P26

/* Update cursor addition */
/* This works */

-- We would like to add all values in a row and then
-- save the sum in a new column. We already created
-- the new column with an "alter" before.
-- But here is how you do the adding.

declare
cursor janet is
  select * from x for update;

begin
  for jane in janet
    loop
      update x
      set fourth = jane.num + jane.sqr + jane.cube
      where current of janet;
    end loop;
  end;
/

P28

/*
 Triggers

 A trigger is a little program that is
 attached to a table.

 Whenever somebody updates/deletes/inserts
 into that table, the trigger program will
 be executed.

 That program can do whatever you want.  
 Like... remember when the update happened.

 */

P29
-- A trigger example.

/* Whenever somebody deletes something from my table,
I just print a message. */

create or replace trigger jennifer
before delete on x
for each row
begin
    dbms_output.put_line('Warning, somebody is trying to delete');
end;
/
/

P29b

-- This does NOT work!!!!
-- You cannot declare a trigger for truncate!

create or replace trigger jennifer
before truncate on x <<<< NO NO NO
for each row
begin
    dbms_output.put_line('Warning, somebody is trying to delete');
end;
/

If you do

truncate table x

the delete trigger will NOT be activated!!!

P29c
-- This does NOT work!!!!
-- No trigger for SELECT

create or replace trigger jennifer
before select on x
for each row
begin
dbms_output.put_line('Warning, somebody is trying to see your secrets.');
end;
/

P29d

-- So how do you stop deleting effectively?

CREATE OR REPLACE TRIGGER stop_delete
BEFORE DELETE ON X
BEGIN
  RAISE_APPLICATION_ERROR(-20001, 'Delete operation not allowed!');
END;

P29e
-- Recursive calls from trigger:

create or replace trigger trident
after insert on x

begin
    dbms_output.put_line("Warning!");
    insert into x values(5,25,125);
end;
/

 insert into x values(6, 36, 216)

ERROR at line 1:
ORA-00036: maximum number of recursive SQL levels (50) exceeded
ORA-06512: at "GELLER.TRIDENT", line 3
ORA-04088: error during execution of trigger 'GELLER.TRIDENT'
ORA-06512: at "GELLER.TRIDENT", line 3
ORA-04088: error during execution of trigger 'GELLER.TRIDENT'
ORA-06512: at "GELLER.TRIDENT", line 3
ORA-04088: error during execution of trigger 'GELLER.TRIDENT'
ORA-06512: at "GELLER.TRIDENT", line 3

P29f

This has nothing to do with triggers.
But it is also useful. Formatting output.

begin
    dbms_output.put_line('Hello' || chr(13) || chr(13) || 'World'); -- New lines
    dbms_output.put_line('Hello' || chr(9) || chr(9) || 'World'); -- Tabs
end;

Even nicer is this:

begin
    dbms_output.put_line(rpad('Hello', 20) || rpad('World', 20));
    dbms_output.put_line(rpad('Hello you', 20) || rpad('Beautiful World', 20));
end;
/

P30
-- This is an improved trigger.
-- Now I am assuming an update.
-- I am storing the value before update and after update in a
-- table called track.  Loaded from track.sql.
-- For each row... means for each row that is affected by my
-- trigger statement.

create or replace trigger jennifer
before update on x
for each row
begin
  dbms_output.put_line('Here I am before changing');
  dbms_output.put_line(:old.num || ' ' || :old.sqr || ' ' || :old.cube);
  dbms_output.put_line('Here I am after changing');
  dbms_output.put_line(:new.num || ' ' || :new.sqr || ' ' || :new.cube);
  insert into track values(:old.num, :old.sqr, :old.cube,
                          :new.num, :new.sqr, :new.cube);
end;
/

Call this with an update statement such as:

update x
set num = 2 * num;

P31

-- This is a statement trigger.
-- As opposed to a row trigger.

create or replace trigger trident
before update on x
-- for each row  -- it's done only ONCE for the whole table
begin
  dbms_output.put_line('');
  dbms_output.put_line('**** Several lines are being changed');
end;
/
/*
Call this with an update statement such as:

update x
set num = 2 * num;

Now, just for fun:

update x
set num = num + 0;

Will this trigger, or not?

P32
-- Now we specialize the trigger to a column.
-- Only that column starts the trigger.

-- :old and :new are just called prefixes.

create or replace trigger trident
  before update of num -- one column only!
on x  -- still table x
  for each row
  begin
    dbms_output.put_line('Here I am before changing');
    dbms_output.put_line(:old.num || ' ' || :old.sqr || ' ' || :old.cube);
    dbms_output.put_line('Here I am after changing');
    dbms_output.put_line(:new.num || ' ' || :new.sqr || ' ' || :new.cube);
  end;
/

NOTE: Cannot do columns with INSERT trigger.

P32b

/* extended version with two columns and
  two possible events: update or delete
*/

create or replace trigger trident
  before update or delete of num, sqr
  on x  -- still table x
  for each row
  begin
    dbms_output.put_line('Here I am before changing');
    dbms_output.put_line(:old.num || ' ' || :old.sqr || ' ' || :old.cube);
    dbms_output.put_line('Here I am after changing');
    dbms_output.put_line(:new.num || ' ' || :new.sqr || ' ' || :new.cube);
  end;
/
-- Now we add an additional trigger condition for selecting rows
-- according to their content.
-- Secondly, the trigger needs to be loaded after the table, or
-- things don't work.
-- Thirdly, it took me a long time to get the trigger condition
-- to work. I got it to work by doing what I thought was the
-- most unlogical thing possible.... Remove the : from the :old.
-- And look... it works...

create or replace trigger trident
before update
on x
for each row
when (old.num > 3) -- Trigger Condition
begin
dbms_output.put_line('Here I am before changing');
dbms_output.put_line(old.num || ' ' || old.sqr || ' ' || old.cube);
dbms_output.put_line('Here I am after changing');
dbms_output.put_line(new.num || ' ' || new.sqr || ' ' || new.cube);
end;
/

P33b

-- Does not work!!!!!

create or replace trigger trident
before update
on x
for each row
when (num > 3) -- Trigger Condition
begin
dbms_output.put_line('Here I am before changing');
dbms_output.put_line(old.num || ' ' || old.sqr || ' ' || old.cube);
dbms_output.put_line('Here I am after changing');
dbms_output.put_line(new.num || ' ' || new.sqr || ' ' || new.cube);
end;
/

P33c
I was asked how to see your own triggers. Try this:

```sql
select * from user_objects
```

Even better:

```sql
select * from user_triggers
```

How many triggers can you put on one table?

I create three identical triggers (trident1, trident2, trident3). They are all:

```sql
before update on x
-- for each row
```

I just changed the printing message. It printed the last (newest) trigger first!!!

---

**P33e**

**Schema Level Triggers:**

```sql
create or replace trigger schema1
before create on geller.schema
begin
    dbms_output.put_line('creating...');
end;
/
```

Can create such triggers for:

LOGON LOGOFF CREATE DROP ALTER

But I did not get logon/logoff to work under Aqua.

You can have local variables in a trigger:

```sql
create or replace trigger trident10
before insert on x
for each row
declare
    num number;
begin
    num :=1;
    dbms_output.put_line(num);
end;
/
```