

Politecnico di Torino

Database Management Systems

September 21st 2011

Trigger - draft solution

1. (7 Points) The following relations are given (primary keys are underlined, optional attributes are denoted with *):

```
EMPLOYEE(ECode, EName, Qualification)
HOURLY_PAY_EMPLOYEE(TypeOfActivity, Qualification, HourlyPay)
TypeOfActivity={Ordinary, Overtime}
DAILY_SUMMARY(ECode, Date, HoursOfWork)
PAY_PACKET(ECode, Month, TotalAmount)
PAY_PACKET_REQUEST(RCode, ECode, Month)
```

A company wants to manage the calculation of the monthly pay-packets for the employees.

For each employee, the monthly pay-packet is computed based on her hourly pay and the number of hours she has worked in the month. The hourly pay depends on the qualification of the employee and the type of activity (ordinary or overtime) she carried out. Each month, the hours of work of the employee are paid as 'ordinary' activity up to 160 hours. The hours of work of the employee exceeding 160 hours are paid as 'overtime' activity.

The EMPLOYEE table contains the qualification for each employee. The HOURLY_PAY_EMPLOYEE table contains the hourly pay based on the type of activity (ordinary or overtime, attribute TypeOfActivity) and the qualification of the employees. The DAILY_SUMMARY table contains, for each employee, the hours of work in each date.

Write the triggers managing the following activities .

(1) *Calculation of a new pay-packet.* The calculation of a new pay-packet is requested (a new record is inserted in the PAY_PACKET_REQUEST table), for the hours of work of a given employee in a specific month. To calculate the pay-packet, all the hours worked by the employee in the month have to be considered, including both ordinary and overtime activity. A new record including the information on the new pay-packet must be inserted in the PAY_PACKET table. The month can be extracted from the Date attribute in the DAILY_SUMMARY table by using the TOMONTH function.

```
EVENT: insert on pay_packet_request
```

```
EXECUTION GRANULARITY: row level (one request at a time)
```

```
EXECUTION MODE: after (business rule)
```

```
CONDITION: NO
```

```
ACTIONS:
```

- ```
(1) compute the total hours of work for the employee in the requested month
(check if there is at least one tuple for the employee in the requested month)
 (2) read the employee qualification
 (3) read the hourly pay for ordinary activity
 (4) if the total hours of work done by the employee is lower than or equal to 160 then
 (5) compute the pay packet for the employee
 else
 (7) read the hourly pay for the overtime activity
 (8) compute the pay packet for the employee
 (9) insert the pay packet in the pay_packet table
```

```

create or replace trigger pay_packet_calculation
after insert on pay_packet_request
for each row
declare
 Q number;
 totalhours number;
 ordinary_cost number;
 overtime_cost number;
 totalcost number;
begin

--compute the total hours of work for the employee in the requested month
select sum(HoursOfWork) into totalhours
from DAILY_SUMMARY
where ECode = :NEW.ECode and tomonth(Date)=:NEW.Month;

if (totalhours IS NOT NULL) then
---there is at least one tuple for the employee in the requested month

--read the employee qualification
select Qualification into Q
from EMPLOYEE
where ECode = :NEW.ECode;

--read the hourly pay for ordinary activity
select HourlyPay into ordinary_cost
from HOURLY_PAY_EMPLOYEE
where Qualification = Q and TypeOfActivity = 'Ordinary';

if (totalhours <= 160) then
 totalcost := totalhours * ordinary_cost;
else
--read the hourly pay for the overtime activity
select HourlyPay into overtime_cost
from HOURLY_PAY_EMPLOYEE
where Qualification = Q and TypeOfActivity = 'Overtime';

 totalcost := 160*ordinary_cost + (totalhours-160)*overtime_cost;
end if;

---insert the new pay packet into the PAY_PACKET table
insert into PAY_PACKET(ECode, Month, TotalAmount)
values (:NEW.ECode, :NEW.Month, totalcost);

end if;
end;

```

(2) *Integrity constraint on the working day.* The integrity constraint requires that any new record inserted in the `DAILY_SUMMARY` table contains the number of hours worked by the employee in the *current date*. Insertion of a new record with the hours of work related to dates preceding or following the current date is not possible. If a new record concerning different dates than the current date is inserted, the insert operation must not be executed. The current date can be extracted from the system date (function `sysdate`) by using the `TODAY` function. Write the trigger enforcing this integrity constraint.

```
EVENT: insert on DAILY_SUMMARY
 update of Date on DAILY_SUMMARY
EXECUTION GRANULARITY: row level (one tuple at a time is checked)
EXECUTION MODE: after (also before is correct)
CONDITION: date <> today(sysdate)
ACTION: rollback

create or replace trigger check_working_date
after insert or update of Date on DAILY_SUMMARY
for each row
when (NEW.Date <> today(sysdate))
begin
 raise_application_error(-20500,'Only hours for the current date can be inserted');
end;
```