Universidad Carlos III

Files and databases 2022-23

Course 2022-23

LabReport P2

Lecturers:	**Jorge Vico Pedrero and Francisco Javier Calle**		
Group:	**88**	**Lab User**	**FSDB228**
Student:	**József Iván Gafo**	**NIA:**	**100456709**
Student:	**Marcos González Vallejo**	**NIA:**	**100472206**

Index

1 Introduction

This lab work consists of the implementation of several operative elements. Specifically those operative elements are: 2 queries, then a package containing several variables, procedures and functions, then 3 views in which the implementation of the trigger is necessary for the last one (read and write) and finally 3 triggers.

For the first query we had to provide the performer, the percentage of recorded tracks that a member of the band has writer and the percentage of performances of songs that were written by a member of the band.

For the second query, we had to take the 10 best percentages of performers who performed a song that was recorded previously by them. Also, we had to include the average age of the songs performed.

For both queries, we made use of natural join renaming those attributes that we wanted to join from both sides of the join so that they match in the join.

For the second exercise we had to create a package containing a variable, assigned by using a procedure, 2 procedures, one for inserting and the other for deleting and finally a report showing information about the current performer. We just created a procedure for each one and applied several exceptions of common errors.

Fo the third exercise, we had to do 3 views, all of them related to the current performer. First a view showing the albums and their duration of the current performer. Secondly a view showing 1 row per month and year where we show how many concerts were done that month and year, the average number of performances, total attendees and average duration. Finally, we had to do a view showing those clients that went to more than 1 concert of the current performer and then delete and insert using an additional banned fans table that is populated whenever a fan is excluded from the view.

For the fourth exercise, we had to do 3 triggers. For the first one, which consists on updating the duration of a concert whenever a new performance is included, we decided to include the trick of the mutating table just in case it appeared. However, the code of the trigger was just to get the inserted duration of the performance and add it to the duration of the concert with the same performer and date. For the second one, we had to reject insertions in attendees whose purchase date - birth date was lower than 18. We just wrote an if conditional to check that part. Finally, in the third trigger we had to reject the insertion of a song whose writer and cowriter already are in the songs table but in the opposite column. For this trigger we just checked if in songs, writer = :NEW.cowriter and cowriter = :NEW.writer.

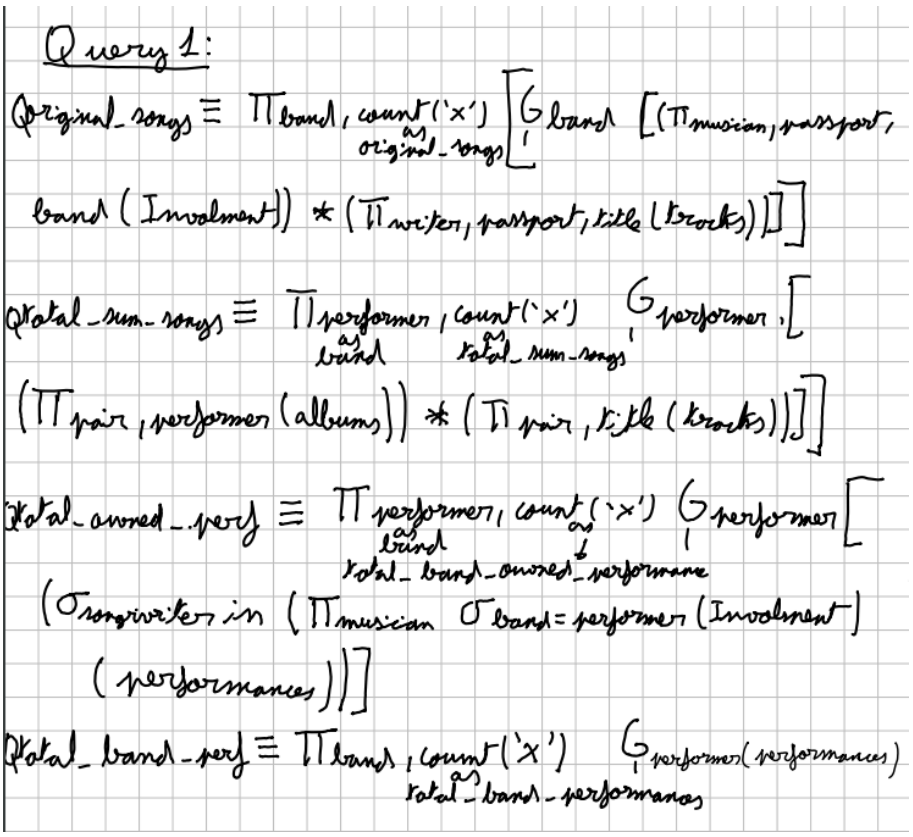
What we considered as the goal of the lab work is to, firstly learn how to implement several elements such as triggers, queries, views and functions and procedures in a package, but mostly, the goal of the project was to get used with queries, because they are used everywhere.

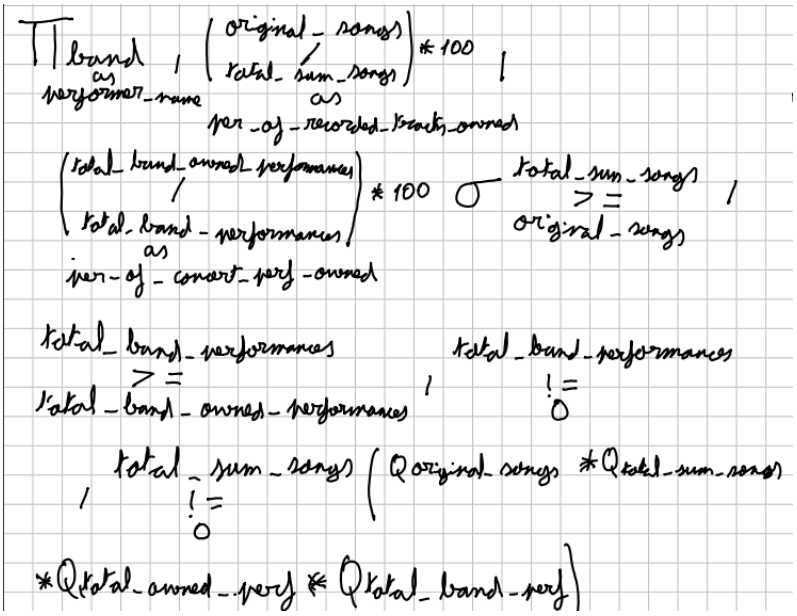
The objective was to get loose with queries because the rest of the implementation was just programming, something that we already know. These queries are used everywhere, procedures, functions, views, triggers… That is why we found very important to have a great knowledge of querying so that the implementation of the rest of the elements is easier. Finally, this document will consist, firstly on queries (with design, implementation and tests), packages (with design, implementation and tests), views (with design implementation and tests) and finally triggers (with design, implementation and tests).

2 Queries

- **Relational Algebra:**

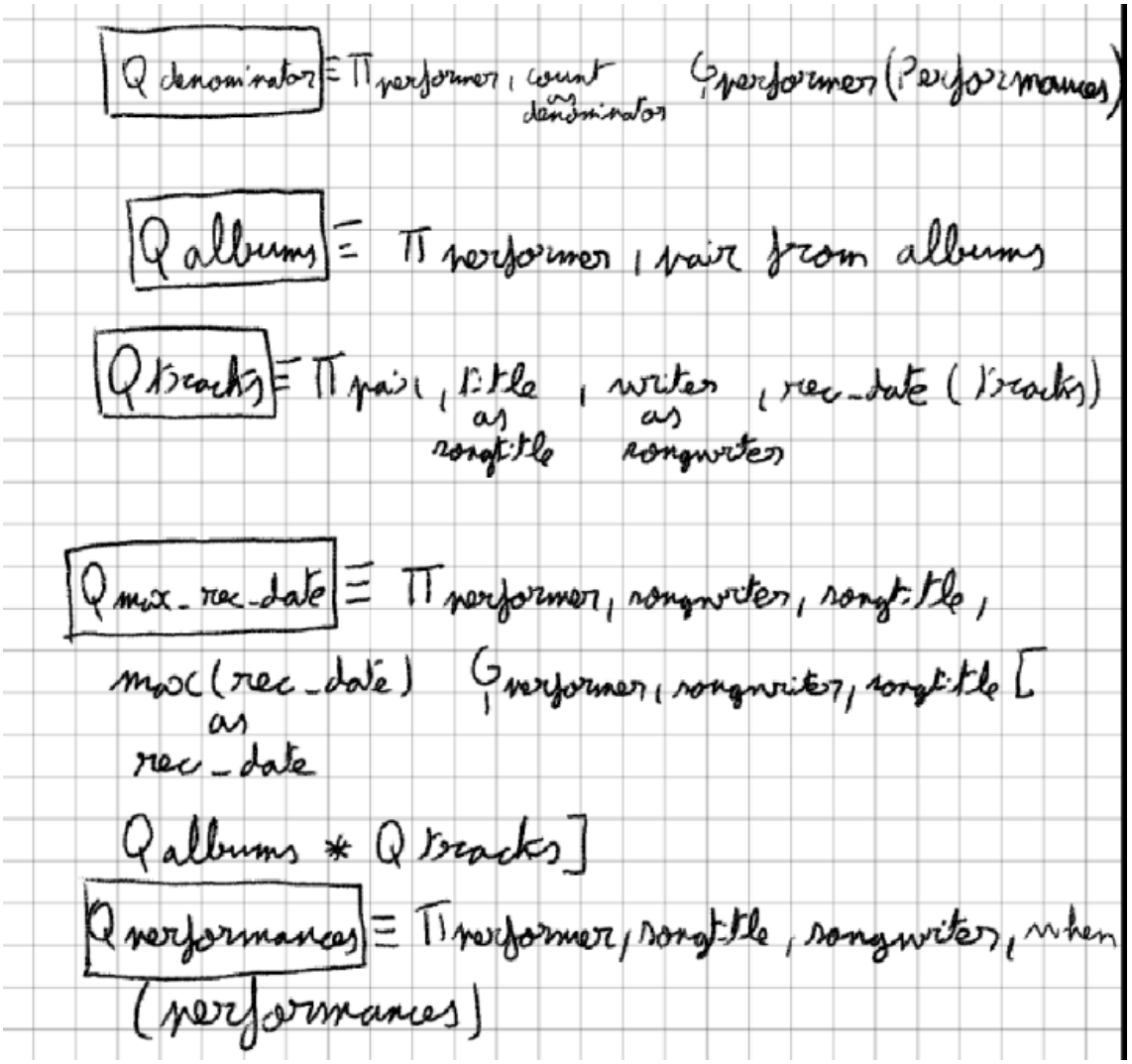
Query1:

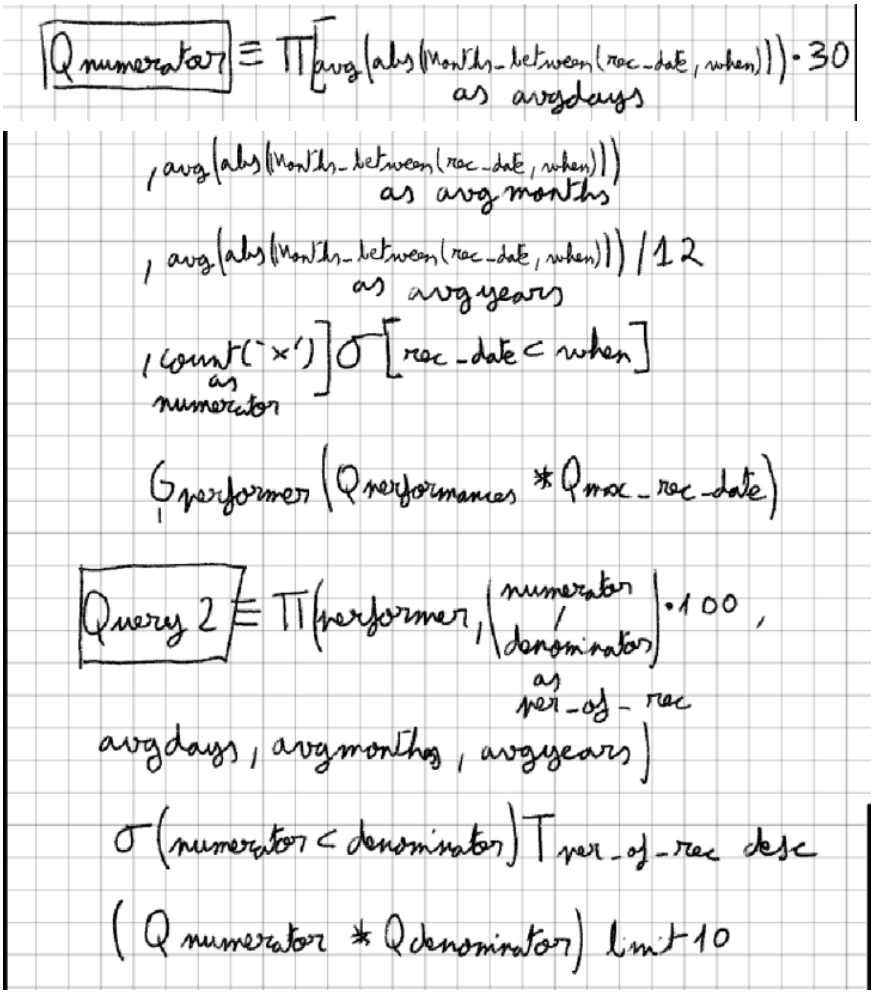




Query2:

First of all, in this query 2, we have decided to take the top 10 best performances under 100 percent, because we decided that it may be more convenient for the tests.





- **SQL:**

Query1:

```
select band as performer_name,
(original_songs/total_sum_songs)*100 as per_of_recorded_tracks_owned ,
(total_band_owned_performances/total_band_performances)*100 as  per_of_concert_perf_owned from
((select band,count('x') original_songs from (select musician passport,band from involvement) NATURAL JOIN (select writer
passport, title from tracks) group by band )
natural join
(select performer as band,count('x') total_sum_songs from (select pair, performer from albums) NATURAL JOIN (select pair,
title from tracks) group by performer)
natural join
(select performer band,count('x') total_band_owned_performances from performances where songwriter in (select musician from
involvement where band = performer)
group by performer)
natural join
```

```
(select performer as band,count('x') total_band_performances from performances group by performer))
where total_sum_songs >= original_songs and total_band_performances>=total_band_owned_performances
and total_band_performances!=0 and total_sum_songs!=0;
```

Query2:

```
select performer, (numerator / denominator)*100 per_of_rec,avgdays,avgmonths,avgyears from (
(select performer, count('x') denominator from performances group by performer)
NATURAL JOIN
(select avg(abs(MONTHS_BETWEEN(rec_date,when)))*30 avgdays,avg(abs(MONTHS_BETWEEN(rec_date,when))) avgmonths,
avg(abs(MONTHS_BETWEEN(rec_date,when)))/12 avgyears,performer, count('x') numerator from (
(select performer, songwriter, songtitle,max(rec_date) rec_date from(
(select performer, pair from albums)
NATURAL JOIN
(select pair,title songtitle, writer songwriter, rec_date from tracks)) group by (performer,songwriter,songtitle))
NATURAL JOIN
(select performer,songtitle, songwriter, when from performances))
where rec_date < when group by performer))where numerator < denominator
order by per_of_rec desc fetch next 10 rows only;
```

- **Tests:**

Query1:

To test the first query 1, we will first execute the query and select a performer and see the results that it gives and then we will individually check that the percentages are correct. The performer that we will be selecting for this test is 'Nici'.

PERFORMER_NAME	PER_OF_RECORDED_TRACKS_OWNED	PER_OF_CONCERT_PERF_OWNED
Alberto Fco. Vilchez	16,2037037	15,4639175
The Boozer Buzz	89,8617512	49,2883895
Dare Streets	40,9090909	46,9187675
Maria de la Salud Sanroman	23,880597	21,5789474
Nici	18,1184669	16,9338266
Muzicalizers	56,4784053	39,9630542

BAND	ORIGINAL_SONGS
Nici	52

BAND	TOTAL_SUM_SONGS
Nici	287

BAND	TOTAL_BAND_OWNED_PERFORMANCES
Nici	412

BAND	TOTAL_BAND_PERFORMANCES
Nici	2433

per_of_recorded_tracks_owned=(52/287)*100=18.1184668999
per_of_concert_perf_owned=(412/2433)*100=16.9338265516

With this we conclude that our query1 works

Now we test by deleting one row from performer Nici. It will be deleted on one of his albums and of a different writer than him, so that the percentage (18,1184669) should increase:
We can delete for instance, this row that is a track recorded in a Nici album but not written by him:

```
SQL> select * from involvement where band = 'Nici';

BAND                                               MUSICIAN       ROLE            START_D  END_D
-------------------------------------------------- -------------- --------------- -------- --------
Nici                                               ES>>0888249678 SOLIST          05/02/87

SQL> select performer,pair,sequ from (select performer, pair from albums where performer = 'Nici') NATURAL JOIN (select pair, sequ from tracks where writer != 'ES>>0888249678');

PERFORMER                                          PAIR                SEQU
-------------------------------------------------- --------------- ----------
Nici                                               G0595CNM2892DNC          1
```

```
SQL> delete from tracks where pair = 'G0595CNM2892DNC' and sequ = 1;

1 fila suprimida.
```

Now we execute again the query: As we can see the percentage has increased.

```
SQL> select band as performer_name,
  2  (original_songs/total_sum_songs)*100 as per_of_recorded_tracks_owned ,
  3  (total_band_owned_performances/total_band_performances)*100 as  per_of_concert_perf_owned from
  4  ((select band,count('x') original_songs from (select musician passport,band from involvement) NATURAL JOIN (select writer passport, title from tracks) group by band )
  5  natural join
  6  (select performer as band,count('x') total_sum_songs from (select pair, performer from albums) NATURAL JOIN (select pair, title from tracks) group by performer)
  7  natural join
  8  (select performer band,count('x') total_band_owned_performances from performances where songwriter in (select musician from involvement where band = performer)
  9  group by performer)
 10  natural join
 11  (select performer as band,count('x') total_band_performances from performances group by performer))
 12  where total_sum_songs >= original_songs and total_band_performances>=total_band_owned_performances
 13  and total_band_performances!=0 and total_sum_songs!=0 and band = 'Nici';

PERF PER_OF_RECORDED_TRACKS_OWNED PER_OF_CONCERT_PERF_OWNED
---- --------------------------- -------------------------
Nici                 18,1818182               16,9338266
```

Query2:

For this test, we decided to first of all choose the 10 best percentages below 100% because there were a lot that were in 100% so we considered that it would be more interesting to choose some that had 99,... percent.
For the test we selected one of the 10 best performances which is Rafi.

```
Rafi                                               99,895288 2697,22662 89,9075539 7,49229616
```

For it, we are going to select all the songs that he recorded, with the natural join between albums and tracks and then, with that result, another natural join with performances in which we join by performer, title and songwriter. This way we get all the songs that are recorded by Rafi which are: 954

```
select performer, count('x') from(
(select performer, songwriter, songtitle,max(rec_date) rec_date from(
(select performer, pair from albums)
NATURAL JOIN
(select pair,title songtitle, writer songwriter, rec_date from tracks)) group by (performer,songwriter,songtitle))
NATURAL JOIN
(select performer,songtitle, songwriter, when from performances)) where performer = 'Rafi' and rec_date < when
group by performer;
```

Now we select how many performances does Rafi have, which are 955

```
select performer, count('x') denominator from performances where performer = 'Rafi' group by performer;
```

If we do the ratio, we get 954/955 which gives 99,895288%

Now we test by deleting one row and observing how does the percentage change. We can delete the following row:

```
SQL> delete from tracks where pair = 'V1584NJO594507F' and sequ = 1;

1 fila suprimida.
```

And now, the top 10 are: where Rafi does not appear because we decreased his numerator to a point where another performer has overtaken him.

```
SQL> select performer, (numerator / denominator)*100 per_of_rec,avgdays,avgmonths,avgyears from (
  2  (select performer, count('x') denominator from performances group by performer)
  3  NATURAL JOIN
  4  (select avg(abs(MONTHS_BETWEEN(rec_date,when)))*30 avgdays,avg(abs(MONTHS_BETWEEN(rec_date,when))) avgmonths,
  5  avg(abs(MONTHS_BETWEEN(rec_date,when)))/12 avgyears,performer, count('x') numerator from (
  6  (select performer, songwriter, songtitle,max(rec_date) rec_date from(
  7  (select performer, pair from albums)
  8  NATURAL JOIN
  9  (select pair,title songtitle, writer songwriter, rec_date from tracks)) group by (performer,songwriter,songtitle))
 10  NATURAL JOIN
 11  (select performer,songtitle, songwriter, when from performances))
 12  where rec_date < when group by performer))where numerator < denominator
 13  order by per_of_rec desc fetch next 10 rows only;

PERFORMER                                          PER_OF_REC    AVGDAYS  AVGMONTHS   AVGYEARS
-------------------------------------------------- ---------- ---------- ---------- ----------
Elisi                                              99,9287749 3688,06314 122,935438 10,2446198
María de la Salud Sanroman                          99,924812 2950,20826 98,3402752 8,19502294
Vacas y Toros                                      99,9239544 2431,73442 81,0578141 6,75481784
Parqui                                             99,9174236 2759,77606 91,9925353 7,66604461
Plati                                              99,9158957 3404,03796 113,467932   9,455661
Rosa Fernanda Queiroz                              99,9122037 3609,45802 120,315267 10,0262723
Bully                                              99,9043977 3425,60457 114,186819 9,51556825
Los Sembrados                                      99,9025341 2261,17766 75,3725885 6,28104904
Beltran                                            99,8957247 2721,11051 90,7036837 7,55864031
Andri                                               99,893447  4116,4991 137,216637 11,4347197
```

3 Package

a) Design

In the malopack package, we included the following functions, procedures and variables:

performer_proc which is the procedure of part a of exercise 3. In this procedure, we assign a value to the variable **current_performer** that comes as a parameter in the procedure.

Then we have **insert_album** procedure, in which we insert an album and a song in the albums table. If the pair already exists, we only include the track. All the parameters needed for the insertion of a new track and album in their respective tables are given as parameters in the procedure. This was first part of part b of the exercise 3. In this procedure, first we check with a count that the number of albums with the pair given as a parameter, already exists, if count('x') = 0, we know that it does not exist so we have to insert the album and then insert the song. However, if count('x')!=0, that means that the album already exists and we only have to include the track. For checking this, we made use of a flag.

Additionally, we did the **delete_track** procedure in which we delete a track from an album and if the album only has 1 song, we delete the song and the album. In implicit semantics, we decided that if the album had no songs, we also delete it. This was second part of part b of the exercise 3. To do this procedure, we only needed 2 parameters which were sequ and pair as are the primary keys of tracks. We checked that there was a sequ in the pair that we give as a parameter, if there were no sequ, that means that the albums was empty so we just delete the album, if there was one sequ, that means that the album has only one song and that we have to delete that song and the album, if we have more than one sequ, we just delete the track.

Moreover, we did the **report** procedure which provides some information about the current performer such as the number of albums of each format, average number of songs per format of album, average album length and periodicity. For the collaborators, we were asked about name and type, number of works with the performer and percentage of works of each collaborator.

Finally, we included an additional function used to return the name of the current performer as we had problems accessing a package variable from the outside of the package. The name of this function is **get_current_performer**.

b) its implementation in SQL

Package

```
CREATE OR REPLACE PACKAGE melopack as
    FUNCTION get_current_performer return VARCHAR2;
    procedure performer_proc (performer varchar2);
    current_performer varchar2(50);
    PROCEDURE insert_album (new_pair CHAR,new_FORMAT CHAR,new_album_title VARCHAR2
    ,new_title CHAR,new_release_date DATE, new_publisher VARCHAR2, new_manager NUMBER,new_seq NUMBER
    ,new_writer VARCHAR2,new_duration NUMBER, new_rec_date DATE, new_studio VARCHAR2, new_engineer varchar2);
    PROCEDURE delete_track(new_pair CHAR, new_seq NUMBER);
    PROCEDURE report;
end melopack;
/
```

package body

```
exec melopack.performer_proc('Begga Ruiz')
CREATE OR REPLACE PACKAGE BODY melopack as
FUNCTION get_current_performer return VARCHAR2 is
aux varchar2(50);
begin
    aux := melopack.current_performer;
    return aux;
end;

PROCEDURE performer_proc(performer varchar2) is
begin
    current_performer := performer;
exception
    when no_data_found then dbms_output.put_line('No row returned');
    when too_many_rows then dbms_output.put_line('too many rows returned');
    when others then dbms_output.put_line('other error occurred');
end performer_proc;

PROCEDURE insert_album (new_pair CHAR,new_FORMAT CHAR,new_album_title VARCHAR2
,new_title CHAR,new_release_date DATE, new_publisher VARCHAR2, new_manager NUMBER,new_seq NUMBER
,new_writer VARCHAR2,new_duration NUMBER, new_rec_date DATE, new_studio VARCHAR2, new_engineer varchar2) is
flag NUMBER;
begin
    select count('x') into flag from (select album_pair from fsdb.recordings where album_pair = new_pair);
    if flag = 0
        then insert into Albums(pair,performer,format,title,rel_date,publisher,manager)
        values(new_pair,melopack.current_performer,new_FORMAT,new_album_title,new_release_date,new_publisher,new_manager);
        insert into Tracks(pair,sequ,title,writer,rec_date,studio,engineer,duration)
        values(new_pair,new_seq,new_title,new_writer,new_rec_date,new_studio,new_engineer,new_duration);
    else insert into Tracks(pair,sequ,title,writer,rec_date,studio,engineer,duration)
        values(new_pair,new_seq,new_title,new_writer,new_rec_date,new_studio,new_engineer,new_duration);
```

```
    end if;
exception
    when no_data_found then dbms_output.put_line('No row returned');
    when too_many_rows then dbms_output.put_line('too many rows returned');
    when others then dbms_output.put_line('other error occurred');
end;

PROCEDURE delete_track(new_pair CHAR, new_seq NUMBER) is
flag NUMBER;
begin
    select count('x') into flag from (select sequ from tracks where pair = new_pair);
    if flag = 0
    then delete from albums where pair = new_pair;
    elsif flag = 1
    then delete from Tracks where pair = new_pair and sequ = new_seq;
    delete from albums where pair = new_pair;
    else
    delete from Tracks where pair = new_pair and sequ = new_seq;
    end if;
exception
    when no_data_found then dbms_output.put_line('No row returned');
    when too_many_rows then dbms_output.put_line('too many rows returned');
    when others then dbms_output.put_line('other error occurred');
end;

PROCEDURE report is
begin
    dbms_output.put_line('Performer`s Statistics');
    dbms_output.put_line('----------------------------------');
    dbms_output.put_line('Number of albums per format');
    for myrow in (select format,count('x') num from albums where performer = melopack.current_performer group by format)
        loop
        dbms_output.put_line(myrow.format||'-'||myrow.num);
    end loop;
    dbms_output.put_line('----------------------------------');
    dbms_output.put_line('Average number of songs per format of album');
    for myrow in (select format,count('x'),avg(ntracks) average from (select pair,format from albums where performer =
melopack.current_performer)
        NATURAL JOIN
        (select count('x') ntracks, pair from tracks group by pair) group by format)
        loop
        dbms_output.put_line(myrow.format||'-'||myrow.average);
    end loop;
    dbms_output.put_line('----------------------------------');
    dbms_output.put_line('Average duration per format of album');
    for myrow in (select format,count('x'),avg(sumation) sumation from (select pair,format from albums where performer =
melopack.current_performer)
        NATURAL JOIN
        (select sum(duration) sumation, pair from tracks group by pair) group by format)
        loop
        dbms_output.put_line(myrow.format||'-'||myrow.sumation);
    end loop;
```

```
    dbms_output.put_line('----------------------------------');
    dbms_output.put_line('Periodicity');
    for myrow in (select datenum/num_alb months_between_albums from
        (select months_between(max(rel_date),min(rel_date)) datenum,count('x') num_alb
        from albums where performer = melopack.current_performer
        group by performer))
        loop
        dbms_output.put_line(myrow.months_between_albums);
    end loop;
    dbms_output.put_line('----------------------------------');
    dbms_output.put_line('Information for publisher');
    for myrow in (select publisher,npublisher,round((npublisher/ntotal)*100,2) pub_per from
    (select publisher, count('x') npublisher from (select distinct publisher,pair from albums where performer =
melopack.current_performer)group by publisher)
    NATURAL JOIN
    (select count('x') ntotal from(select distinct performer ,publisher,pair from albums where performer =
melopack.current_performer)group by performer))
        loop
        dbms_output.put_line('PUBLISHER: '||myrow.publisher||'| Number of works: '||myrow.npublisher||' -> percentage:
'||myrow.pub_per);
    end loop;
    dbms_output.put_line('----------------------------------');
    dbms_output.put_line('Information for studio');
    for myrow in (select studio,nstudio, round((nstudio/ntotal)*100,2) stu_per from (
    (select studio , count('x') nstudio from((select pair from albums where performer = melopack.current_performer) NATURAL
JOIN (select pair, studio, engineer from tracks)) group by studio)
    NATURAL JOIN
    (select count('x') ntotal from ((select performer,pair from albums where performer = melopack.current_performer) NATURAL
JOIN (select pair, studio from tracks)) group by performer)))
        loop
        dbms_output.put_line('STUDIO: '||myrow.studio||'| Number of works: '||myrow.nstudio||' -> percentage '||myrow.stu_per);
    end loop;

    dbms_output.put_line('----------------------------------');
    dbms_output.put_line('Information for Engineer');
    for myrow in (select engineer,nengineer, round((nengineer/ntotal)*100,2) as eng_per from (
    (select engineer , count('x') nengineer from((select pair from albums where performer = melopack.current_performer) NATURAL
JOIN (select pair, engineer from tracks)) group by engineer)
    NATURAL JOIN
    (select count('x') ntotal from ((select performer,pair from albums where performer = melopack.current_performer) NATURAL
JOIN (select pair,engineer from tracks)) group by performer)))
        loop
        dbms_output.put_line('ENGINEER: '||myrow.engineer||'| Number of works: '||myrow.nengineer||' -> percentage
'||myrow.eng_per);
    end loop;
    dbms_output.put_line('----------------------------------');
    dbms_output.put_line('Information for album manager');
    for myrow in (select manager,nmanager, round((nmanager/ntotal)*100,2) mana_per from (
    (select manager , count('x') nmanager from((select pair, manager from albums where performer = melopack.current_performer))
group by manager)
    NATURAL JOIN
    (select count('x') ntotal from (select performer,pair,manager from albums where performer = melopack.current_performer)
```

```
group by performer)))
        loop
        dbms_output.put_line('ALBUM MANAGER: '||myrow.manager||'| Number of works: '||myrow.nmanager||' -> percentage
'||myrow.mana_per);
    end loop;
    dbms_output.put_line('---------------------------------');
    dbms_output.put_line('Information for concerts manager');
    for myrow in (select manager,nmanager, round((nmanager/ntotal)*100,2) as con_man_per from (
    (select manager , count('x') nmanager from((select manager from concerts where performer = melopack.current_performer))
group by manager)
    NATURAL JOIN
    (select count('x') ntotal from (select performer,manager from concerts where performer = melopack.current_performer) group
by performer)))
        loop
        dbms_output.put_line('CONCERT MANAGER: '||myrow.manager||'| Number of works: '||myrow.nmanager||' -> percentage
'||myrow.con_man_per);
    end loop;
    dbms_output.put_line('---------------------------------');
exception
    when no_data_found then dbms_output.put_line('No row returned');
    when too_many_rows then dbms_output.put_line('too many rows returned');
    when others then dbms_output.put_line('other error occurred');
end report;

end melopack;
/
```

c) tests

First test will consist on assigning a value to the variable current_performer. For this, we will execute the following code using the **performer_proc** procedure:

```
exec melopack.performer_proc('Begga Ruiz')
```

To check if it has been done correctly, we print the content of current_performer and it should be Begga Ruiz:

```
begin dbms_output.put_line(melopack.current_performer); end; /
```

The output is:

```
SQL> begin dbms_output.put_line(melopack.current_performer); end;
  2  /
Begga Ruiz

Procedimiento PL/SQL terminado correctamente.
```

For the **insert_album**, we will execute the procedure and introduce an album with name: '72 Seasons' and pair = Z1290OHZ7079WKA so that we can distinguish that it has been created.

After executing, we should have a row in albums with the name '72 Seasons' and a new row in tracks with engineer Ulrich which was not in the tracks before The code is:

```
exec melopack.insert_album('Z1290OHZ7079WKA','S','72 Seasons','Holiday of
blues','28/11/22','Archer',555399004,2,'SE>>0222003242',275,'17/08/95','Stretchers
Studios','Ulrich')
```

The output:

```
SQL> exec melopack.insert_album('Z1290OHZ7079WKA','S','72 Seasons','Holiday of blues','28/11/22','Archer',555399004,2,'SE>>0222003242',275,'17/08/95','Stretchers Studios','Ulrich')
Procedimiento PL/SQL terminado correctamente.
SQL> select * from albums where title = '72 Seasons';
PAIR            PERFORMER                                          F TITLE                                            REL_DATE PUBLISHER                  MANAGER
--------------- -------------------------------------------------- - -------------------------------------------------- -------- -------------------- ----------
Z1290OHZ7079WKA Begga Ruiz                                         S 72 Seasons                                         28/11/22 Archer                  555399004
SQL> select * from tracks where engineer = 'Ulrich';
PAIR                SEQU TITLE                                          WRITER          DURATION REC_DATE STUDIO                                                ENGINEER
--------------- ---------- -------------------------------------------------- -------------- ---------- -------- -------------------------------------------------- ----------
Z1290OHZ7079WKA          2 Holiday of blues                                   SE>>0222003242         275 17/08/95 Stretchers Studios                                 Ulrich
```

This occurs when the album don't exist, but if the album exists, it will only include the track. We will try with the album with pair : I7397KU62383S88 that already exists and belongs to Begga Ruiz. Now we just include the song with engineer = 'Ulrich';

```
exec melopack.insert_album('I7397KU62383S88','S','72 Seasons','Holiday of
blues','28/11/22','Archer',555399004,3,'SE>>0222003242',275,'17/08/95','Stretchers
Studios','Ulrich')
```

The output:

```
SQL> exec melopack.insert_album('Z1290OHZ7079WKA','S','72 Seasons','Holiday of blues','28/11/22','Archer',555399004,3,'SE>>0222003242',275,'17/08/95','Stretchers Studios','Ulrich')
Procedimiento PL/SQL terminado correctamente.
SQL> select * from tracks where engineer = 'Ulrich';
PAIR                SEQU TITLE                                          WRITER          DURATION REC_DATE STUDIO                                                ENGINEER
--------------- ---------- -------------------------------------------------- -------------- ---------- -------- -------------------------------------------------- ----------
Z1290OHZ7079WKA          3 Holiday of blues                                   SE>>0222003242         275 17/08/95 Stretchers Studios                                 Ulrich
```

In procedure **delete_track** first we insert the album, using the previous function, and now, we just include the parameters pair and sequ into the delete_track function and execute it.
First, this is the insertion:

```
SQL> exec melopack.performer_proc('Begga Ruiz')
Procedimiento PL/SQL terminado correctamente.
SQL> exec melopack.insert_album('Z1290OHZ7079WKA','S','72 Seasons','Holiday of blues','28/11/22','Archer',555399004,3,'SE>>0222003242',275,'17/08/95','Stretchers Studios','Ulrich')
Procedimiento PL/SQL terminado correctamente.
```

And this is the deletion:

```
SQL> exec melopack.delete_track('Z1290OHZ7079WKA',3)
Procedimiento PL/SQL terminado correctamente.
SQL> select * from albums where pair = 'Z1290OHZ7079WKA';
ninguna fila seleccionada
```

Finally in the **report** procedure, for the number of albums per format, average number of songs per format of album and average duration per format of album, we will use the format 'T' for current performer 'Begga Ruiz'. This is the execution of the report for the performer:

```
SQL> exec melopack.report
Performer`s Statistics
-----------------------------------
Number of albums per format
C-5
S-22
T-2
V-18
-----------------------------------
Average number of songs per format of album
V-13,77777777777777777777777777777777777778
C-6,8
T-2
S-2
-----------------------------------
Average duration per format of album
V-3877,22222222222222222222222222222222222222
C-1951,4
T-470,5
S-572,54545454545454545454545454545454545455
-----------------------------------
Periodicity
8,43239533287577213452299245024021962938
-----------------------------------
Information for publisher
PUBLISHER: United Vinyls  -> 100
-----------------------------------
```

The number of albums per format:

```
SQL> select * from albums where format = 'T' and performer = 'Begga Ruiz';

PAIR           PERFORMER                                          F TITLE                                              REL_DATE PUBLISHER                 MANAGER
-------------- -------------------------------------------------- - -------------------------------------------------- -------- ------------------------- ---------
Q4027WZ13830W4Z Begga Ruiz                                        T Discussion or sad willows                          17/09/99 United Vinyls             555336234
Y15975EO6600KBF Begga Ruiz                                        T Waltz pictures                                     09/01/00 United Vinyls             555336234
```

Average number of songs per format album: (2+2)/2=2
Average duration per format of album (using the below data)
(259+199+229+254)/2 = 470,5

```
SQL> select * from tracks where pair = 'Q4027WZ13830W4Z';

PAIR                SEQU TITLE                                              WRITER          DURATION REC_DATE STUDIO                                             ENGINEER
--------------- ---------- -------------------------------------------------- --------------- ---------- -------- -------------------------------------------------- --------------------
Q4027WZ13830W4Z          1 Discussion or sad willows                          SE>>0308420752         259 11/02/99 Ipurr Recordings                                   Mariana Henriquez Ipurre
Q4027WZ13830W4Z          2 Discussion or sad willows (unplugged)              SE>>0308420752         199 12/02/99 Stretchers Studios                                 Milagros Francia Cuadra

SQL> select * from tracks where pair = 'Y15975EO6600KBF';

PAIR                SEQU TITLE                                              WRITER          DURATION REC_DATE STUDIO                                             ENGINEER
--------------- ---------- -------------------------------------------------- --------------- ---------- -------- -------------------------------------------------- --------------------
Y15975EO6600KBF          1 Waltz pictures                                     SE>>0176485161         229 22/10/99 Ipurr Recordings                                   Mariana Henriquez Ipurre
Y15975EO6600KBF          2 Petunia or fiction                                 SE>>0026547375         254 11/10/99 Ipurr Recordings                                   Mariana Henriquez Ipurre
```

Periodicity:

```
SQL> select max(rel_date) from albums where performer = melopack.get_current_performer;

MAX(REL_
--------
04/05/02

SQL> select min(rel_date) from albums where performer = melopack.get_current_performer;

MIN(REL_
--------
25/04/69

SQL> select months_between(max(rel_date),min(rel_date)) from albums where performer = melopack.get_current_performer;

MONTHS_BETWEEN(MAX(REL_DATE),MIN(REL_DATE))
-------------------------------------------
                                396,322581

SQL> select count('x') from albums where performer = melopack.get_current_performer;

COUNT('X')
----------
        47
```

Now we show the info about collaborators:
Publisher: (This comes from the report with performer = 'Begga Ruiz')

```
Information for publisher
PUBLISHER: United Vinyls | Number of works: 47 -> percentage: 100
```

To test the publisher: All albums are published by the same publisher. 47/47 *100 = 100

```
SQL> select count('x') from albums where performer = 'Begga Ruiz' and publisher = 'United Vinyls ';

COUNT('X')
----------
        47

SQL> select count('x') from albums where performer = 'Begga Ruiz';

COUNT('X')
----------
        47
```

Studio: (This comes from the report with performer = 'Begga Ruiz')

```
STUDIO: B.P.O. Studios| Number of works: 18 -> percentage 5,45
```

To test studio: 18/330 = 5,45

```
SQL> select count('x')
  2  from (select performer, pair from albums where performer = 'Begga Ruiz')
  3  NATURAL JOIN
  4  (select sequ, pair from tracks where studio = 'B.P.O. Studios') group by performer;

COUNT('X')
----------
        18

SQL> select count('x')
  2  from (select performer, pair from albums where performer = 'Begga Ruiz')
  3  NATURAL JOIN
  4  (select sequ, pair from tracks) group by performer;

COUNT('X')
----------
       330
```

Engineer: (This comes from the report with performer = 'Begga Ruiz')

```
ENGINEER: Hugo Nakama Hernandez-Llorente| Number of works: 15 -> percentage 4,55
```

To test engineer: 15/330 = 4,545

```
SQL> select count('x')
  2  from (select performer, pair from albums where performer = 'Begga Ruiz')
  3  NATURAL JOIN
  4  (select sequ, pair from tracks where engineer = 'Hugo Nakama Hernandez-Llorente');

COUNT('X')
----------
        15

SQL> select count('x')
  2  from (select performer, pair from albums where performer = 'Begga Ruiz')
  3  NATURAL JOIN
  4  (select sequ, pair from tracks) group by performer;

COUNT('X')
----------
       330
```

Album manager: (This comes from the report with performer = 'Begga Ruiz')

```
Information for album manager
ALBUM MANAGER: 555399004| Number of works: 5 -> percentage 10,64
```

To test album manager: 5/47 * 100 = 10,638

```
SQL> select count('x') from albums where performer = 'Begga Ruiz' and manager = '555399004';

COUNT('X')
----------
         5

SQL> select count('x') from albums where performer = 'Begga Ruiz';

COUNT('X')
----------
        47
```

Concert manager: (This comes from the report with performer = 'Begga Ruiz')

```
CONCERT MANAGER: 555707910| Number of works: 2 -> percentage 1,07
```

To test concert manager: 2/187 *100= 1,069

```
SQL> select count('x') from concerts where performer = 'Begga Ruiz' and manager = '555707910';

COUNT('X')
----------
         2

SQL> select count('x') from concerts where performer = 'Begga Ruiz';

COUNT('X')
----------
       187
```

4 External Design

a) its design in relational algebra

View1:

$$\Pi_{performer,\ pair,\ sumation}(Q)$$

$$Q \equiv \left[\Pi_{pair,\ sum(duration)\ as\ sumation}\left(\mathcal{G}_{pair}(tracks)\right)\right] * \left[\Pi_{pair,\ performer}\left(\sigma_{performer\ =\ melopack.get\_current\_performer}(albums)\right)\right]$$

(pair)

View2:

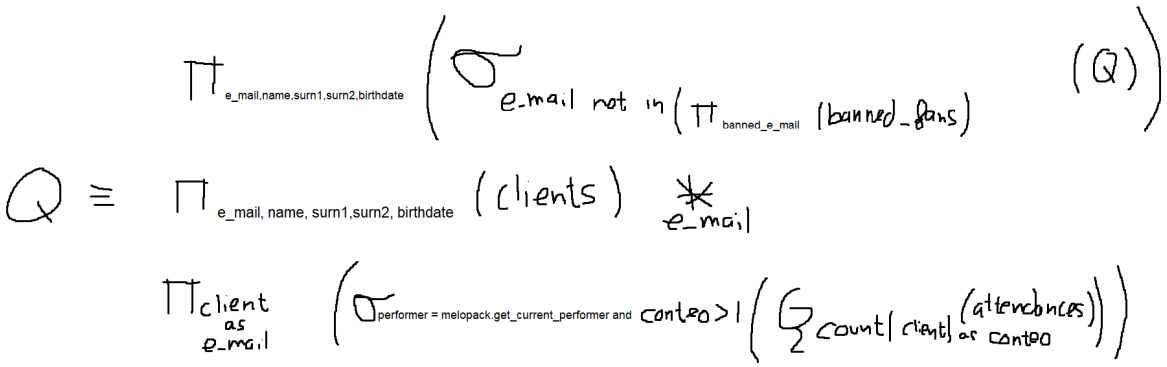
$$\Pi_{\text{to_char(when,'MM-YY') date_event,count('x') nconcerts,avg(nperformances) avgperformances,sum(attendance) sumatt, avg(duration) avgdur}}\left(\mathcal{G}_{\text{to_char(when,'MM-YY')}}(Q1)\right)$$

$$Q \equiv \left[\Pi_{performer,when,attendance,duration}\left(\sigma_{performer\ =\ melopack.get\_current\_performer}(concerts)\right)\right] * \left[\Pi_{performer,\ when,\ count('x')\ as\ nperformances}\left(\sigma_{performer\ =\ melopack.get\_current\_performer}\left(\mathcal{G}_{performer,when}(performances)\right)\right)\right]$$

(performer, when)

View3:

The code provided in the next section for the view 3 contains several components, firstly we find the assignment of a current performer into the variable current_performer in the melopack to execute the view3 for that current performer.
Then we create the view fans.
Then we create the auxiliary table banned_fans to get track of those fans that we will delete when deleting on the view, using the del_fan trigger.
Now we find the del_fan trigger to prevent the deletion of fans in the base table and to send them to banned_fans.
Finally, we find the ins_fan trigger, to insert fans in the view, what we really do is to, if we receive a new client, we insert him in clients and then we insert him in attendances for the last 2 concerts of the current performer. If we receive a client that already exists but has not attended to any of the concerts of the current performer, we insert him in attendances for the last 2 concerts of the current performer. If we receive a client that only went to one of the concerts of the current performer, we insert him in the last, or penultimate concert in attendances. Finally if we receive a client that is banned, we just eliminate him from the banned_fans table.
We have altered the attendances table so that the attribute rfid is optional because when inserting we don't have that value and we have also deleted this row from table attendance: CONSTRAINT UK_ATTENDANCES UNIQUE (performer,when), because we were having problems when inserting in attendances the null value of rfid. It was not enough with removing the not null from the rfid.



b) its implementation in SQL

View1:

```
CREATE OR REPLACE VIEW my_albums AS (select performer,pair,sumation from (select pair, sum(duration) sumation from tracks
group by pair)
NATURAL JOIN (select pair, performer from albums) where performer = melopack.get_current_performer) WITH READ ONLY;
```

view2:

```
CREATE OR REPLACE VIEW events AS (select to_char(when,'MM-YY') date_event,count('x') nconcerts,avg(nperformances)
avgperformances,sum(attendance) sumatt, avg(duration) avgdur
from (select performer,when,attendance,duration from concerts
where performer = melopack.get_current_performer)
NATURAL JOIN (select count('x') nperformances,performer, when from performances
where performer = melopack.get_current_performer group by(performer,when))
group by to_char(when,'MM-YY')) WITH READ ONLY;
```

View3:

```
exec melopack.performer_proc('Amapola')

CREATE OR REPLACE VIEW fans AS (select distinct e_mail,name,surn1,surn2,birthdate from(
(select e_mail, name, surn1,surn2, birthdate from clients)
NATURAL JOIN
(select client e_mail from attendances where performer = melopack.get_current_performer group by client having count('x')>1))
where e_mail not in (select banned_e_mail from banned_fans));

Create table banned_fans(
    banned_e_mail varchar2(100) NOT NULL,
    banned_performer varchar2(50) NOT NULL,
    CONSTRAINT PK_banned_fans PRIMARY KEY(banned_e_mail),
    CONSTRAINT FK_Clients20 FOREIGN KEY (banned_e_mail) REFERENCES CLIENTS,
    CONSTRAINT FK_Performer20 FOREIGN KEY (banned_performer) REFERENCES performers
);

Create or replace Trigger del_fan
instead of delete on fans
for each row
begin
    insert into banned_fans(banned_e_mail,banned_performer) values(:OLD.e_mail,melopack.get_current_performer);
end del_fan;
/
Create or replace Trigger ins_fan
instead of insert on fans
for each row
DECLARE
flag number :=-1;
aux_date date;
aux_date2 date;
aux_date3 date;
begin
    select count('x') into flag from (select banned_e_mail from banned_fans where banned_e_mail = :NEW.e_mail);
    if flag >0
```

```
        then delete from banned_fans where banned_e_mail = :NEW.e_mail;
    end if;
    select count('x') into flag from (select e_mail from clients where e_mail = :NEW.e_mail);
    if flag = 0
        then
        select max(when) into aux_date from attendances where performer = melopack.get_current_performer;
        select max(when) into aux_date2 from attendances where performer = melopack.get_current_performer and when !=
aux_date;
        insert into clients(e_mail,name,surn1,surn2,birthdate)
        values (:NEW.e_mail,:NEW.name,:NEW.surn1,:NEW.surn2,:NEW.birthdate);
        insert into attendances(client,performer,when,purchase)
        values (:NEW.e_mail,melopack.get_current_performer,aux_date,sysdate);
        insert into attendances(client,performer,when,purchase)
        values (:NEW.e_mail,melopack.get_current_performer,aux_date2,sysdate);
    else
    select count('x') into flag from attendances where client = :NEW.e_mail and performer = melopack.get_current_performer;
    if flag = 0
        then
        select max(when) into aux_date from attendances where performer = melopack.get_current_performer;
        select max(when) into aux_date2 from attendances where performer = melopack.get_current_performer and when !=
aux_date;
        insert into attendances(client,performer,when,purchase)
        values (:NEW.e_mail,melopack.get_current_performer,aux_date,sysdate);
        insert into attendances(client,performer,when,purchase)
        values (:NEW.e_mail,melopack.get_current_performer,aux_date2,sysdate);
    end if;
    if flag = 1
        then
            select max(when) into aux_date from attendances where performer = melopack.get_current_performer;
            select max(when) into aux_date2 from attendances where performer = melopack.get_current_performer and when !=
aux_date;
            select max(when) into aux_date3 from attendances where performer = melopack.get_current_performer and client =
:NEW.e_mail;
            if aux_date3 = aux_date
                then
                insert into attendances(client,performer,when,purchase)
                values (:NEW.e_mail,melopack.get_current_performer,aux_date2,sysdate);
            else
                insert into attendances(client,performer,when,purchase)
                values (:NEW.e_mail,melopack.get_current_performer,aux_date,sysdate);
            end if;
    end if;
    end if;
end ins_fan;
/
```

c) **Tests:**

View1:

For this view, which is read only, no trigger is necessary. However we have to assign a performer to the current_performer variable by executing:

```
exec melopack.performer_proc('Begga Ruiz')
```

First we execute only the query:

```
PERFORMER                                          PAIR              SUMATION
-------------------------------------------------- --------------- ----------
Begga Ruiz                                         A7172J6J40518FA          645
Begga Ruiz                                         E5659BON77253MU          592
Begga Ruiz                                         G9398NXR4450BO          3661
Begga Ruiz                                         H28157WI331340K          636
Begga Ruiz                                         J69605QE6675OFW          596
Begga Ruiz                                         K39044XZ4494QYU         4256
Begga Ruiz                                         S5651CCD2823H0U          642
Begga Ruiz                                         N4996TUZ114QJK           488
Begga Ruiz                                         S7983UBG32792A4         4112
Begga Ruiz                                         Z3564L7O6774X2G         4147
Begga Ruiz                                         Z3832DPO9276WT7         3496

PERFORMER                                          PAIR              SUMATION
-------------------------------------------------- --------------- ----------
Begga Ruiz                                         E1921ZA3360QPQ           477
Begga Ruiz                                         I7397KU62383S88          546
Begga Ruiz                                         M8630RYB3494XPQ          496

47 filas seleccionadas.
```

Then, when executing the code in oracle, we don't receive any error and the view is created without problems.

```
SQL> CREATE OR REPLACE VIEW my_albums AS (select performer,pair,sumation from (select pair, sum(duration) sumation from tracks group by pair)
  2  NATURAL JOIN (select pair, performer from albums) where performer = melopack.get_current_performer) WITH READ ONLY;

Vista creada.

SQL> select * from my_albums;

PERFORMER                                          PAIR              SUMATION
-------------------------------------------------- --------------- ----------
Begga Ruiz                                         F1870DW71242V5V         3589
Begga Ruiz                                         H7819HXH9504482          477
Begga Ruiz                                         H8601O6P82865DO         3739
Begga Ruiz                                         I23365ZN479R82           552
Begga Ruiz                                         M1312RCQ2981SZA         4032
Begga Ruiz                                         R7999ZFT4024L11          459
Begga Ruiz                                         Z3474DAL6343L3P         3787
Begga Ruiz                                         B6779YLQ1233578         4080
Begga Ruiz                                         J2727TNN8373H91         3816
Begga Ruiz                                         L53816D8205GQL           640
Begga Ruiz                                         M4294JB37859PZW          621

PERFORMER                                          PAIR              SUMATION
-------------------------------------------------- --------------- ----------
Begga Ruiz                                         X57149LS223085           640
```

We can also test the deletion of one album and see how instead of 47 rows we get 46 by executing the exact same query:

```
SQL> delete from albums where pair = 'H7819HXH9504482';

1 fila suprimida.
```

```
46 filas seleccionadas.
```

View2:

This second view is also read only, so there is no need for a trigger.
First we execute the query with the current performer (in this case Amapola):

```
SQL> select to_char(when,'MM-YY') date_event,count('x') nconcerts,avg(nperformances) avgperformances,sum(attendance) sumatt, avg(duration) avgdur
  2  from (select performer,when,attendance,duration from concerts
  3  where performer = melopack.get_current_performer)
  4  NATURAL JOIN (select count('x') nperformances,performer, when from performances
  5  where performer = melopack.get_current_performer group by(performer,when))
  6  group by to_char(when,'MM-YY');

DATE_  NCONCERTS AVGPERFORMANCES     SUMATT     AVGDUR
----- ---------- --------------- ---------- ----------
09-98          3              11          0        141
10-15          2              12          0        121
05-91          1              10          0        107
09-92          2              11          0        118
08-95          2              11          0        137
08-97          1              11          0        122
10-10          3              12          0        117
11-18          1              12          0        145
06-92          2              11          0        126
05-18          2              12          0        145
05-20          1              12          0        145
```

Now we create and select from the view:

```
SQL> CREATE OR REPLACE VIEW events AS (select to_char(when,'MM-YY') date_event,count('x') nconcerts,avg(nperformances) avgperformances,sum(attendance) sumatt, avg(duration) avgdur
  2  from (select performer,when,attendance,duration from concerts
  3  where performer = melopack.get_current_performer)
  4  NATURAL JOIN (select count('x') nperformances,performer, when from performances
  5  where performer = melopack.get_current_performer group by(performer,when))
  6  group by to_char(when,'MM-YY')) WITH READ ONLY;

Vista creada.

SQL> select * from events;

DATE_  NCONCERTS AVGPERFORMANCES     SUMATT     AVGDUR
----- ---------- --------------- ---------- ----------
09-98          3              11          0        141
10-15          2              12          0        121
05-91          1              10          0        107
09-92          2              11          0        118
08-95          2              11          0        137
08-97          1              11          0        122
10-10          3              12          0        117
11-18          1              12          0        145
06-92          2              11          0        126
05-18          2              12          0        145
05-20          1              12          0        145
```

Finally, we can check that the query works by selecting and seeing how many rows are there in the month 09-98:

```
SQL> select * from concerts where to_char(when,'MM-YY') = '09-98' and performer = 'Amapola';

PERFORMER                                 WHEN     TOUR                                                                                                MUNICIPALITY
----------------------------------------- -------- --------------------------------------------------------------------------------------------------- ----------------------------------------
Amapola                                   06/09/98 La Shuffle '98 Tourn?e                                                                              Streamforest
Amapola                                   13/09/98 La Shuffle '98 Tourn?e                                                                              Mellonside
Amapola                                   18/09/98 La Shuffle '98 Tourn?e                                                                              Smilesside

SQL>
```

Finally we can try deleting some row. In another case, when executing the query for performer = Begga Ruiz, we get 99 rows:

```
DATE_  NCONCERTS AVGPERFORMANCES     SUMATT     AVGDUR
----- ---------- --------------- ---------- ----------
05-84          3              11          0        145
09-85          1              11          0        121
08-86          4              11          0        104
05-98          2              12          0        131
06-85          2              11          0        121
07-01          1              12          0        147
08-92          1              12          0        138
09-03          2              12          0        133
07-91          2              12          0        142
08-91          1              12          0        142
10-91          1              12          0        142

99 filas seleccionadas.
```

Now we delete one (Month-year) for instance 10-91 that only has one concert so that we will receive 98 rows instead of 99:

```
SQL> delete from concerts where to_char(when,'MM-YY') = '10-91' and performer = 'Begga Ruiz';

1 fila suprimida.
```

Now 98 rows:

```
DATE_  NCONCERTS AVGPERFORMANCES     SUMATT     AVGDUR
----- ---------- --------------- ---------- ----------
05-84          3              11          0        145
09-85          1              11          0        121
08-86          4              11          0        104
05-98          2              12          0        131
06-85          2              11          0        121
07-01          1              12          0        147
08-92          1              12          0        138
09-03          2              12          0        133
07-91          2              12          0        142
08-91          1              12          0        142

98 filas seleccionadas.
```

View3:

First we are going to test the query alone, first we show how many rows appear, and then we will delete one and see how many appear again: (everything for performer Amapola)

```
E_MAIL                                                           NAME                                               SURN1
---------------------------------------------------------------- -------------------------------------------------- --------
tallest@clients.vinylinc.com                                     Mario Ramon                                        Medina
canchumanya@clients.vinylinc.com                                 Felix                                              Aragone
diegoarroyo@clients.vinylinc.com                                 Diego                                              Arroyo
zuti@clients.vinylinc.com                                        Zutano                                             Torres
repe@clients.vinylinc.com                                        Victor Andres                                      Perez
arani@clients.vinylinc.com                                       Roberto Oscar                                      Mellado

83 filas seleccionadas.
```

We receive 83 rows, now we delete one client that has 2 attendances and see how the number goes to 82 (leona@clients.vinylinc.com has 2 attendances).

```
SQL> delete from attendances where when = '22/06/19' and performer = 'Amapola' and client = 'leona@clients.vinylinc.com';

1 fila suprimida.
```

we delete and now:

```
E_MAIL                                                           NAME
---------------------------------------------------------------- --------------------------------------------------
tallest@clients.vinylinc.com                                     Mario Ramon
arani@clients.vinylinc.com                                       Roberto Oscar
diegoarroyo@clients.vinylinc.com                                 Diego
zuti@clients.vinylinc.com                                        Zutano
repe@clients.vinylinc.com                                        Victor Andres

82 filas seleccionadas.
```

1 row less as the row of client leona@clients.vinylinc.com has been removed because we deleted one of the two attendances that the client had to the current performer in this case 'Amapola'.

Now we will delete several rows from fans and see how the del_fan works as well as how is the banned_fans auxiliary table increasing.

```
delete from fans where e_mail = 'repe@clients.vinylinc.com' and name = 'Victor Andres' and surn1 = 'Perez' and surn2 =
'Matayoshi' and birthdate = '09/09/87';
delete from fans where e_mail = 'arani@clients.vinylinc.com' and name = 'Roberto Oscar' and surn1 = 'Mellado' and surn2 =
'Arana' and birthdate = '17/02/53';
delete from fans where e_mail = 'diegoarroyo@clients.vinylinc.com' and name = 'Diego' and surn1 = 'Arroyo' and surn2 =
'Ruiz' and birthdate = '15/01/99';
```

```
SQL> delete from fans where e_mail = 'repe@clients.vinylinc.com' and name = 'Victor Andres' and surn1 = 'Perez' and surn2 = 'Matayoshi' and birthdate = '09/09/87';

1 fila suprimida.

SQL> delete from fans where e_mail = 'arani@clients.vinylinc.com' and name = 'Roberto Oscar' and surn1 = 'Mellado' and surn2 = 'Arana' and birthdate = '17/02/53';

1 fila suprimida.

SQL> delete from fans where e_mail = 'diegoarroyo@clients.vinylinc.com' and name = 'Diego' and surn1 = 'Arroyo' and surn2 = 'Ruiz' and birthdate = '15/01/99';

1 fila suprimida.
```

Now we see how many rows are there in the view and in the banned_ fans table:
When selecting from the view:

```
E_MAIL                                  NAME                                  SURN1
--------------------------------------- ------------------------------------- ------
canchumanya@clients.vinylinc.com        Felix                                 Aragon
bravi@clients.vinylinc.com              Ursulina                              Nunez-
zuti@clients.vinylinc.com               Zutano                                Torres

80 filas seleccionadas.
```

when selecting from the banned_fans table:

```
SQL> select * from banned_fans;

BANNED_E_MAIL                                   BANNED_PERFORMER
----------------------------------------------- ------------------------------
repe@clients.vinylinc.com                       Amapola
arani@clients.vinylinc.com                      Amapola
diegoarroyo@clients.vinylinc.com                Amapola
```

Finally, we can test how does the insert trigger (ins_fan) works:
We insert first a client that does not exist, then a client that has only one attendance to the current performer and then a fan that exists but has not attended any concert of the current performer:

```
insert into fans(e_mail,name,surn1,surn2,birthdate) values('123456@gmail.com','Guillermo','Gonzalez','Perez','10-10-01');
insert into fans(e_mail,name,surn1,surn2,birthdate) values('cuy@clients.vinylinc.com','Maximino','Cuy','Lopez','27/02/93');
insert into fans(e_mail,name,surn1,surn2,birthdate) values('cruz@clients.vinylinc.com','Maria de la
Cruz','Garcia','Gago','29/12/75');
```

This is the new client '123456@gmail.com' and his 2 new attendances to a concert of 'Amapola':

```
SQL> select * from clients where e_mail = '123456@gmail.com';

E_MAIL                                  NAME                                  SURN1
--------------------------------------- ------------------------------------- -------
123456@gmail.com                        Guillermo                             Gonzale
```

```
SQL> select * from attendances where performer = 'Amapola' and client = '123456@gmail.com';

CLIENT                                  PERFORMER
--------------------------------------- ---------------------------------------
123456@gmail.com                        Amapola
123456@gmail.com                        Amapola

SQL> _
```

Now we check the new attendance of already existent client with only one attendance to a concert of Amapola '`cuy@clients.vinylinc.com`' :

```
CLIENT                                                                                   PERFORMER
---------------------------------------------------------------------------------------- ----------
cuy@clients.vinylinc.com                                                                 Amapola
cuy@clients.vinylinc.com                                                                 Amapola
```

```
        WHEN     RFID
------- -------- ----------------------------------------------------------------
        07/09/18 M3GFG0UVO8TFTKMEZO05MB7LCMBZ3D9O9VNLQILGL9EG672RP4QGL38ZPSFKD3JX
        25/09/20
```

Makes sense that one has rfid and the other not because the second row has been inserted by us, and when we insert rfid, we insert null.

And finally, we check the 2 new attendances of already existent client but without any attendance to a concert of Amapola '`cruz@clients.vinylinc.com`' :

```
SQL> select * from attendances where performer = 'Amapola' and client = 'cruz@clients.vinylinc.com';

CLIENT                                                                                   PERFORMER
---------------------------------------------------------------------------------------- ---------
cruz@clients.vinylinc.com                                                                Amapola
cruz@clients.vinylinc.com                                                                Amapola
```

```
                WHEN     RFID                                                              PURCHASE
--------------- -------- ----------------------------------------------------------------- --------
                12/09/20                                                                   29/03/23
                25/09/20                                                                   29/03/23
```

Finally, we check how we insert a client in fans when that client was in banned:
lets ban this client: 'tied@clients.vinylinc.com', now we insert into fans that client:
We delete:

```
delete from fans where e_mail = 'tied@clients.vinylinc.com';
```

```
SQL> delete from fans where e_mail = 'tied@clients.vinylinc.com';

1 fila suprimida.

SQL> select * from banned_fans;

BANNED_E_MAIL                                                                BANNED_PERFORMER
---------------------------------------------------------------------------- ----------------
tied@clients.vinylinc.com                                                    Amapola
```

And now we insert a banned client:

```
insert into fans(e_mail,name,surn1,surn2,birthdate)
values('tied@clients.vinylinc.com','Maria Jose','Malca','Orendo','20/01/66');
```

```
SQL> insert into fans(e_mail,name,surn1,surn2,birthdate) values('tied@clients.vinylinc.com','Maria Jose','Malca','Orendo','20/01/66');

1 fila creada.

SQL> select * from banned_fans;

ninguna fila seleccionada
```

Finally, we test the trigger with a bulk insert:

```
--test view3
--create test table
create table test_table (
    e_mail VARCHAR2(100),
    name VARCHAR2(80),
    surn1 VARCHAR2(80),
    surn2 VARCHAR2(80),
    birthdate DATE);

--insert rows on the test table
insert into test_table(e_mail,name,surn1,surn2,birthdate)
values('123456@gmail.com','Guillermo','Gonzalez','Perez','10-10-01');
insert into test_table(e_mail,name,surn1,surn2,birthdate)
values('cuy@clients.vinylinc.com','Maximino','Cuy','Lopez','27/02/93');
insert into test_table(e_mail,name,surn1,surn2,birthdate) values('cruz@clients.vinylinc.com','Maria de la
Cruz','Garcia','Gago','29/12/75');
--bulk insert into fans
exec melopack.performer_proc('Amapola')
insert into fans (e_mail,name,surn1,surn2,birthdate) (select * from test_table);
--drop test table
drop table test_table;
```

```
SQL> insert into fans (e_mail,name,surn1,surn2,birthdate) (select * from test_table);

6 filas creadas.
```

We get the same as before:

```
SQL> select * from clients where e_mail = '123456@gmail.com';

E_MAIL                                                                                                NAME
----------------------------------------------------------------------------------------------------- ----------
123456@gmail.com                                                                                      Guillermo

SQL> select * from attendances where client = '123456@gmail.com';

CLIENT                                                                                                PERFORMER
----------------------------------------------------------------------------------------------------- ----------
123456@gmail.com                                                                                      Amapola
123456@gmail.com                                                                                      Amapola

SQL> select * from attendances where client = 'cuy@clients.vinylinc.com' and performer = 'Amapola';

CLIENT                                                                                                PERFORMER
----------------------------------------------------------------------------------------------------- ----------
cuy@clients.vinylinc.com                                                                              Amapola
cuy@clients.vinylinc.com                                                                              Amapola

SQL> select * from attendances where client = 'cruz@clients.vinylinc.com' and performer = 'Amapola';

CLIENT                                                                                                PERFORMER
----------------------------------------------------------------------------------------------------- ----------
cruz@clients.vinylinc.com                                                                             Amapola
cruz@clients.vinylinc.com                                                                             Amapola
```

5 Explicitly required Triggers

a) Description of the design:

Trigger1:

On **trigger** 1 we used a structure to avoid mutating tables (with a temporary table and 2 triggers)(for table performances).
The first trigger for the **trigger1** called **update_concert1**, is in charge of detecting an insert on performances. **Before that insert** and **for each row** we check that the new duration is bigger than 0 because it would not make sense for a concert under or equal 0 values, and in that case it would **raise a personal error**. If the duration is valid we would insert it on the temporary table called **tmp_update_concert** where it will store the new duration/60 because we want the duration to be on minutes and not seconds, the new performer, the new when and the new_sequ from the new row inserted at **performances**.
The second trigger for the **trigger1** called **update_concert2** is activated after the insert on performances. This triggers what it does is a loop from where it iterates the temporary table. Inside that loop we store on the variable **aux** the duration of the actual **concert** and then we update on concert the new duration **aux+row.new_dur.** After the update we remove that row from the temporary table because we don't want to remain in it and avoid future errors (e.g if we did another insert it would add the previous value + the new value so that is the motive that we decided to use this approach).
Following the ECA rules, as we have 2 triggers, we will do it separately, for the first trigger, the event is before insert in performances, the condition is that duration is greater than 0 and the action is that if the previous occurs, we insert the new duration in a temporary table to avoid the mutating table error. For the second trigger, the event is after insert on performances, there is not really a condition, and the action is to update the duration of the concert, adding it the duration of a performance.

Trigger2:

The trigger 2 called **less_age** is activated **before an insert** on **attendance** and **for each row**. What this trigger does is stored on the variable **aux** the **birthdate** of the client inserted on **attendance**. Then we make a conditional "**if**" and we check that the client is older than 18 years old from the purchase date of the **attendance** in case is younger we **raise a personalized error** (`-20001,'BAD AGE!'`).
Finally, following the ECA rules, the event is before insert, the condition is whether purchase date - birth date > 18 and finally, the action is raise an error if its under age or insert if he is not.

Trigger3:

The trigger 3 is called reverse_trigger1 activated before insert and reverse_trigger2 activated after insert in **songs**. We did 2 because we believed that, as when using one trigger, we would be using the table we are preventing the insertion, we could have received a mutating table

error, so we just prevented it to happen by using the first general solution provided in the lab classes.
The temporary table that we had to create for the general solution, we called it: tmp_reverse_song, with the same attributes as table song.
What the first trigger does, is to before inserting in songs, **for each row**, we introduce into the temporary table, the values being inserted, but when inserting writer and cowriter, we insert them in the opposite order. The second trigger does a for loop in which it iterates in the temporary table. The trigger2 counts how many songs are there in table songs with title = :NEW.title, writer = row.writer and cowriter = row.cowriter and insert this count into a flag. If the flag is 0, we just do the normal flow of the insert, but if the flag is larger than 0, that means that there is already a song with the two authors reversed, so we raise an exception.
For the first trigger (reverse_trigger1), following the ECA rules, the event is before insert, there is no condition and the action is to insert in the global temporary table tmp_reverse_song.
For the second trigger (reverse_trigger2), following the ECA rules, the event is after insert, the condition is that the flag is larger than 0, and the action is, if the flag is 0, insert, in any other case, raise an exception.

b) Code (PL/SQL)

Trigger1:

```
--Temporary table to avoid mutation errors
Create global temporary TABLE tmp_update_concert
(new_dur NUMBER(4), new_performer VARCHAR2(50),new_when DATE,new_sequ Number(3));

--Trigger to insert on the temporary table
Create or replace Trigger update_concert_1
BEFORE  insert on performances
for each row
DECLARE
bad_dur Exception;
begin
    If :NEW.duration>0 then
    insert into tmp_update_concert
        values(:NEW.duration/60,:NEW.performer,:NEW.when,:NEW.sequ);
    else raise bad_dur;
    end if;
exception
    when bad_dur then raise_application_error(-20001,'WRONG song duration it must be bigger than 0!');
end update_concert_1;
/

--Trigger insert row by row on the destination table
```

```
Create or replace Trigger update_concert_2
AFTER insert on performances
DECLARE
aux Number;
begin
    for row in (select * from tmp_update_concert) loop
        select duration into aux from concerts where performer=row.new_performer and when=row.new_when;
        update concerts set duration=(aux+row.new_dur)
            where performer=row.new_performer and when=row.new_when;
        delete from tmp_update_concert where new_dur=row.new_dur and new_performer=row.new_performer and
new_when=row.new_when and new_sequ=row.new_sequ;
    end loop;
end update_concert_2;
/
```

Trigger2:

```
CREATE OR REPLACE TRIGGER less_age
BEFORE INSERT ON attendances
FOR EACH ROW
DECLARE
badage EXCEPTION;
aux number;
BEGIN
    select to_number(to_char(birthdate,'YY')) into aux from clients where e_mail = :new.client;
    if abs(to_number(to_char(:new.purchase,'YY')) - aux) < 18
        then raise badage;
    end if;
exception
    when badage then raise_application_error(-20001,'BAD AGE!');
end less_age;
/
```

Trigger3:

```
Create global temporary TABLE tmp_reverse_song
(new_title varchar2(50),new_writer varchar(14),new_cowriter varchar(14));


CREATE OR REPLACE TRIGGER reverse_trigger1
BEFORE INSERT ON songs
FOR EACH ROW
BEGIN
```

```
    insert into tmp_reverse_song(new_title,new_writer,new_cowriter) values(:NEW.title,:NEW.cowriter,:NEW.writer);
end reverse_trigger1;
/

CREATE OR REPLACE TRIGGER reverse_trigger2
AFTER INSERT ON songs
DECLARE
badsong EXCEPTION;
flag number :=0;
BEGIN
    for row in (select * from tmp_reverse_song) loop
        select count('x') into flag
        from (select * from songs where title = row.new_title and writer = row.new_writer and cowriter = row.new_cowriter);
        if flag > 0
            then raise badsong;
        end if;
    end loop;
exception
    when badsong then raise_application_error(-20001,'WRONG SONG!');
end reverse_trigger2;
/
```

c) Tests

Trigger1:

For **trigger 1** we decided to test the mutation error with bulk insert and individual insert and how it handles errors.

Test1:Bulk insert

Initial values on concert: performer='Rici', when='10/06/88',..., duration=86
Expected values after the trigger: performer='Rici', when='10/08/88',..., duration=91

```
--create new table
CREATE TABLE bulk_insert(
performer  VARCHAR2(35) not null,
when  DATE not null,
sequ number(3),
songtitle varchar2(100),
songwriter  varchar2(14),
duration number(4),
CONSTRAINT PK_MANAGERS33 PRIMARY KEY(performer,when,sequ)
);
--insert the data
insert into bulk_insert (performer,when,sequ,songtitle,songwriter,duration) values ('Rici','10/06/88',100,'Acabar
bandera','SE>>0866705629',60);
insert into bulk_insert (performer,when,sequ,songtitle,songwriter,duration) values ('Rici','10/06/88',101,'Absurdity
thunder','SE>>0414837052',60);
```

```
insert into bulk_insert (performer,when,sequ,songtitle,songwriter,duration) values ('Rici','10/06/88',102,'Absurdity or
valley','FR>>0512630289',60);
insert into bulk_insert (performer,when,sequ,songtitle,songwriter,duration) values ('Rici','10/06/88',103,'Absurdity
thunder','SE>>0414837052',60);
insert into bulk_insert (performer,when,sequ,songtitle,songwriter,duration) values
('Rici','10/06/88',104,'Acabar','ES>>0971759229',60);
--repopulate to performances
insert into PERFORMANCES (performer,when,sequ,songtitle,songwriter,duration) (select
performer,when,sequ,songtitle,songwriter,duration from bulk_insert);

drop CASCADE table bulk_insert;
```

After executing the code from before we obtain:

```
PERFORMER                                 WHEN     TOUR                                                                MUNICIPALITY
----------------------------------------- -------- ------------------------------------------------------------------- ------------------------------
Rici                                      10/06/88 The '88 genie Tour                                                  Wolvestown on Yangtze

ADDRESS                                                                                                               COUNTRY
----------------------------------------------------------------------------------------------------------------------- ----------
182 Graveyard Square                                                                                                  Denmark

 ATTENDANCE   DURATION    MANAGER
 ---------- ---------- ----------
          0         91  555097557
```

Test2:Individual positive number insert

Initial values on concert: performer='Rici', when='10/06/88',..., duration=86
Expected values after the trigger: performer='Rici', when='10/08/88',..., duration=91

```
insert into performances (performer,when,sequ,songtitle,songwriter,duration) values ('Rici','10/06/88',100,'Acabar
bandera','SE>>0866705629',60);
insert into performances (performer,when,sequ,songtitle,songwriter,duration) values ('Rici','10/06/88',101,'Absurdity
thunder','SE>>0414837052',60);
insert into performances (performer,when,sequ,songtitle,songwriter,duration) values ('Rici','10/06/88',102,'Absurdity or
valley','FR>>0512630289',60);
insert into performances (performer,when,sequ,songtitle,songwriter,duration) values ('Rici','10/06/88',103,'Absurdity
thunder','SE>>0414837052',60);
insert into performances (performer,when,sequ,songtitle,songwriter,duration) values
('Rici','10/06/88',104,'Acabar','ES>>0971759229',60);
```

Before:

```
PERFORMER                                          WHEN       DURATION
-------------------------------------------------- -------- ----------
Rici                                               10/06/88         86
```

After:

```
PERFORMER                                          WHEN       DURATION
-------------------------------------------------- -------- ----------
Rici                                               10/06/88         91
```

Test3:Individual negative and null number insert

Initial values on concert: performer='Rici', when='10/06/88',..., duration=86
Expected values after the trigger: performer='Rici', when='10/08/88',..., duration=89 and 2 errors giving the following error (`-20001,'WRONG song duration it must be bigger than 0!'`)

```
insert into performances (performer,when,sequ,songtitle,songwriter,duration) values ('Rici','10/06/88',100,'Acabar bandera','SE>>0866705629',-60);
insert into performances (performer,when,sequ,songtitle,songwriter,duration) values ('Rici','10/06/88',101,'Absurdity thunder','SE>>0414837052',NULL);
insert into performances (performer,when,sequ,songtitle,songwriter,duration) values ('Rici','10/06/88',102,'Absurdity or valley','FR>>0512630289',60);
insert into performances (performer,when,sequ,songtitle,songwriter,duration) values ('Rici','10/06/88',103,'Absurdity thunder','SE>>0414837052',60);
insert into performances (performer,when,sequ,songtitle,songwriter,duration) values ('Rici','10/06/88',104,'Acabar','ES>>0971759229',60);
```

Before:

```
PERFORMER                                          WHEN       DURATION
-------------------------------------------------- -------- ----------
Rici                                               10/06/88         86
```

After:

```
SQL> insert into performances (performer,when,sequ,songtitle,songwriter,duration) values ('Rici','10/06/88',100,'Acabar bandera','SE>>0866705629',-60);
insert into performances (performer,when,sequ,songtitle,songwriter,duration) values ('Rici','10/06/88',100,'Acabar bandera','SE>>0866705629',-60)
            *
ERROR en linea 1:
ORA-20001: WRONG song duration it must be bigger than 0!
ORA-06512: en "FSDB228.UPDATE_CONCERT_1", linea 10
ORA-04088: error durante la ejecucion del disparador 'FSDB228.UPDATE_CONCERT_1'


SQL> insert into performances (performer,when,sequ,songtitle,songwriter,duration) values ('Rici','10/06/88',101,'Absurdity thunder','SE>>0414837052',NULL);
insert into performances (performer,when,sequ,songtitle,songwriter,duration) values ('Rici','10/06/88',101,'Absurdity thunder','SE>>0414837052',NULL)
            *
ERROR en linea 1:
ORA-20001: WRONG song duration it must be bigger than 0!
ORA-06512: en "FSDB228.UPDATE_CONCERT_1", linea 10
ORA-04088: error durante la ejecucion del disparador 'FSDB228.UPDATE_CONCERT_1'
```

```
PERFORMER                                          WHEN       DURATION
-------------------------------------------------- -------- ----------
Rici                                               10/06/88         89
```

Trigger2:

For trigger 2, we will create some clients and then insert them at the same time in attendances. The insert will be of underage clients, so that the error we prepared arises.
To create the clients we did:

```
insert into clients(e_mail,birthdate) values('123456@gmail.com','12/06/10');
insert into clients(e_mail,birthdate) values('654321@gmail.com','13/06/10');
insert into clients(e_mail,birthdate) values('333333@gmail.com','11/06/10');
```

Then the insert of the clients in the attendance table, to purchase a ticket is:

```
insert into attendances(client,performer,when,rfid,purchase)
values('123456@gmail.com','Amapola','07/09/18','BHAZJA7JM57PVS9LED2IKHFBIQQY8CYSBKHLY6B2X474ZCAQUUJZFZ0U80XRM6BY','14/06/15');
insert into attendances(client,performer,when,rfid,purchase)
values('654321@gmail.com','Amapola','07/09/18','BHAZJA7JM57PVS9LED2IKHFBIQQY8CYSBKHLY6B2X474ZCAQUUJZFZ0U80XRM6BZ','15/06/15');
insert into attendances(client,performer,when,rfid,purchase)
values('333333@gmail.com','Amapola','07/09/18','BHAZJA7JM57PVS9LED2IKHFBIQQY8CYSBKHLY6B2X474ZCAQUUJZFZ0U80XRM6BX','13/06/15');
```

The result of executing this in oracle should give an error as the purchase dates - the birth dates, do not give 18, so the error that we prepared (BAD AGE!) should arise:

```
SQL>
SQL> insert into attendances(client,performer,when,rfid,purchase) values('123456@gmail.com','Amapola','07/09/18','BHAZJA7JM57PVS9LED2IKHFBIQQY8CYSBKHLY6B2X474ZCAQUUJZFZ0U80XRM6BY','14/06/1
insert into attendances(client,performer,when,rfid,purchase) values('123456@gmail.com','Amapola','07/09/18','BHAZJA7JM57PVS9LED2IKHFBIQQY8CYSBKHLY6B2X474ZCAQUUJZFZ0U80XRM6BY','14/06/15')
            *
ERROR en linea 1:
ORA-20001: BAD AGE!
ORA-06512: en "FSDB228.LESS_AGE", linea 10
ORA-04088: error durante la ejecucion del disparador 'FSDB228.LESS_AGE'


SQL> insert into attendances(client,performer,when,rfid,purchase) values('654321@gmail.com','Amapola','07/09/18','BHAZJA7JM57PVS9LED2IKHFBIQQY8CYSBKHLY6B2X474ZCAQUUJZFZ0U80XRM6BZ','15/06/1
insert into attendances(client,performer,when,rfid,purchase) values('654321@gmail.com','Amapola','07/09/18','BHAZJA7JM57PVS9LED2IKHFBIQQY8CYSBKHLY6B2X474ZCAQUUJZFZ0U80XRM6BZ','15/06/15')
            *
ERROR en linea 1:
ORA-20001: BAD AGE!
ORA-06512: en "FSDB228.LESS_AGE", linea 10
ORA-04088: error durante la ejecucion del disparador 'FSDB228.LESS_AGE'


SQL> insert into attendances(client,performer,when,rfid,purchase) values('333333@gmail.com','Amapola','07/09/18','BHAZJA7JM57PVS9LED2IKHFBIQQY8CYSBKHLY6B2X474ZCAQUUJZFZ0U80XRM6BX','13/06/1
insert into attendances(client,performer,when,rfid,purchase) values('333333@gmail.com','Amapola','07/09/18','BHAZJA7JM57PVS9LED2IKHFBIQQY8CYSBKHLY6B2X474ZCAQUUJZFZ0U80XRM6BX','13/06/15')
            *
ERROR en linea 1:
ORA-20001: BAD AGE!
ORA-06512: en "FSDB228.LESS_AGE", linea 10
ORA-04088: error durante la ejecucion del disparador 'FSDB228.LESS_AGE'
```

which is the exception we prepared for that type of error.
Finally we test it with a bulk insert:

```
--To prepare:
insert into clients(e_mail,birthdate) values('123456@gmail.com','12/06/10');
insert into clients(e_mail,birthdate) values('654321@gmail.com','13/06/10');
insert into clients(e_mail,birthdate) values('333333@gmail.com','11/06/10');
--test trigger 2
create table test_table (
    client VARCHAR2(100),
    performer VARCHAR2(50),
    when DATE,
    rfid VARCHAR2(120),
    purchase DATE);
--insert rows on the test table
insert into test_table(client,performer,when,rfid,purchase)
values('123456@gmail.com','Amapola','07/09/18','BHAZJA7JM57PVS9LED2IKHFBIQQY8CYSBKHLY6B2X474ZCAQUUJ
ZFZ0U80XRM6BY','14/06/15');
insert into test_table(client,performer,when,rfid,purchase)
values('654321@gmail.com','Amapola','07/09/18','BHAZJA7JM57PVS9LED2IKHFBIQQY8CYSBKHLY6B2X474ZCAQUUJ
```

```
ZFZ0U80XRM6BZ','15/06/15');
insert into test_table(client,performer,when,rfid,purchase)
values('333333@gmail.com','Amapola','07/09/18','BHAZJA7JM57PVS9LED2IKHFBIQQY8CYSBKHLY6B2X474ZCAQUUJ
ZFZ0U80XRM6BX','13/06/15');
--bulk insert
insert into ATTENDANCES(client,performer,when,rfid,purchase) (select * from test_table);
--drop test table
drop table test_table;
```

```
SQL> insert into ATTENDANCES(client,performer,when,rfid,purchase) (select * from test_table);
insert into ATTENDANCES(client,performer,when,rfid,purchase) (select * from test_table)
            *
ERROR en linea 1:
ORA-20001: BAD AGE!
ORA-06512: en "FSDB228.LESS_AGE", linea 10
ORA-04088: error durante la ejecucion del disparador 'FSDB228.LESS_AGE'
```

The error we wanted appears.
Finally, we insert a client whose age is larger than 18

```
insert into clients(e_mail,birthdate) values('123456@gmail.com','12/06/80');
insert into attendances(client,performer,when,rfid,purchase)
values('123456@gmail.com','Amapola','07/09/18','BHAZJA7JM57PVS9LED2IKHFBIQQY8CYSBKHLY6B2X474ZCAQUUJ
ZFZ0U80XRM6BY','14/06/15');
```

No problem arises:

```
SQL> insert into clients(e_mail,birthdate) values('123456@gmail.com','12/06/80');
1 fila creada.
SQL> insert into attendances(client,performer,when,rfid,purchase) values('123456@gmail.com','Amapola','07/09/18','BHAZJA7JM57PVS9LED2IKHFBIQQY8CYSBKHLY6B2X474ZCAQUUJZFZ0U80XRM6BY','14/06/15
1 fila creada.
```

Trigger3:

For trigger 3, we will insert several songs at the same time with a title that already exists, but with its two authors reversed. When inverting writer and cowriter, the result should be the error we prepared for this case: (WRONG SONG!)

```
insert into songs(title,writer,cowriter) values('Absence and houses','ES>>0287689961','ES>>0522298796');
insert into songs(title,writer,cowriter) values('Ad and sad willows','IT>>0852065639','IT>>0211101352');
insert into songs(title,writer,cowriter) values('Ad and secret','US>>0546211813','US>>0037011803');
insert into songs(title,writer,cowriter) values('Ad and wine','GB>>0596598495','GB>>0600247598');
```

```
SQL>  insert into songs(title,writer,cowriter) values('Absence and houses','ES>>0287689961','ES>>0522298796');
 insert into songs(title,writer,cowriter) values('Absence and houses','ES>>0287689961','ES>>0522298796')
            *
ERROR en linea 1:
ORA-20001: WRONG SONG!
ORA-06512: en "FSDB228.REVERSE_TRIGGER2", linea 13
ORA-04088: error durante la ejecucion del disparador 'FSDB228.REVERSE_TRIGGER2'


SQL> insert into songs(title,writer,cowriter) values('Ad and sad willows','IT>>0852065639','IT>>0211101352');
insert into songs(title,writer,cowriter) values('Ad and sad willows','IT>>0852065639','IT>>0211101352')
            *
ERROR en linea 1:
ORA-20001: WRONG SONG!
ORA-06512: en "FSDB228.REVERSE_TRIGGER2", linea 13
ORA-04088: error durante la ejecucion del disparador 'FSDB228.REVERSE_TRIGGER2'


SQL> insert into songs(title,writer,cowriter) values('Ad and secret','US>>0546211813','US>>0037011803');
insert into songs(title,writer,cowriter) values('Ad and secret','US>>0546211813','US>>0037011803')
            *
ERROR en linea 1:
ORA-20001: WRONG SONG!
ORA-06512: en "FSDB228.REVERSE_TRIGGER2", linea 13
ORA-04088: error durante la ejecucion del disparador 'FSDB228.REVERSE_TRIGGER2'


SQL> insert into songs(title,writer,cowriter) values('Ad and wine','GB>>0596598495','GB>>0600247598');
insert into songs(title,writer,cowriter) values('Ad and wine','GB>>0596598495','GB>>0600247598')
            *
ERROR en linea 1:
ORA-20001: WRONG SONG!
ORA-06512: en "FSDB228.REVERSE_TRIGGER2", linea 13
ORA-04088: error durante la ejecucion del disparador 'FSDB228.REVERSE_TRIGGER2'
```

Finally, we have also included a bulk insert with some songs with its authors reversed:

```
--test trigger3
--create test table
create table test_table (
    title VARCHAR2(50),
    writer VARCHAR2(14),
    cowriter VARCHAR2(24));
--insert rows on the test_table
insert into test_table(title,writer,cowriter) values('Absence and
houses','ES>>0287689961','ES>>0522298796');
insert into test_table(title,writer,cowriter) values('Ad and sad
willows','IT>>0852065639','IT>>0211101352');
insert into test_table(title,writer,cowriter) values('Ad and
secret','US>>0546211813','US>>0037011803');
insert into test_table(title,writer,cowriter) values('Ad and
wine','GB>>0596598495','GB>>0600247598');
--bulk insert into songs
insert into songs(title,writer,cowriter) (select * from test_table);
--drop test table
drop table test_table;
```

```
SQL> insert into songs(title,writer,cowriter) (select * from test_table);
insert into songs(title,writer,cowriter) (select * from test_table)
            *
ERROR en linea 1:
ORA-20001: WRONG SONG!
ORA-06512: en "FSDB228.REVERSE_TRIGGER2", linea 13
ORA-04088: error durante la ejecucion del disparador 'FSDB228.REVERSE_TRIGGER2'
```

6 Concluding Remarks

To start with, we noticed that at the end of this project, we had a much greater knowledge of queries than before. We are much faster when writing them and we really understand what we are doing and it's not just to try and see if it works or not. We have to say that along the whole project, all queries have been done with NATURAL JOINS as we find them one of the easiest way of retrieving information from 2 different tables using a key.

To defend our result, we really think that along the whole project, we have written an understandable and efficient code, that accomplishes the function asked in the statement. The second query may take a little too long (about 8 seconds) but the first one is much shorter. As commented in the semantics of query 2, we have selected the best 10 percentages under 100 percent for being more clear. For the second exercise, just by watching the videos we were able to perform the 3 exercises, as they are mainly programming. We have to say that the code of the report might be a little long, but to have a readable report, we decided to sacrifice a little bit of space to have a better report which is more readable. We have also added an auxiliary function to return the name of the current performer outside the package. If our understanding of the statement was correct, we really think that the package also accomplishes what is asked in exercise2.

Additionally, in the third exercise, the one about views, the first 2 views were "easy" and as before, if we understood what the statement said, the view1 and view2 should provide the desired information. However, in view3, we spent much more time, not on the query which was not too difficult but on the insert trigger. The most difficult part was to identify the different cases, 1 new client, an already existing client with no attendances to current performer, an already existing client with only one attendance to a current performer and the banned fan.

Finally, in the triggers section, we were somehow conservative and applied the general solution of the mutating table error to triggers 1 and 3, which were the ones that we thought that could give us the error. Trigger 2 was "save" because inside, you use a table different from the one we are preventing the insert. We checked with several tests how these triggers work and looks fine for us.

In terms of time, we would say that we have spent around 2 weeks with a pace of 3-5 hours per day.

In terms of how much we have achieved thanks to this project, we agree that our understanding of sql code has increased a lot, but as we said before, the concept that we have developed the most is the creation of queries. We now feel much more confident and comfortable when we have to create a difficult query. We have also discovered how to program in sql and how to use triggers and why they are useful. Moreover, we have fully understood what a view is and how to deal with it because it is actually not possible to insert or delete only in the view, so we learned different methods with which we can erase from a view and not from a table which is basically a trigger "instead of".

Finally, we would like to make a suggestion. We think that there should not be exercises that depend on other exercises. In this case, exercise 3 depends on exercise 2. We say that because we did the views before exercise 2, so when we received an email from a professor where our 'current performer' was something different to what we thought, we had to change it. Apart from that point, which we may be wrong, we think that everything else was very accurate.