Scheduler[1]

Gli addetti vendite del Music Store sono abili venditori e tra loro vi è una gran competizione.
Il supervisore del reparto vendite intende scegliere il migliore ed offrirgli un'importante promozione.
Il criterio scelto consiste nella vendita del maggior numero di articoli in un periodo prefissato.
Ogni primo lunedì del mese si conclude la challenge ed è nominato "addetto del mese" il venditore che vanta il maggior numero di vendite in tale periodo.

In modo sintetico è possibile considerare un job come un programma che è eseguito coerentemente alla frequenza dettata dallo schedule.
Nel nostro caso, data la semplicità dell'operazione, sarebbe stato possibile definire direttamente il programma e lo schedule nel job stesso (shortcut), tuttavia si è preferito seguire l'iter consigliato da Oracle:

1. Creazione di un programma, ovvero un blocco PL/SQL, una procedura o uno shell script.
2. Definizione dello schedule (detta la frequenza d'esecuzione del programma).
3. Creazione del job, includendo il nome del programma e lo schedule creati precedentemente.

Tabella

E' stata creata una tabella supplementare per l'inserimento dei dati relativi il numero di articoli venduti e il fatturato generato dall'addetto.

```
CREATE SEQUENCE add_mese_seq START WITH 1;
CREATE TABLE addetto_del_mese (
id_add_mese              NUMBER(3,0)    DEFAULT add_mese_seq.nextval NOT NULL,
cod_tess_add_mese        NUMBER(5)      NOT NULL,
no_vendite               NUMBER(3,0)    NOT NULL,
fatturato                NUMBER(5,0)    NOT NULL,
mese                     VARCHAR2(20)   NOT NULL,
anno                     NUMBER(4,0)    NOT NULL,
CONSTRAINT pk_add_mese PRIMARY KEY (id_add_mese)
);
```

1 Nota: prima della release 10g di Oracle i vari task erano eseguiti e schedulati via crontab in Linux. La versione 10g ha introdotto il DBMS_SCHEDULER che semplifica la schedulazione di jobs ed introduce diverse funzionalità (indipendenza dall'OS, esecuzione di jobs su macchine remote v.11g e successive).

Procedura

Per motivi di eleganza, si è scelto includere il blocco PL/SQL (il cuore dell'operazione) nella procedura "nomina_addetto_mese" e non direttamente nella program_action.
Tale procedura recupera l'addetto con il maggior numero di vendite registrate nel corso del mese precedente (a quello corrente), il relativo fatturato generato ed il numero di articoli venduti e inserisce la tupla nella tabella creata precedentemente (addetto_del_mese).
E' prevista un'eccezione nel caso in cui non è stata effettuata alcuna vendita nel mese precedente (lo Store era verosimilmente in ferie).

```
CREATE OR REPLACE PROCEDURE NOMINA_ADDETTO_DEL_MESE
AS
    -- Il mese target è il mese precedente
    mese_target VARCHAR2(15) := to_char(add_months(sysdate, -1), 'mon');
    -- Anno relativo al mese precedente (fa la differenza se il mese corrente è
gennaio)
    anno_target NUMBER := extract(year from to_date(add_months(sysdate, -1)));
    addetto NUMBER(5);
    no_vendite NUMBER(3);
    fatturato NUMBER(7,2);
BEGIN
/*
La query seleziona l'addetto che ha effettuato più vendite, il numero di  vendite
ed il fatturato nel corso del mese_target
*/
    SELECT * INTO addetto, fatturato, no_vendite FROM (
       SELECT cod_tess_add_vendita, sum(prezzo), count(*)
       FROM acquisto ACQ, articolo A JOIN dettaglio_acquisto DA
       ON A.cod_art=DA.cod_art_det
        WHERE ACQ.cod_scon = DA.cod_scon_det AND
       to_char(data_a, 'mon') = lower(mese_target) AND
       extract(year from data_a) = anno_target
              GROUP BY cod_tess_add_vendita ORDER BY count(*) DESC)
    WHERE rownum=1;
    -- Inseriamo in addetto_del_mese
    INSERT INTO addetto_del_mese(cod_tess_add_mese, no_vendite, fatturato, mese,
    anno)
    VALUES (addetto, no_vendite, fatturato, mese_target, anno_target);
    EXCEPTION
    WHEN NO_DATA_FOUND THEN
       RAISE_APPLICATION_ERROR(-20050, 'Nessuna vendita in tale periodo, lo Store
       è chiuso causa ferie!');
END NOMINA_ADDETTO_DEL_MESE;
```

Schedule

Come da requisiti la frequenza è fissata al primo Lunedì del mese.
L'inizio è posto al 1 Ottobre poiché Settembre risulta essere il primo mese lavorativo dopo il rientro dalle ferie estive.

```
BEGIN
    DBMS_SCHEDULER.CREATE_SCHEDULE (
        repeat_interval  => 'FREQ=MONTHLY;BYDAY=1MON',
        start_date => TO_TIMESTAMP_TZ('2020-10-01 12:00:00.000000000
EUROPE/BERLIN','YYYY-MM-DD HH24:MI:SS.FF TZR'),
        comments => 'Runs every 1st monday',
        schedule_name  => 'EVERY_MONTH');
END;
```

Job

Segue l'implementazione del job sulla base del programma e dello schedule definiti precedentemente.

```
BEGIN
    DBMS_SCHEDULER.CREATE_JOB (
            job_name => 'JOB_ADDETTO_DEL_MESE',
            program_name => 'P_ADDETTO_DEL_MESE',
            schedule_name => 'EVERY_MONTH',
            enabled => TRUE,
            auto_drop => FALSE,
            comments => 'Job based on p_addetto_del_mese that runs
            every_month');
END;
```

Esecuzione manuale (testing porpouse)

Ricordiamo ancora una volta che il job è eseguito in modo asincrono (la frequenza è imposta dallo schedule), attraverso tale esecuzione manuale si testa il corretto funzionamento dello stesso.

```
BEGIN
  -- Run job
  DBMS_SCHEDULER.run_job (job_name            => 'job_addetto_del_mese',
                          use_current_session => TRUE);
  -- Stop jobs.
  DBMS_SCHEDULER.stop_job (job_name => 'job_addetto_del_mese');
END;
```