2015

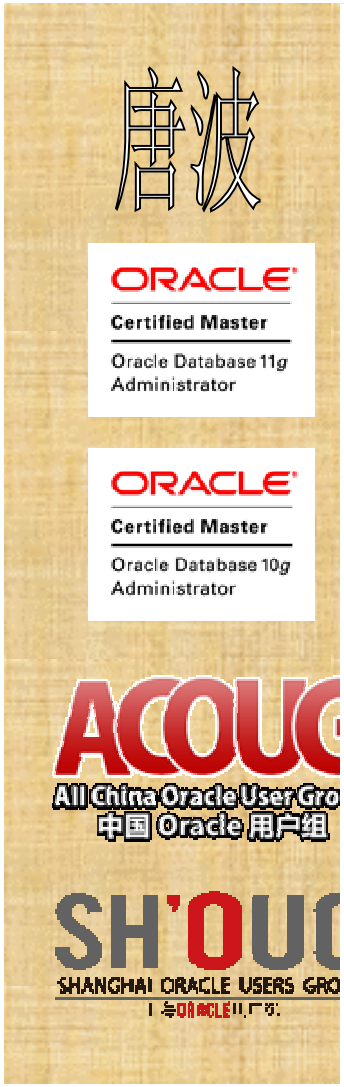
Oracle技术嘉年华

Oracle Technology Carnival 2015

稳健●高效●云端 - 数据技术最佳实践

讲师：**Botang**唐波

主题：**Oracle Scheduler**作业链规则编程在工控响应系统中的应用



福建省第一批Oracle ERP实施顾问。精通Oracle财务系统的11i Financial Functional Fundation、9i Oracle Discoverer、Oracle 11i System Administrator Fundamentals产品的实施配置。2004年4月到2006年12月在北京担任中国科学院ARP项目组数据仓库架构师，数据中心核心团队成员。参与完成该项目中的数据仓库设计、数据仓库建模、维度和立方开发、ETL stream过程，建立完善的数据仓库前台展示系统，利用Oracle AS应用服务器结合Discoverer进行前台展示。最终被中国科学院办公厅和中国科学院ARP项目管理办公室评为最佳技术实施顾问。

现任职于中国科学院某单位。被聘为高级工程师。目前一方面继续负责Oracle ERP大型分布式系统及其附属RedFlag Linux的运行维护；另一方面负责一套30万亿次运行RedHat的科学计算超级集群的运行维护。8小时外兼职Redhat RHCI和Oracle WDP讲师，福建省RHCE/OCP/OCM培训学员数量最多的讲师。

Oracle Scheduler作业链规则编程在工控响应系统中的应用

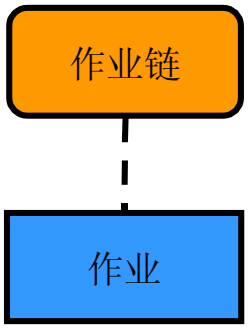
简介：Oracle Scheduler CHAIN，也就是作业链，是将一串需要完成的作业逻辑编程连在一起，根据每一个步骤完成的不同返回结果来确定下面的哪一个动作需要被完成。链的第一步一般是监听接收ADT的基于事件的schedule，其他步骤可能是基于事件的schedule或program。整个chain，可以设计多个exit，但每次特定执行，只会根据当次规则满足情况从一个出口退出。CHAIN的主要步骤包括定义程序(做什么)、定义CHAIN步骤(总步骤)，以及CHAIN的规则(如何做)。

创建作业链

1. 创建作业链对象。
2. 定义作业链步骤。
3. 编写作业链规则。
4. 启动作业链：

   使能作业链。

   创建作业指向作业链。



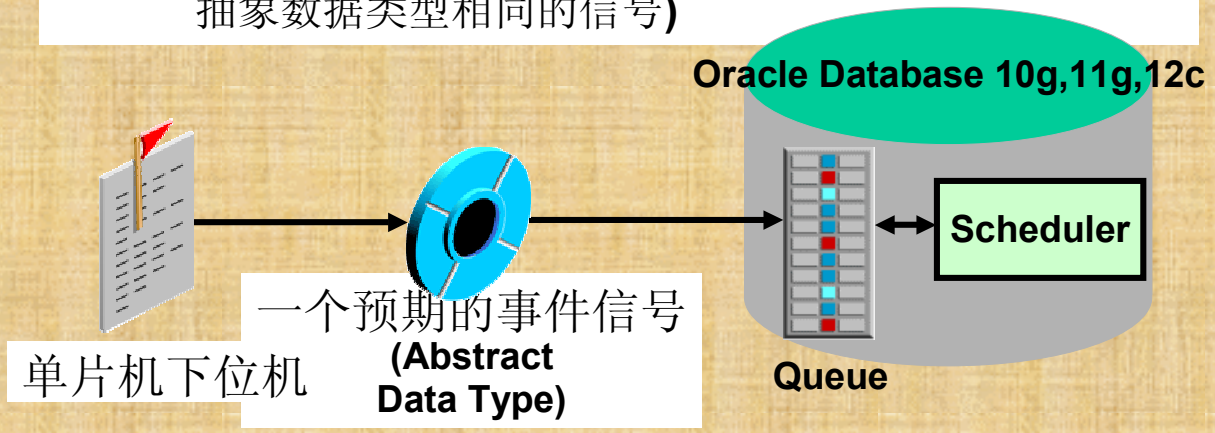
链的第一步一般是监听接收ADT的基于事件的schedule，其他步骤可能是基于事件的schedule或program。

创建基于事件的Schedule

创建基于事件的**schedule:**

高级队列定义（单片机下位机捕获工控信号，由上位机使用**ADT**格式 **enqueues**该高级队列来启动作业链第一步）

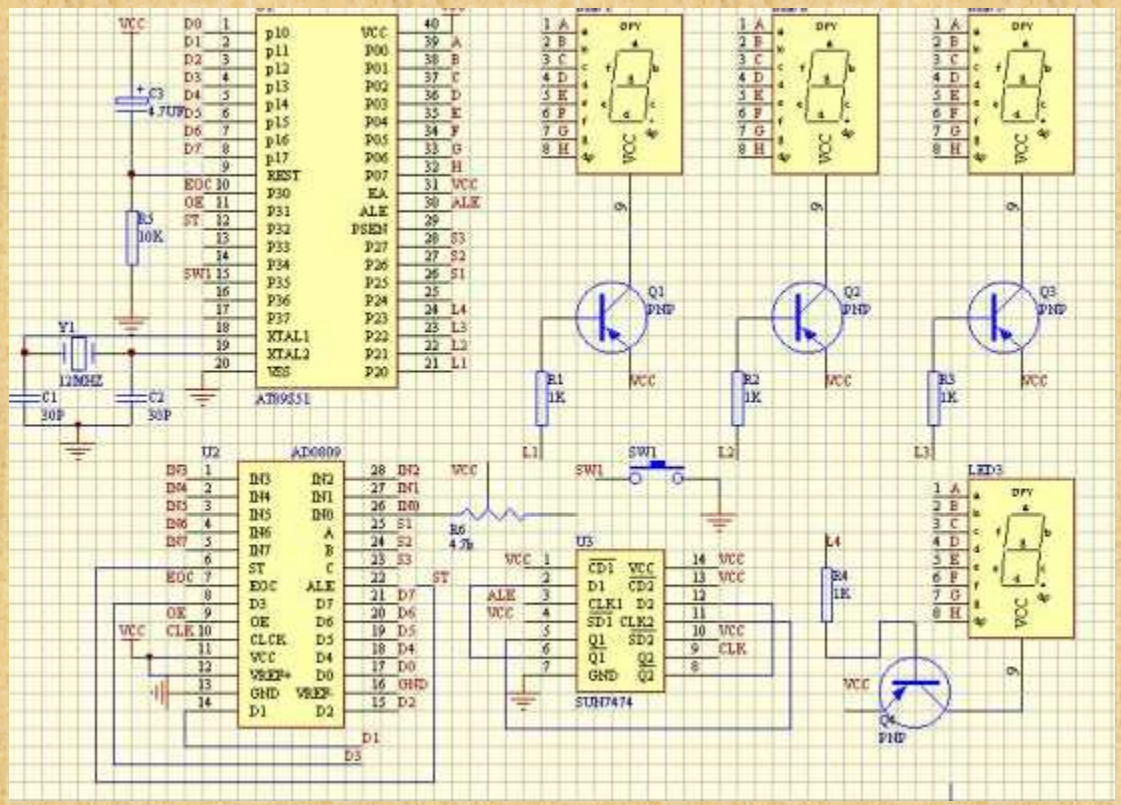
一个预期的事件信号 （与**Oracle Streams**高级队列中的抽象数据类型相同的信号**)**



上位机是指：人可以直接发出操控命令的计算机，一般是PC，屏幕上显示各种信号变化（液压，水位，温度等）。

下位机是直接控制设备获取设备状况的的计算机，一般是PLC/单片机之类的。上位机发出的命令首先给下位机，下位机再根据此命令解释成相应时序信号直接控制相应设备。下位机不时读取设备状态数据（一般模拟量），转化成数字信号反馈给上位机。简言之如此，真实情况千差万别不离其宗。上下位机都需要编程，都有专门的开发系统。

单片机下位机电路图


VCC
AT89S51
12MHZ
C1
30P
C2
30P
C3
4.7UF
R3
10K
SW1
U2
AD0809
R6
4.7k
U3
SUH7474
Q1
PNP
Q2
PNP
Q3
PNP
Q4
PNP
R1
1K
R2
1K
R3
1K
R4
1K
L1
L2
L3
L4
LED3
DPY

上位机用Java进行Enqueuing and Dequeuing：

```
public static void AQObjectPayloadTest(AQSession aq_sess)
      throws AQException, SQLException, ClassNotFoundException
 {
   Connection          db_conn   = null;
   AQQueue             queue     = null;
   AQMessage           message   = null;
   AQObjectPayload     payload   = null;
   AQEnqueueOption     eq_option = null;
   AQDequeueOption     dq_option = null;
   PERSON          pers = null;
   PERSON          pers2= null;
   ADDRESS          addr = null;

   db_conn = ((AQOracleSession)aq_sess).getDBConnection();


   queue = aq_sess.getQueue("aquser", "test_queue2");


   /* Enable enqueue/dequeue on this queue */
   queue.start();

   /* Enqueue a message in test_queue2 */
   message = queue.createMessage();

   pers = new PERSON();
   pers.setName("John");
   addr = new ADDRESS();
   addr.setStreet("500 Easy Street");
   addr.setCity("San Francisco");
   pers.setHome(addr);

   payload = message.getObjectPayload();
   payload.setPayloadData(pers);
   eq_option = new AQEnqueueOption();

   /* Enqueue a message into test_queue2 */
   queue.enqueue(eq_option, message);
```

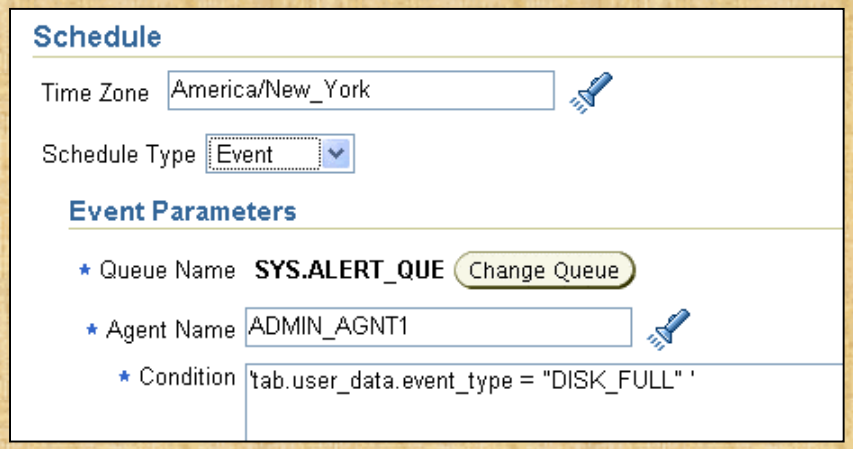
上位机用PL/SQL进行Enqueuing and Dequeuing：

```
BEGIN
  DBMS_SCHEDULER.create_job (
    job_name        => 'job4',
    job_type        => 'PLSQL_BLOCK',
    job_action      => 'BEGIN
                        INSERT INTO scheduler_test (id, created_date)
                        VALUES (scheduler_test_seq.NEXTVAL, SYSDATE);
                        COMMIT;
                       END;',
    start_date      => SYSTIMESTAMP,
    event_condition => 'tab.user_data.event_name = ''give_me_a_prod''',
    queue_spec      => 'event_queue',
    enabled         => TRUE);
END;

DECLARE
 l_enqueue_options     DBMS_AQ.enqueue_options_t;
 l_message_properties  DBMS_AQ.message_properties_t;
 l_message_handle      RAW(16);
 l_queue_msg           sys.t_event_queue_payload;
BEGIN
 l_queue_msg := sys.t_event_queue_payload('give_me_a_prod');

 DBMS_AQ.enqueue(queue_name          => 'sys.event_queue',
                 enqueue_options     => l_enqueue_options,
```

使用Enterprise Manager 创建基于事件的Schedules



基于事件的 Schedule

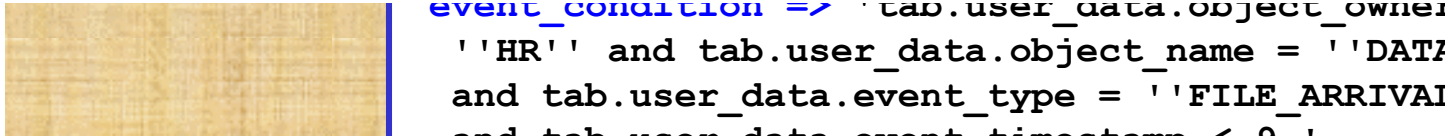
--SYS:

队列

```
select  * from dba_queues   q  where
q.QUEUE_TYPE='NORMAL_QUEUE' ;
select  * from dba_queues q where
q.QUEUE_TYPE='EXCEPTION_QUEUE';
select  * from dba_queues q where q.QUEUE_TYPE   not in
('NORMAL_QUEUE','EXCEPTION_QUEUE');
```

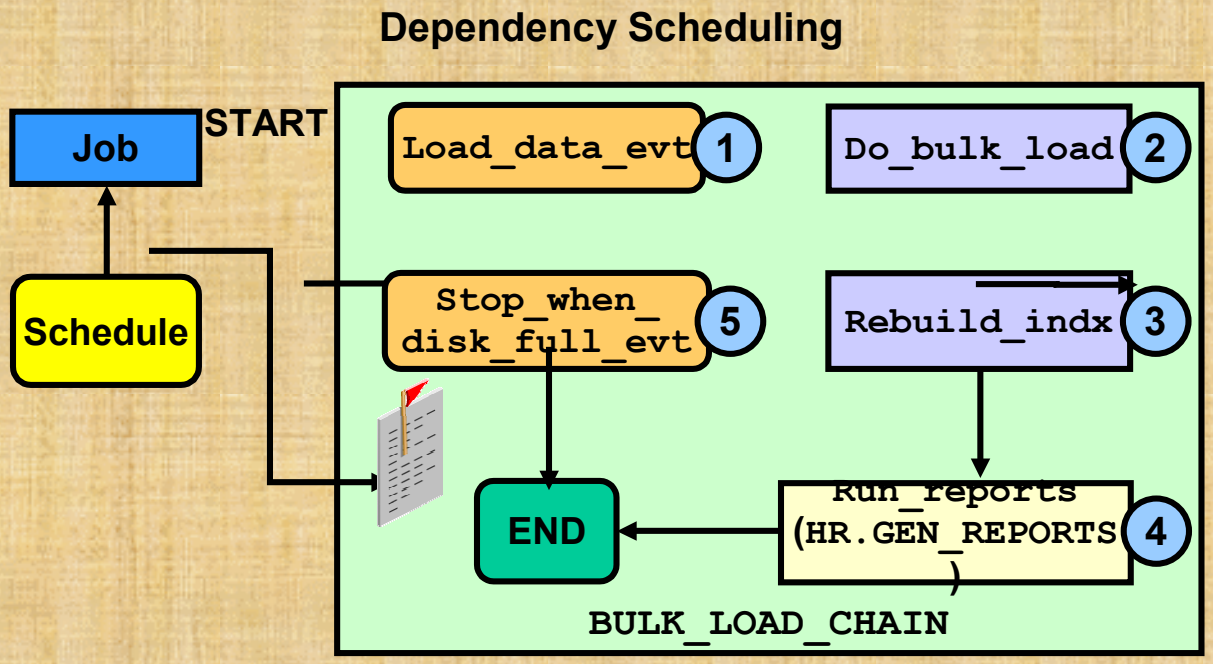
类型

```
CREATE OR REPLACE TYPE t_event_queue_payload AS OBJEC
  event_name  VARCHAR2(30)
);
grant execute on t_event_queue_payload to hr;
```

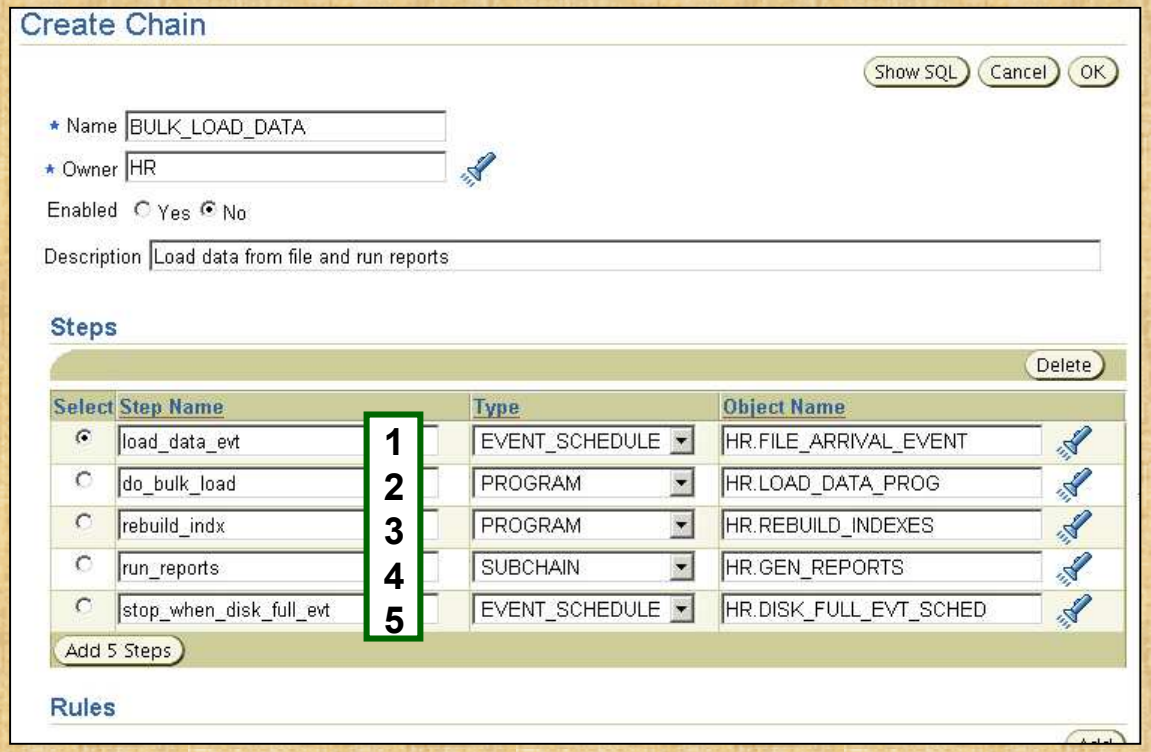


Oracle Scheduler CHAIN，也就是作业链，是将一串需要完成的作业逻辑编程连在一起，根据每一个步骤完成的不同返回结果来确定下面的哪一个动作需要被完成。

作业链的例子


Dependency Scheduling
Job
START
Schedule
Load_data_evt
1
Do_bulk_load
2
Stop_when_
disk_full_evt
5
Rebuild_indx
3
END
Run_reports
(HR.GEN_REPORTS
)
4
BULK_LOAD_CHAIN

1. 创建链对象


Create Chain
Show SQL
Cancel
OK
* Name BULK_LOAD_DATA
* Owner HR
Enabled Yes No
Description Load data from file and run reports
Steps
Delete
Select
Step Name
Type
Object Name
load_data_evt
EVENT_SCHEDULE
HR.FILE_ARRIVAL_EVENT
1
do_bulk_load
PROGRAM
HR.LOAD_DATA_PROG
2
rebuild_indx
PROGRAM
HR.REBUILD_INDEXES
3
run_reports
SUBCHAIN
HR.GEN_REPORTS
4
stop_when_disk_full_evt
EVENT_SCHEDULE
HR.DISK_FULL_EVT_SCHED
5
Add 5 Steps
Rules

2. 定义链步骤

```
DBMS_SCHEDULER.DEFINE_CHAIN_EVENT_STEP (
   chain_name  => 'bulk_load_chain',                    1
   step_name  => 'load_data_evt',
   event_condition => 'tab.user_data.object_owner
=
     ''HR'' and tab.user_data.object_name =
     ''DATA.TXT'' and tab.user_data.event_type =
     ''FILE_ARRIVAL'' ',
   queue_spec => 'HR.LOAD_JOB_EVENT_Q');
DBMS_SCHEDULER.DEFINE_CHAIN_STEP (                      2
   chain_name  => 'bulk_load_chain',
   step_name  => 'do_bulk_load',
   program_name => 'hr.load_data_prog);
 DBMS_SCHEDULER.DEFINE_CHAIN_STEP (                     3
   chain_name  => 'bulk_load_chain',
   step_name  => 'rebuild_indx',
   program_name => 'hr.rebuild_indexes');
```

整个chain，可以设计多个exit，但每次特定执行，只会根据当次规则满足情况从一个出口退出。CHAIN的主要步骤包括定义程序(做什么)、定义CHAIN步骤(总步骤)，以及CHAIN的规则(如何做)。

规则制定：
每一条规则都要有“condition”和“action”。
如果condition为TRUE, the action执行。

3. 定义链规则

Rules

Add | Edit | Delete

Select	Name	Condition	Action	Description
◉	System_Assigned_1	1=1	START load_data_evt, stop_when_disk_full_evt	Start the chain, by waiting for the file to arrive
○	System_Assigned_2	:load_data_evt.COMPLETED="TRUE"	START do_bulk_load	
○	System_Assigned_3	:do_bulk_load.STATE="SUCCEEDED"	START rebuild_indx	
○	System_Assigned_4	:rebuild_idnx.COMPLETED="TRUE"	START recalc_stats	recalc stats after indexes rebuilt
○	System_Assigned_5	:recalc_stats.STATE="SUCCEEDED"	START run_reports	Start generating reports ONLY if stats recalculated
○	System_Assigned_6	:recalc_stats.STATE != "SUCCEEDED"	END 10	Exit chain (10) if stats recalc fails
○	System_Assigned_7	:stop_when_disk_full.COMPLETED="TRUE"	END 99	Exit chain (99) if disk full
○	System_Assigned_8	:run_reports.STATE="SUCCEEDED"	END	Exit normally when successful

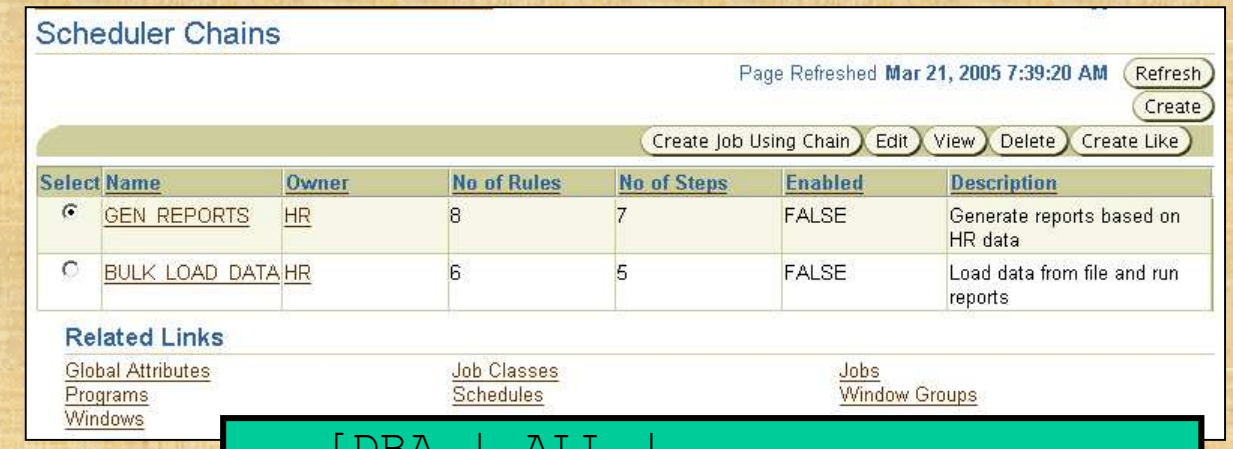
Evaluation Interval (minutes)

4. 启动作业链

```
BEGIN
  DBMS_SCHEDULER.ENABLE ('bulk_load_chain');
END;
/
```

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
    job_name         => 'bulk_load_chain_job',
    job_type         => 'CHAIN',
    job_action       => 'bulk_load_chain',
    repeat_interval => 'freq=daily;byhour=7;
                        byminute=5;bysecond=0',
    enabled          => TRUE);
END;
/
```

监控作业链




```
[DBA | ALL |
  USER]_SCHEDULER_CHAINS

[DBA | ALL |
  USER]_SCHEDULER_CHAIN_RULES

[DBA | ALL |
  USER]_SCHEDULER_CHAIN_STEPS
```


THANKS