a special schedule that also activates a resource plan while it is open. Windows are created with `DBMS_SCHEDULER.CREATE_WINDOW` (requires `MANAGE_SCHEDULER`) and are used, for example, to give batch jobs more resources at night.

8.6.2 Event-Based Schedules

An **event-based** job starts when something happens instead of at a fixed time. Examples: a new order message arrives on a queue, another job completes, or a file appears on a system. Events are delivered as messages on an **Oracle Advanced Queuing (AQ)** queue. There are three kinds of events:

- **Application events** — raised by your own code by placing a message on a queue.
- **Scheduler events** — raised by the Scheduler itself (for example `JOB_STARTED`, `JOB_SUCCEEDED`, `JOB_FAILED`) when a job has the `raise_events` attribute.
- **File arrival events** — raised by a file watcher when a file appears (requires the Scheduler agent for remote hosts).

8.6.3 Creating an Event-Based Schedule (steps)

- **1.** Create a payload type that describes the event message.
- **2.** Create and start a queue table and queue (DBMS_AQADM); the user needs AQ privileges (e.g. AQ_ADMINISTRATOR_ROLE or specific grants) and a tablespace quota.
- **3.** Create an event schedule with CREATE_EVENT_SCHEDULE, giving the queue and an optional condition.
- **4.** Create a job that uses this schedule (or specify queue_spec and event_condition directly in the job).
- **5.** Application code enqueues a message (DBMS_AQ.ENQUEUE); when it matches the condition, the job runs.

```
-- 1. payload type
CREATE TYPE order_event_t AS OBJECT (order_id NUMBER, status VARCHAR2(20));
/
-- 2. queue table and queue
BEGIN
    DBMS_AQADM.CREATE_QUEUE_TABLE(queue_table => 'order_evt_qt',
                                  queue_payload_type => 'ORDER_EVENT_T');
    DBMS_AQADM.CREATE_QUEUE(queue_name => 'order_evt_q',
                             queue_table => 'order_evt_qt');
    DBMS_AQADM.START_QUEUE(queue_name => 'order_evt_q');
END;
/
-- 3. event schedule (runs a job only for messages with status NEW)
BEGIN
    DBMS_SCHEDULER.CREATE_EVENT_SCHEDULE(
        schedule_name => 'NEW_ORDER_SCHED',
        start_date    => SYSTIMESTAMP,
        event_condition => 'tab.user_data.status = ''NEW''',
        queue_spec     => 'order_evt_q');
END;
/
-- 4. job triggered by the event
BEGIN
    DBMS_SCHEDULER.CREATE_JOB(
        job_name       => 'PROCESS_ORDER_JOB',
        job_type        => 'STORED_PROCEDURE',
        job_action      => 'PROCESS_NEW_ORDERS',
        schedule_name   => 'NEW_ORDER_SCHED',
        enabled         => TRUE);
END;
/
```

Listing 8.7: Event-based schedule and job (outline)

Note: By default an event-based job runs once for each matching message. PROCESS_NEW_ORDERS is a procedure you must create yourself, and the code that enqueues messages is application-specific. Details vary by release; check the Scheduler chapter of the Administrator's Guide.

To have the Scheduler itself raise events about a job (for example to start a cleanup job when it fails), set `raise_events` on the job, e.g.

DBMS_SCHEDULER.SET_ATTRIBUTE('MY_JOB','raise_events',
DBMS_SCHEDULER.JOB_FAILED);, and subscribe to the scheduler event queue.

Point	Time-based schedule	Event-based schedule
Trigger	A date/time or repeating calendar interval	A message on a queue or a file arrival
Main attribute	repeat_interval	queue_spec, event_condition
Typical use	Nightly backup, monthly report	Process an order when it arrives
Extra requirement	None	AQ queue and payload type

Example 8.2

Which view would you query to find out why last night's job failed?

Solution

```
SELECT job_name, status, actual_start_date, errors, additional_info
FROM dba_scheduler_job_run_details WHERE job_name =
'PURGE_OLD_LOGS_JOB' ORDER BY log_date DESC;
```

The ERRORS and ADDITIONAL_INFO columns show the error message. Also check the alert log and job trace files.

Key Takeaways

- ✓ The **Scheduler** (DBMS_SCHEDULER) automates tasks: PL/SQL blocks, procedures and OS executables; it replaces DBMS_JOB.
- ✓ Privileges: CREATE JOB, CREATE EXTERNAL JOB, MANAGE SCHEDULER (classes and windows), or the SCHEDULER_ADMIN role.
- ✓ Components: **job, program, schedule, job class, window, chain, event**. Coordinator CJQO starts job slaves Jnnn; JOB_QUEUE_PROCESSES limits them.
- ✓ CREATE_JOB needs job_type, job_action and either repeat_interval or a schedule; manage with RUN_JOB, ENABLE, DISABLE, STOP_JOB, DROP_JOB.
- ✓ A **job class** links jobs to a resource consumer group and controls logging.
- ✓ **Time-based** schedules use calendar syntax (FREQ=...); **event-based** schedules use an AQ queue, queue_spec and event_condition.

Reminder: these notes may contain errors — cross-check with the official syllabus, your textbook and Oracle documentation.

— End of Unit 8 —