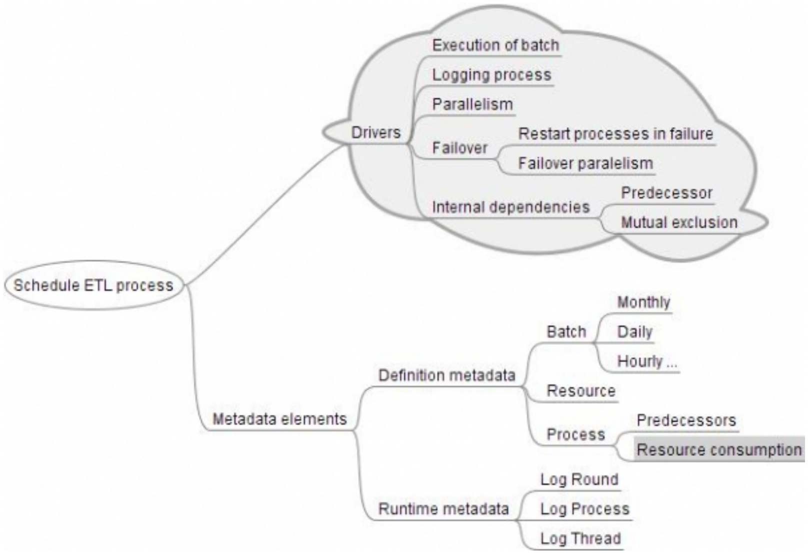
ETL PROCESS SCHEDULING - SAMPLE SOLUTION FOR ORACLE SIMPLETASK

HEAP WORKFLOW

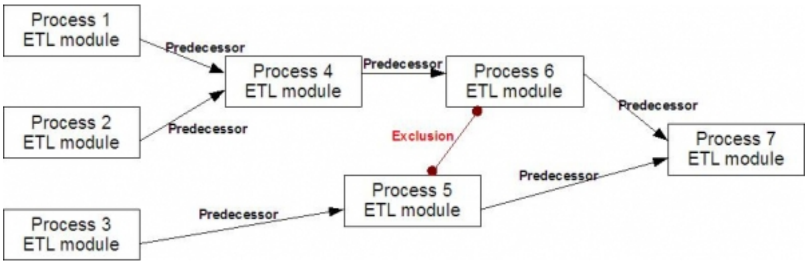
22-Sep-2008

Implementing ETL or ELT processes based on PL/SQL modules there is usually one important task in the queue - Schedule execution of atomic modules following internal dependencies and another rules. Enterprise-wide schedulers usually do not fulfill requirements of fine grain dependencies and rules, so they are often used as trigger of ETL batch managed by local ETL scheduler. Most of schedulers built-in into standard ETL tools are too static and rigid, not scaleable. Following solution I call SIMPLETASK, because it is simple by use method despite of teh powers it covers. Following sample is the BASIC version of the solution. Basic, to allow understand better the principle and to be easily included into simplier solutions like small ETL or migration processes.



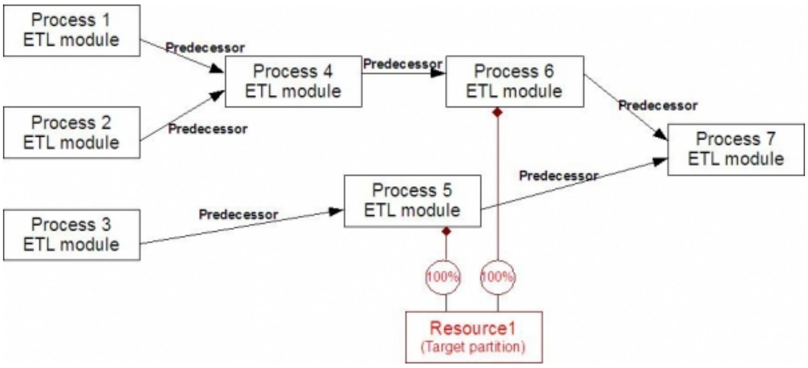
Solution of the scheduling process come from the standard network graph task with resources. It is possible share a lot of methods with another solutions (Finding critical path etc.)

- **Heap** (Batch) - defines closed group of processes run together driven by internal dependencies.
- **Task** (Process) - defines atomic module to be run.
- **Predecessor** - defines dependency between two modules, the second one should not be run before the first one is completed.
- **Exclusion** - defines dependency between two modules, any of them should not be run when another process is in progress. *(Note - in our solution exclusions are solved using resources)*
- **Resource** - defines limited resource consumed by running processes.
- **Resource consumption** - defines consumption of resource by particular process.
- **Round** (Run) - defines one cycle of execution, for example a processed day of daily process.



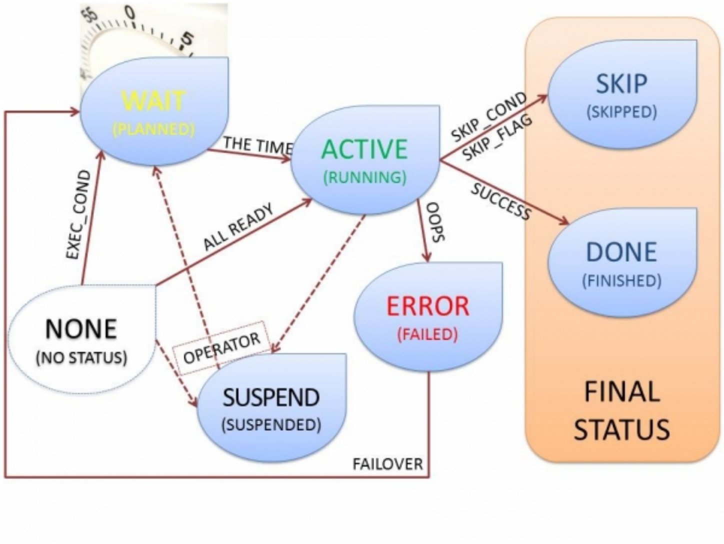
Example above shows dependencies between processes including both predecessor and exclusion types.

Example below substitutes exclusion dependency with resources. Resources allows either full exclusion (consumption 100%) or limit number of huge modules at the same time (consumption 50% allows processing up to 2 such processes).



See: More about the Heap Workflow principles you can read here

State Diagram of task in SIMPLETASK solution



DOWNLOAD SIMPLE SOLUTION

Following downloads, one for metadata structure and second one for PL/SQL packages, allow you to run your ETL or Data Migration tasks.
To install follow these steps:

1. Create database user (e.g. BWTA_OWNER)
2. Allow database space quota to store metadata (not extensive space required)
3. Grant following privileges to the user:
   - *CREATE SESSION*
   - *CREATE TABLE*
   - *CREATE TRIGGER*
   - *CREATE VIEW*
   - *CREATE SEQUENCE*
   - *CREATE JOB*
   - *CREATE PROCEDURE*
   - *EXECUTE ON DBMS_LOCK*
4. Install metadata structures running the first following script downoad
5. Install packages running the second following script download
6. Install views running the third following script download

Script: Create_BWTA_metadata_structure.sql

```
--------------------------------------------------------
--  DDL for Sequence BWTA_HEAP_SEQ
--------------------------------------------------------
   CREATE SEQUENCE  "BWTA_HEAP_SEQ"  MINVALUE 1 INCREMENT BY 1 START WITH 1 NOCACHE  ORDER  NOCYCLE;
--------------------------------------------------------
--  DDL for Sequence BWTA_LOG_ROUND_SEQ
--------------------------------------------------------
   CREATE SEQUENCE  "BWTA_LOG_ROUND_SEQ"  MINVALUE 1 INCREMENT BY 1 START WITH 1 NOCACHE  ORDER  NOCYCLE;
--------------------------------------------------------
--  DDL for Sequence BWTA_LOG_THREAD_SEQ
```

...more

DOWNLOAD

Script: Create_BWTA_core_packages.sql

```
----------------------------------------------------------------------
-- PACKAGE BWTA_METADATA                                            --
----------------------------------------------------------------------
Create or replace PACKAGE "BWTA_METADATA" IS
  ---------------------------------------------------------------------------
  --Purpose: Simple processes and task management / METADATA API          --
  --Author:  Bob Jankovsky, copyleft 2008, 2013                            --
  --History: 1.3 /22-JUN-2013 -- extracted from central utility and enhanced --
  --         1.4 /24-JUL-2013 -- new Rename methods                        --
  ---------------------------------------------------------------------------
```

...more

DOWNLOAD

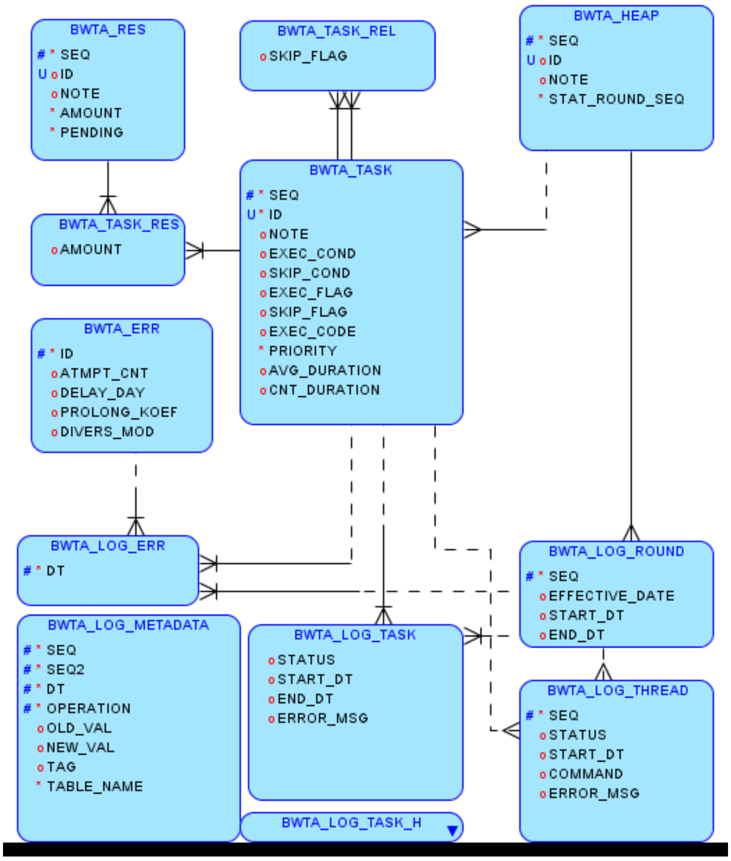
Script: Create_BWTA_core_views.sql

```
----------------------------------------------------------------------
-- View BWTA_V_ERR                                                  --
----------------------------------------------------------------------
CREATE OR REPLACE FORCE VIEW BWTA_V_ERR AS
SELECT
 'ORA'||to_char(ID,'00000') as ID,
 ATMPT_CNT,
 DELAY_DAY*24*60 AS DELAY_MIN,
 PROLONG_KOEF,
 DIVERS_MOD
```

...more

DOWNLOAD

METADATA MODEL



Metadata structure description:

BWTA_TASK_RES	**Relation between processes - predecessor specification**		
TASK_SEQ	NUMBER	1	Sequence key of task
RES_SEQ	NUMBER	2	Sequence key of resource
AMOUNT	NUMBER	O	Consumption of specified resource by the process

BWTA_TASK_REL	**Relation between processes - predecessor specification**		
SEQ1	NUMBER	1	Sequence key of dependent task
SEQ2	NUMBER	2	Sequence key of the predecessor task
SKIP_FLAG	NUMBER(1)	O	Skip flag to disable dependency

BWTA_TASK	**Process (task) to be run**		
SEQ	NUMBER	1	Sequence key of task
HEAP_SEQ	NUMBER	M	Sequence key of heap
ID	VARCHAR2(100)	M	ID of process
NOTE	VARCHAR2(500)	O	Description of task
EXEC_COND	VARCHAR2(2000)	O	Condition code of execution to block task based on external condition
SKIP_COND	VARCHAR2(2000)	O	Condition code of skip to skip task based on external condition
EXEC_FLAG	NUMBER	O	Flag of execution to block task
SKIP_FLAG	NUMBER	O	Flag of skip to skip task
EXEC_CODE	VARCHAR2(2000)	O	Execution PL/SQL code
PRIORITY	NUMBER	M	Priority of execution (less is more important)
AVG_DURATION	NUMBER	O	Average duration of process based on statistics
CNT_DURATION	NUMBER	O	Count of runs involved in statistics

BWTA_RES	**Resource definition**		
SEQ	NUMBER	1	Sequence key of resource
ID	VARCHAR2(100)	O	ID of resource
NOTE	VARCHAR2(500)	O	Description of resource
AMOUNT	NUMBER	M	Total amount of resource (e.g. 100 [%])
PENDING	NUMBER(1)	M	0..resource released after end of process, 1..resource should be released by negative consumption

BWTA_LOG_THREAD	**Thread of execution**		
SEQ	NUMBER	1	Sequence key of thread
ROUND_SEQ	NUMBER	O	Sequence key of round
STATUS	VARCHAR2(10)	O	Status of thread (ACTIVE, INACTIVE, SLEEP)
TASK_SEQ	NUMBER	O	Sequence key of active task for the ACTIVE status
START_DT	DATE	O	Start date of current activity
COMMAND	VARCHAR2(100)	O	command for thread - STOP to stop thread after current activity
ERROR_MSG	VARCHAR2(2000)	O	Error message for the ERROR status

BWTA_LOG_TASK_H	**Process execution log**		
TASK_SEQ	NUMBER	O	Sequence key of process
ROUND_SEQ	NUMBER	O	Sequence key of round
STATUS	VARCHAR2(10)	O	Status of process (ACTIVE, ERROR, DONE, SUSPEND, SKIP, WAIT)
START_DT	DATE	O	Start date and time of process execution or time to wait for
END_DT	DATE	O	End date and time of process execution
ERROR_MSG	VARCHAR2(4000)	O	Error message for the ERROR status
TS	TIMESTAMP(6)	O	Timestamp of the change

BWTA_LOG_TASK	**Process execution log**		
TASK_SEQ	NUMBER	1	Sequence key of process
ROUND_SEQ	NUMBER	2	Sequence key of round

STATUS	VARCHAR2(10)	O	Status of process (ACTIVE, ERROR, DONE, SUSPEND, SKIP, WAIT)
START_DT	DATE	O	Start date and time of process execution or time to wait for
END_DT	DATE	O	End date and time of process execution
ERROR_MSG	VARCHAR2(4000)	O	Error message for the ERROR status

BWTA_LOG_ROUND	**Round of the batch execution**		
SEQ	NUMBER	1	Sequence key of round
HEAP_SEQ	NUMBER	M	Sequence key of heap
EFFECTIVE_DATE	DATE	O	Effective date of round
START_DT	DATE	O	Start date and time of round
END_DT	DATE	O	End date and time of round - indicates completed rounds

BWTA_LOG_METADATA			
SEQ	NUMBER	1	Primary key of changed record
SEQ2	NUMBER	2	Optional extension of key of changed record
DT	TIMESTAMP(6)	3	Timestamp of realized change
OPERATION	CHAR(1)	4	Operation (I,U,D)
OLD_VAL	XMLTYPE	O	Old value of record XML element
NEW_VAL	XMLTYPE	O	New value of record XML element
TAG	VARCHAR2(100)	O	
TABLE_NAME	VARCHAR2(30)	M	Table of the change

BWTA_LOG_ERR	**Log of all the realized failover actions**		
ROUND_SEQ	NUMBER	1	Round when it happened
TASK_SEQ	NUMBER	2	Sequence key of Task
ERR_ID	NUMBER	3	Error ID
DT	TIMESTAMP(6)	4	Timestamp when it happened

BWTA_HEAP	**Batch of processes to be run**		
SEQ	NUMBER	1	Sequence key of batch, -1 is default one
ID	VARCHAR2(100)	O	ID of batch
NOTE	VARCHAR2(500)	O	Description of batch
STAT_ROUND_SEQ	NUMBER	M	Last round the statistics has been gathered

BWTA_ERR	**Maintained errors**		
ID	NUMBER	1	ORA error number
ATMPT_CNT	NUMBER	O	Number of attempts
DELAY_DAY	NUMBER	O	Delay specified as a fragment of day
PROLONG_KOEF	NUMBER	O	Koefficient of prolongation of each next attempt
DIVERS_MOD	NUMBER	O	Modulo used for diversification of particular processes

EXAMPLE

In following example of filling metadata we will use processes from the network graph above.
Begin DBMS_LOCK.sleep([duration]);end; will be used instead of real modules to simulate duration of processes.

```
BEGIN
     BWTA_METADATA.SETRES('R1','Resource 1 - target partition',100,0,'INIT');
     BWTA_METADATA.SETTASK('P1', 'Task1', 'begin DBMS_LOCK.sleep(120);end;', 50, 'INIT');
     BWTA_METADATA.SETTASK('P2', 'Task2', 'begin DBMS_LOCK.sleep(140);end;', 50, 'INIT');
     BWTA_METADATA.SETTASK('P3', 'Task3', 'begin DBMS_LOCK.sleep(250);end;', 50, 'INIT');
     ----
     BWTA_METADATA.SETTASK('P4', 'Task4', 'begin DBMS_LOCK.sleep(160);end;', 50, 'INIT');
       BWTA_METADATA.SETTASKREL('P4','P1',-1,0,'INIT'); --dependency P4 on P1
       BWTA_METADATA.SETTASKREL('P4','P2',-1,0,'INIT'); --dependency P4 on P2
     BWTA_METADATA.SETTASK('P5', 'Task5', 'begin DBMS_LOCK.sleep(100);end;', 50, 'INIT');
       BWTA_METADATA.SETTASKREL('P5','P3',-1,0,'INIT'); --dependency P5 on P3
       BWTA_METADATA.SetTaskRes('P5',-1,'R1',100,'INIT'); --Task P5 requires resource R1 (amount 100)
     ----
     BWTA_METADATA.SETTASK('P6', 'Task6','begin DBMS_LOCK.sleep(300);end;', 50, 'INIT');
       BWTA_METADATA.SETTASKREL('P6','P4',-1,0,'INIT'); --dependency P6 on P4
       BWTA_METADATA.SetTaskRes('P6',-1,'R1',100,'INIT'); --Task P5 requires resource R1 (amount 100)
     ----
     BWTA_METADATA.SetTask('P7', 'Task7', 'begin DBMS_LOCK.sleep(309);end;', 50, 'INIT');
       BWTA_METADATA.SETTASKREL('P7','P5',-1,0,'INIT'); --dependency P7 on P5
       BWTA_METADATA.SetTaskRel('P7','P6',-1,0,'INIT'); --dependency P7 on P6
    Commit;
end;
/
```

[Download]

Now we can start daily process using following command:

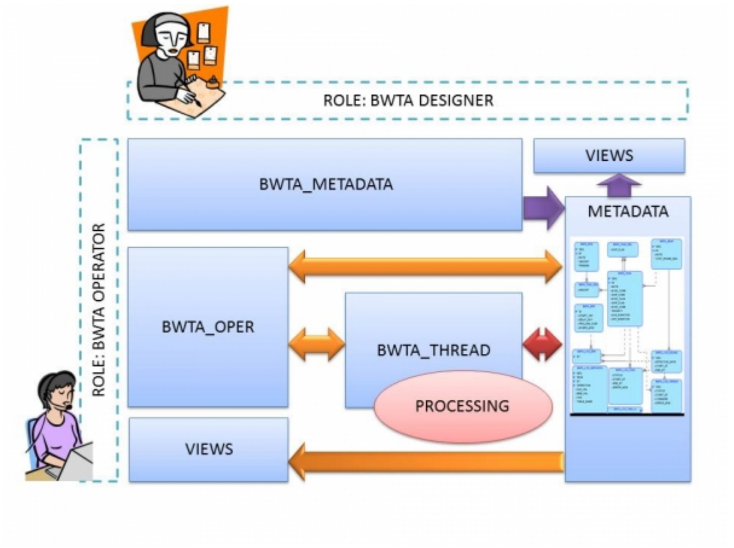
```
BEGIN
  bwta_oper.startround(DATE'2008-08-22');
END;
/
```

Process is started in 5 parallel threads for an effective date 22-AUG-2008. We can operatively check execution from runtime metadata. Following list contains snapshots taken during the execution from the very start to the end. Column BLOCKING_TASKS hints what are particular tasks waiting for

```
SQL> SELECT
  2    TASK_ID
  3  , EFFECTIVE_DATE
  4  , STATUS
  5  , START_DT
  6  , END_DT
  7  , MINUTES
  8  , AVG_MINUTES
  9  , PREDECESSOR_TASKS
 10  , BLOCKING_TASKS
 11  FROM BWTA_V_LOG_TASK_WIDE
 12  WHERE HEAP_SEQ=-1
 13  /

TASK_ID              EFFECTIV STATUS     START_DT END_DT       MINUTES AVG_MINUTES PREDECESSOR_TASKS    BLOCKING_TASKS
-------------------- -------- ---------- -------- -------- ---------- ----------- -------------------- --------------------
P1                   22.08.08 ACTIVE     29.06.13          ,133333333  2,03541667
P2                   22.08.08 ACTIVE     29.06.13                 ,05  2,33333333
P3                   22.08.08 ACTIVE     29.06.13          ,033333333  4,16666667
P4                   22.08.08                                               5,175 P1,P2                P1,P2
P5                   22.08.08                                               1,675 P3                   P3
P6                   22.08.08                                                   5 P4                   P4
P7                   22.08.08                                          5,15833333 P5,P6                P5,P6

SQL> r
```

```
TASK_ID              EFFECTIV STATUS     START_DT END_DT      MINUTES AVG_MINUTES PREDECESSOR_TASKS    BLOCKING_TASKS
-------------------- -------- ---------- -------- -------- ---------- ----------- -------------------- --------------------
P1                   22.08.08 DONE       29.06.13 29.06.13       2,05  2,03541667
P2                   22.08.08 DONE       29.06.13 29.06.13 2,33333333  2,33333333
P3                   22.08.08 ACTIVE     29.06.13                2,55  4,16666667
P4                   22.08.08 ACTIVE     29.06.13          ,233333333       5,175 P1,P2
P5                   22.08.08                                                1,675 P3                   P3
P6                   22.08.08                                                    5 P4                   P4
P7                   22.08.08                                           5,15833333 P5,P6                P5,P6

SQL> r

TASK_ID              EFFECTIV STATUS     START_DT END_DT      MINUTES AVG_MINUTES PREDECESSOR_TASKS    BLOCKING_TASKS
-------------------- -------- ---------- -------- -------- ---------- ----------- -------------------- --------------------
P1                   22.08.08 DONE       29.06.13 29.06.13       2,05  2,03541667
P2                   22.08.08 DONE       29.06.13 29.06.13 2,33333333  2,33333333
P3                   22.08.08 DONE       29.06.13 29.06.13 4,16666667  4,16666667
P4                   22.08.08 ACTIVE     29.06.13          2,01666667       5,175 P1,P2
P5                   22.08.08 ACTIVE     29.06.13          ,166666667       1,675 P3
P6                   22.08.08                                                    5 P4                   P4
P7                   22.08.08                                           5,15833333 P5,P6                P5,P6

SQL> r

TASK_ID              EFFECTIV STATUS     START_DT END_DT      MINUTES AVG_MINUTES PREDECESSOR_TASKS    BLOCKING_TASKS
-------------------- -------- ---------- -------- -------- ---------- ----------- -------------------- --------------------
P1                   22.08.08 DONE       29.06.13 29.06.13       2,05  2,03541667
P2                   22.08.08 DONE       29.06.13 29.06.13 2,33333333  2,33333333
P3                   22.08.08 DONE       29.06.13 29.06.13 4,16666667  4,16666667
P4                   22.08.08 DONE       29.06.13 29.06.13 2,66666667       5,175 P1,P2
P5                   22.08.08 ACTIVE     29.06.13          1,56666667       1,675 P3
P6                   22.08.08                                                    5 P4
P7                   22.08.08                                           5,15833333 P5,P6                P5,P6

SQL> r

TASK_ID              EFFECTIV STATUS     START_DT END_DT      MINUTES AVG_MINUTES PREDECESSOR_TASKS    BLOCKING_TASKS
-------------------- -------- ---------- -------- -------- ---------- ----------- -------------------- --------------------
P1                   22.08.08 DONE       29.06.13 29.06.13       2,05  2,03541667
P2                   22.08.08 DONE       29.06.13 29.06.13 2,33333333  2,33333333
P3                   22.08.08 DONE       29.06.13 29.06.13 4,16666667  4,16666667
P4                   22.08.08 DONE       29.06.13 29.06.13 2,66666667       5,175 P1,P2
P5                   22.08.08 DONE       29.06.13 29.06.13 1,66666667       1,675 P3
P6                   22.08.08 ACTIVE     29.06.13          ,883333333           5 P4
P7                   22.08.08                                           5,15833333 P5,P6                P6

SQL> r

TASK_ID              EFFECTIV STATUS     START_DT END_DT      MINUTES AVG_MINUTES PREDECESSOR_TASKS    BLOCKING_TASKS
-------------------- -------- ---------- -------- -------- ---------- ----------- -------------------- --------------------
P1                   22.08.08 DONE       29.06.13 29.06.13       2,05  2,03541667
P2                   22.08.08 DONE       29.06.13 29.06.13 2,33333333  2,33333333
P3                   22.08.08 DONE       29.06.13 29.06.13 4,16666667  4,16666667
P4                   22.08.08 DONE       29.06.13 29.06.13 2,66666667       5,175 P1,P2
P5                   22.08.08 DONE       29.06.13 29.06.13 1,66666667       1,675 P3
P6                   22.08.08 DONE       29.06.13 29.06.13          5           5 P4
P7                   22.08.08 ACTIVE     29.06.13          ,133333333  5,15833333 P5,P6
```

SECURITY ARCHITECTURE



The most common tasks of the workflow management:

Monitoring of current processes:

```
SELECT
  TASK_ID
, EFFECTIVE_DATE
, STATUS
, to_char(START_DT,'DD.MM.YYYY HH24:MI:SS') AS START_DT
, to_char(END_DT,'DD.MM.YYYY HH24:MI:SS') AS START_DT
, END_DT
, MINUTES
, AVG_MINUTES
, PREDECESSOR_TASKS
, BLOCKING_TASKS
FROM BWTA_V_LOG_TASK_WIDE
WHERE HEAP_SEQ=-1
ORDER BY
  CASE STATUS WHEN 'ERROR' THEN 1
              WHEN 'SUSPEND' THEN 2
              WHEN 'ACTIVE' THEN 3
              WHEN 'DONE' THEN 5
              WHEN 'SKIP' THEN 5
              ELSE 4
  END
, TASK_ID
```

Error treatment *the first column returns command to restart process*

```
SELECT 'BEGIN BWTA_OPER.restartTask('''||TASK_ID||''');END;' as CMD
, ERROR_MSG
FROM BWTA_V_LOG_TASK_WIDE
WHERE status = 'ERROR'
ORDER BY START_DT
```

List of threads of current round(s)

```
SELECT b.seq
, b.round_seq
, b.status
, TO_CHAR(d.start_dt, 'DD-MM-YYYY HH24:MI:SS') start_dt
, b.task_seq
, c.id
, b.command
, b.error_msg
FROM BWTA_LOG_THREAD B
LEFT JOIN BWTA_TASK C
ON B.TASK_SEQ = C.SEQ
LEFT JOIN BWTA_LOG_TASK d
ON d.task_seq = c.seq
  AND d.round_seq = b.round_seq
ORDER BY b.seq
```

```
 SEQ  ROUND_SEQ STATUS     START_DT              TASK_SEQ ID           COMMAND           ERROR_MSG
---- ---------- ---------- ------------------- ---------- ------------ ----------------- --------------------
  46         20 ACTIVE     29-06-2013 16:49:24         13 P1
  47         20 ACTIVE     29-06-2013 16:49:27         14 P2
  48         20 INACTIVE
  49         20 SLEEP
  50         20 SLEEP
```

Number of processes by status (done, to be done, error)

```
WITH LRL AS
  (
    SELECT --+materialize
      HEAP_SEQ
    , MAX(SEQ) ROUND_SEQ
    FROM BWTA_LOG_ROUND
    GROUP BY HEAP_SEQ
  )
,STAT AS
  (
    SELECT
      T.HEAP_SEQ
     ,NVL(LT.STATUS,'WAIT') AS STATUS
     ,LRL.ROUND_SEQ
     ,COUNT(1) CNT
    FROM BWTA_TASK T
    JOIN LRL ON LRL.HEAP_SEQ=T.HEAP_SEQ
    LEFT JOIN BWTA_LOG_TASK LT ON LT.TASK_SEQ=T.SEQ AND LT.ROUND_SEQ=LRL.ROUND_SEQ
    GROUP BY T.HEAP_SEQ
     ,NVL(LT.STATUS,'WAIT')
     ,LRL.ROUND_SEQ
   )
,L1 as(
    SELECT
       H.SEQ   AS HEAP_SEQ
      ,H.ID    AS HEAP_ID
      ,H.NOTE  AS HEAP_NOTE
      ,LR.START_DT
      ,LR.END_DT
      ,S.STATUS
      ,S.CNT
    FROM BWTA_HEAP H
    JOIN STAT S ON S.HEAP_SEQ=H.SEQ
    JOIN BWTA_LOG_ROUND LR on LR.SEQ=S.ROUND_SEQ
)
Select * from L1
PIVOT
(SUM(CNT) FOR STATUS IN (
  'DONE' AS Done_cnt
 ,'WAIT' AS Wait_cnt
 ,'ACTIVE' AS Active_cnt
 ,'SUSPEND' AS Suspend_cnt
 ,'ERROR' AS Error_cnt
 ,'SKIP' AS Skip_cnt
))
```

```
  HEAP_SEQ HEAP_ID    HEAP_NOTE                 START_DT            END_DT                DONE_CNT   WAIT_CNT ACTIVE_CNT SUSPEND_CNT  ERROR_CNT   SKIP_CNT
---------- ---------- ------------------------- ------------------- ------------------- ---------- ---------- ---------- ----------- ---------- ----------
        -1 DEFAULT    Default batch process     29.06.13 17:33:06                                           4          3
```

List of remaining tasks

```
SELECT
  TASK_ID
, EFFECTIVE_DATE
, STATUS
, to_char(START_DT,'DD.MM.YYYY HH24:MI:SS') AS START_DT
, to_char(END_DT,'DD.MM.YYYY HH24:MI:SS') AS START_DT
, END_DT
, MINUTES
, AVG_MINUTES
, PREDECESSOR_TASKS
, BLOCKING_TASKS
FROM BWTA_V_LOG_TASK_WIDE
where HEAP_SEQ=-1
AND NVL(STATUS,'#') not in ('DONE','SKIP')
ORDER BY
  CASE STATUS WHEN 'ERROR' THEN 1
              WHEN 'SUSPEND' THEN 2
              WHEN 'ACTIVE' THEN 3
              ELSE 4
  END
, TASK_ID
```

Active operational tasks:

Wake up sleeping threads
(after change of metadata)

```
Begin
  BWTA_OPER.WakeupThread;
  commit;
End;
/
```

Stop workflow
Stops run new processes, keeps current processes to be finished

```
Begin
  BWTA_OPER.stop;
End;
/
```

Restart of workflow after stop

```
Begin
  BWTA_OPER.ReleaseThreads;
End;
/
```

Remove inconsistence between system and metadata
check orphans

```
Begin
  BWTA_OPER.checkOrphans;
End;
/
```

Add threads into processing
number of additional threads should be specified

```
Begin
  BWTA_OPER.addThreads(<number of threads>);
End;
/
```

Reduce number of threads
Deletes required number of sleeping threads. If there is not enough sleeping threads next will be marked with the STOP command.

```
Begin
  BWTA_OPER.remThreads(<number of threads>);
End;
/
```

Success story of mentioned solution:
Despite it seems sofisticated parallelizing solutions are domain of stable ETL processes more than one time migration processes, we used it for one time migration of historical data from 4 country-based sites. Challenge of the migration was, the data we migrated slightly differed in time and slightly differed between particular sites.
Migration processes had been developed and tested on samples of data so run-time problems was expected and there was just limited frame of time to migrate all the data history.
Implementing mentioned SIMPLETASK solution in heavy parallel mode the effect of failover resulted in following result:

- ***Two developers permanently solved problems of data inconsistencies, what would be show stopper in most other situations***
- ***Failover effect of Simple task caused not a single minute has been lost on the critical path of the migration***

Despite I consider it almost incredible, if you decide to remove the solution from your system, I add a script to drop all the created database object belonging to SIMPLETASK solution above.

Script: Drop_scheduler_solution.sql

```
Drop package BWTA_METADATA;
Drop package BWTA_OPER;
Drop package BWTA_THREAD;
Drop table BWTA_HEAP cascade constraints;
Drop table BWTA_TASK cascade constraints;
Drop table BWTA_TASK_REL cascade constraints;
Drop table BWTA_RES cascade constraints;
Drop table BWTA_TASK_RES cascade constraints;
Drop table BWTA_ERR cascade constraints;
Drop table BWTA_LOG_ROUND cascade constraints;
```

...more

Ludek Bob Jankovsky