



BS DEGREE IN INFORMATICS ENGINEERING Academic year: 2022/2023 - 2 <sup>nd</sup> year, 2 <sup>nd</sup> term Subject: File Structures and Databases Second Assignment's Report: DB development and Querying	 
--	---

<b>Lecturer:</b>	<b>FRANCISCO JAVIER CALLE GÓMEZ</b>		
<b>Group:</b>	<b>265</b>	<b>Lab User</b>	<b>fsdb265</b>
<b>Student:</b>	<b>MARÍA ISABEL HERNÁNDEZ BARRIO</b>	<b>NIA:</b>	<b>100472315</b>
<b>Student:</b>	<b>VÍCTOR CARNICERO PRÍNCIPE</b>	<b>NIA:</b>	<b>100472280</b>
<b>Student:</b>	<b>DAVID SÁNCHEZ ARRANZ</b>	<b>NIA:</b>	<b>100460314</b>

## 1. INTRODUCTION

The aim of this report is to collect all the implementation and explanation of our database queries, blocks, views and triggers along with their definition and a series of tests to check its correct functioning. The document structure is specified below in the index.

It is a first introduction into managing more dynamic parts of database management and getting familiar with the processes to correctly manage data, restrict data from users and make their interactions with the database easier and more efficient, also preventing some actions from being performed.

We dive into functions that return values and procedures that do complex inserts in the tables of the database. We explore the types of users and set specific views for them, also identifying which actions they are performing and registering them in order to control this user's interactions with the database in the future.

Along this document we present the design, code in sql and different tests we have used to make sure each part was correctly developed.

# INDEX

<b>1. INTRODUCTION</b>	<b>1</b>
<b>2. QUERIES</b>	<b>3</b>
First Query:	3
MAIN QUERIES:	3
RELATIONAL ALGEBRA:	4
CODE:	4
TESTS:	5
Second Query:	6
MAIN QUERIES:	6
RELATIONAL ALGEBRA:	7
CODE:	7
TESTS:	8
<b>3. PACKAGE</b>	<b>10</b>
A. Design	10
B. Implementation	12
C. Test	17
<b>4. EXTERNAL DESIGN (views)</b>	<b>22</b>
First View:	22
Relational Algebra Design	22
SQL Implementation	22
TEST	22
Second View:	25
Relational Algebra Design	25
SQL Implementation	25
TEST	26
Third View:	28
Relational Algebra Design	28
SQL Implementation	28
TEST	28
<b>5. EXPLICITLY REQUIRED TRIGGERS</b>	<b>30</b>
First Trigger:	30
CODE:	30
TESTS:	30
Second Trigger:	31
CODE:	31
TESTS:	31
Third Trigger:	32
CODE:	32
TESTS:	33
<b>6. CONCLUDING REMARKS</b>	<b>35</b>

## 2. QUERIES

### First Query:

The first query consists on obtaining the percentage of songs recorded and performed by a band that are their own. For this we separate them into different named subqueries for understanding and management purposes.

It is important to note that we count the percentage of **INDIVIDUAL** songs, if the performer has performed/recorded a song more than once we don't take it into account. This was not explicitly stated but we think it is the solution that makes more sense for us.

### MAIN QUERIES:

- **OWN\_PERFORMED:** Queries the number of own songs performed by each performer.
- **TOTAL\_PERFORMED:** Queries the total number of songs performed by each performer.
- **PERCENT\_PERFORMED:** Divides the number obtained in *OWN\_PERFORMED* by the one obtained in *TOTAL\_PERFORMED*.
- **OWN\_RECORDED:** Queries the number of own songs recorded by each performer.
- **TOTAL\_RECORDED:** Queries the total number of songs recorded by each performer.
- **PERCENT\_RECORDED:** Divides the number obtained in *OWN\_RECORDED* by the one obtained in *TOTAL\_RECORDED*
- **FINAL\_QUERY:** Join by performer of *PERCENT\_PERFORMED* and *PERCENT\_RECORDED*

## RELATIONAL ALGEBRA:

$$\begin{aligned}
 \text{own\_performed} &\equiv \pi_{\text{count('x') as times, performer}} \left( \left( \pi_{\text{b.performer, a.title, a.writer}} \left( \left( \pi_{\text{title, writer, pair}} (\text{tracks}) \right) \times_{\text{pair}} \left( \pi_{\text{album}} (\text{albums}) \right) \right) \times_{\text{musician, as writer}} \left( \pi_{\text{musician, band}} (\text{involvement}) \right) \right) \right) \\
 \text{total\_performed} &\equiv \pi_{\text{count('x') as times, performer}} \left( \sigma_{\text{performer}} \left( \pi_{\text{performer, songtitle, songwriter}} (\text{performances}) \right) \right) \\
 \text{percent\_performed} &\equiv \pi_{\text{performer, round} \left( \frac{\text{own\_performed.times}}{\text{total\_performed.times}} \right) * 100 \text{ as performed}} \left( \text{total\_performed} \times_{\text{performer}} \text{own\_performed} \right) \\
 \text{total\_recorded} &\equiv \pi_{\text{count('x') as times, performer}} \left( \sigma_{\text{performer}} \left( \pi_{\text{performer, a.title, a.writer}} \left( \left( \pi_{\text{title, writer, pair}} (\text{tracks}) \right) \times_{\text{pair}} \left( \pi_{\text{album}} (\text{albums}) \right) \right) \right) \right) \\
 \text{percent\_recorded} &\equiv \pi_{\text{performer, round} \left( \frac{\text{own\_recorded.times}}{\text{total\_recorded.times}} \right) * 100 \text{ as recorded}} \left( \text{total\_recorded} \times_{\text{performer}} \text{own\_recorded} \right) \\
 \text{Query 1: } &\text{percent\_performed} \times_{\text{performer}} \text{percent\_recorded}
 \end{aligned}$$

## CODE:

WITH own\_performed AS (SELECT count('x') times, performer FROM (SELECT distinct songtitle, songwriter, performer from PERFORMANCES)

JOIN INVOLVEMENT

ON (songwriter = musician and performer = band)

GROUP BY performer),

total\_performed AS (SELECT count('x') times, performer FROM (SELECT distinct performer, songtitle, songwriter from PERFORMANCES)

GROUP BY performer),

percent\_performed AS (SELECT performer,

round(own\_performed.times/total\_performed.times, 4)\*100 performed FROM total\_performed JOIN own\_performed USING (performer)),

own\_recorded AS (SELECT count('x') times, performer FROM (SELECT distinct b.performer, a.title, a.writer FROM ((SELECT distinct title, writer, pair FROM TRACKS) A

JOIN ALBUMS B USING (pair))

JOIN (SELECT distinct musician, band FROM INVOLVEMENT) C ON (musician=writer)) GROUP BY performer),

```
total_recorded AS (SELECT count('x') times, performer FROM (SELECT distinct performer,
a.title, a.writer FROM ((SELECT distinct title, writer, pair FROM TRACKS) A
JOIN ALBUMS B USING (pair))) GROUP BY performer),
```

```
percent_recorded AS (SELECT performer, round(own_recorded.times/total_recorded.times,
4)*100 recorded FROM total_recorded
JOIN own_recorded USING (performer))
```

```
SELECT * FROM percent_performed JOIN percent_recorded USING (performer);
```

## TESTS:

First we show some of the results from the query:

PERFORMER	PERFORMED	RECORDED
Piedras del Camino	47,8	47,04
Maria de la Victoria Sanchez	16,09	18,24
Alfredo Andres Cabrera	19,72	20
Rici	20,33	19,74
Amapola	43,9	44,85
David Perez	19,08	19,16
Ambico Pomez	20,98	20,98
Eva Infante	23,08	16,87
Penalosa	23,3	21,99
Sidonio Bravo	14,71	17,41
Los Caseros	43,09	42,15

Taking the first performer “Piedras del camino”, we identify a person from the band and eliminate all the songs that do not belong to this person:

```
delete from performances where performer='Piedras del Camino' and songwriter !=
'GB>>0774702141';
```

Then we execute the query again:

PERFORMER	PERFORMED	RECORDED
1 Piedras del Camino	100	47,04
2 Maria de la Victoria Sanchez	16,09	18,24
3 Alfredo Andres Cabrera	19,72	20
4 Rici	20,33	19,74
5 Amapola	43,9	44,85
6 David Perez	19,08	19,16
7 Ambico Pomez	20,98	20,98
8 Penalosa	23,3	21,99
9 Sidonio Bravo	14,71	17,41

As we expected the percentage of performed songs for “Piedras del Camino” is now 100% and the number of recorded songs has not changed.

## Second Query:

The second query consists on identifying the top ten performers with the highest percentage of songs recorded and performed in a concert. For these we have to analyze the average age of the songs performed and display it all together.

It is important to note that we count the percentage of INDIVIDUAL songs, if the performer has performed/recorded a song more than once we don't take it into account when calculating the percentage. But, when calculating the average age of the songs, we take into account all the performances, even of the same song, as it would be difficult to establish a rule to filter which performance should count to calculate the age. This was not explicitly stated but we think it is the solution that makes more sense for us.

## MAIN QUERIES:

- TOTAL\_RECORDED:** Queries all songs recorded by each performer.
- TOTAL\_PERFORMED:** Queries all songs performed by each performer.
- TOTAL\_NUMBER\_RECORDED:** Counts the total number of songs recorded by each performer.
- TOTAL\_NUMBER\_BOTH:** Counts the total number of songs recorded and performed by each performer.
- BEST\_10\_BANDS:** Using *TOTAL\_NUMBER\_RECORDED* and *TOTAL\_NUMBER\_BOTH*, obtains the 10 bands with the highest percentage of recorded and performed songs.
- FINAL\_QUERY:** Obtains all the average ages for the songs of these bands and displays it along with the percentage.

## RELATIONAL ALGEBRA:

$$\begin{aligned}
 \text{total\_recorded} &= \pi_{\text{performer, a.title, a.writer}} \left( \rho_a \left( \pi_{\text{title, writer, pair, rec\_date}} (\text{tracks}) \right) * \rho_b (\text{albums}) \right) \\
 \text{total\_performed} &= \pi_{\text{performer, songtitle as title, songwriter as writer}} (\text{performances}) \\
 \text{best\_10\_bands} &= \sigma_{\text{ranknum} \leq 10} \pi_{\text{performer}} \left( \text{round} \left( \frac{\text{times\_both}}{\text{times\_recorded}} \right) - 10.0 \right) \text{ as recorded-performances} \\
 &\quad \text{as recorded-performances} \\
 \text{total\_number\_recorded} &= \pi_{\text{count('x') as times\_recorded, performer}} \left( \sigma_{\text{total\_recorded}} \right) \\
 \text{total\_number\_both} &= \pi_{\text{count('x') as times\_both, performer}} \left( \sigma_{\text{total\_recorded} \theta \text{ total\_performed}} \right) \\
 &\quad \text{performer, title, writer} \\
 \text{Query 2:} \\
 &\pi_{\text{performer, avg(per\_date - rec\_date) as average\_days, recorded-performances, performer}} \left( \sigma_{\text{best\_10\_bands} \theta \left( \pi_{\text{performances}} \right) \theta \left( \rho_a \left( \pi_{\text{title, writer, pair, rec\_date}} (\text{tracks}) \right) \theta \rho_b (\text{albums}) \right) \right) \theta \text{ best\_10\_bands} \\
 &\quad \text{performer} \\
 &\quad \left. \begin{aligned} &\text{average\_days/365 "y"} \\ &\text{mod(average\_days, 768.12) "m"} \\ &\text{mid(mod(average\_days, 768.12), 12) "d"} \end{aligned} \right\} \text{ as avg\_days}
 \end{aligned}$$

## CODE:



```
WITH total_recorded AS (SELECT distinct performer, a.title, a.writer FROM((SELECT
distinct title, writer, pair, rec_date FROM TRACKS) A
JOIN ALBUMS B USING (pair))),
```

```
total_performed AS (SELECT distinct performer, songtitle title, songwriter writer from
PERFORMANCES),
```

```
total_number_recorded as (SELECT count('x') times_recorded, performer FROM
total_recorded GROUP BY performer),
```

```
total_number_both as (SELECT count('x') times_both, performer FROM (SELECT distinct
performer, title, writer FROM total_recorded
INNER JOIN total_performed USING (title, writer, performer))
GROUP BY performer),
```



<p>BS DEGREE IN INFORMATICS ENGINEERING</p> <p>Academic year: 2022/2023 - 2<sup>nd</sup> year, 2<sup>nd</sup> term</p> <p>Subject: File Structures and Databases</p> <p>Second Assignment's Report: DB development and Querying</p>	 
---	---

```
best_10_bands AS (SELECT * FROM(SELECT performer, round(times_both/times_recorded,
4)*100 recorded_performances FROM total_number_recorded INNER JOIN total_number_both
USING(performer) ORDER BY recorded_performances DESC) WHERE rownum <= 10)
```

```
SELECT performer, recorded_performances percent_recorded_performances,
trunc(average_days/365)||'y-'||
trunc(mod(average_days,365)/12)||'m-'||
trunc(mod(mod(average_days,365),12))||'d' avg_age
FROM((SELECT round(avg(per_date-rec_date),4) average_days, performer
FROM(best_10_bands
JOIN (SELECT distinct performer, songtitle title, songwriter
writer, when per_date from PERFORMANCES) USING (performer)
JOIN (SELECT distinct performer, a.title, a.writer,
a.rec_date FROM((SELECT distinct title, writer, pair, rec_date FROM TRACKS) A
JOIN ALBUMS B USING (pair))) USING (title,
writer,performer)) GROUP BY (PERFORMER))
JOIN best_10_bands USING (performer));
```

## TESTS:

To start we present the results of the second query:

PERFORMER	PERCENT_RECORDED_PERFORMANCES	AVG_AGE
Joaqui	70	1y-17m-0d
Sofia Jimenez	67,86	1y-14m-8d
Kohatsu	66,67	2y-5m-3d
Sisebuto Sanchez	64,29	1y-29m-2d
Vigilia	64,29	2y-12m-10d
Canta Mañanitas	64,29	2y-0m-7d
Francy	61,9	2y-10m-2d
Felisa Ruiz	60,71	2y-14m-3d
Rosa Fedora Valleriestra	60,47	2y-29m-6d
Zapata Garcia	60,34	2y-23m-8d

Now, taking the maximum performer “Joaqui”, we execute the following command to see the changes in the query:



```
delete from performances where performer='Joaqui' and rownum < 5;
```

PERFORMER	PERCENT_RECORDED_PERFORMANCES	AVG_AGE
1 Sofia Jimenez	67,86	1y-14m-8d
2 Joaqui	66,67	1y-17m-5d
3 Kohatsu	66,67	2y-5m-3d
4 Sisebuto Sanchez	64,29	1y-29m-2d
5 Vigilia	64,29	2y-12m-10d
6 Canta Mañanitas	64,29	2y-0m-7d
7 Francy	61,9	2y-10m-2d
8 Felisa Ruiz	60,71	2y-14m-3d
9 Rosa Fedora Valleriestra	60,47	2y-29m-6d
10 Zapata Garcia	60,34	2y-23m-8d

We see that we deleted just enough performances for “Joaqui” to come down to the second position, we also see how the average age has changed. It is also important to note that only the results for “Joaqui” change and all of the others stay the same.

This convinces us of the correctness of our result.

### 3. PACKAGE

#### A. Design

We create a package with the variable `current_performer` and several procedures:

- **AssignPerformer:** takes the name of a performer and assigns it to the package variable `current_performer`. For this to work we needed to implement a function so that that variable using the label “`current_performer`” can be accessed. That function is the following:

```
FUNCTION get_performer
RETURN varchar2
IS
BEGIN
RETURN current_performer;
END;
```

- **InsertAlbum:** this function takes the attributes of a track and an album, and inserts the new album and the new track. If the album already existed, that is, the number of albums with the same PAIR as the one given by parameters is greater than zero, the procedure just inserts the track into that existing album.

This procedure also takes into account the necessity of changing the “sequ” number of a track when inserting it into an album. If the album is empty, the sequ of the new track will be 1, otherwise the sequ will be the following corresponding one (that is, the greater sequ in the album + 1).

#### Implicit semantic assumptions and comments:

- We assume the track is added in the last position.
- We take into account that there may be albums without tracks.
- We only require values for the album parameters if the album does not exist, else we set them to NULL, but it really does not matter what value they have, since we only take into consideration that the primary key is correct to insert the tracks.

- **DeleteTrack:** this procedure takes the PAIR of an album and the Title of a track, and deletes that track from the album. If after deleting it the album is empty (the auxiliary variable “aux”, storing the number of tracks with the PAIR of the album, is 0) then the album is also deleted.

- **PerformerReport:** this procedure uses several implicit cursors to calculate some data about the `current_performer`. Some FOR loops are used to access the cursors’ data.

- **Number of albums of each type:** we count the number of albums whose performer is the `current_performer` and group them by format. In order to make the report more readable, we used a CASE statement to replace the format notation ('V', 'C', and so on) by their actual names ('vinyl', 'CD', etc).

- **Average number of songs of each type:** to construct the cursor needed, we group the tracks by pair, getting the number of songs on each album and then join this selection with the pairs and formats of all the albums of the `current_performer` grouping both tables by format.

- **Average album duration (by types):** here we calculate the average total duration in minutes (sum of durations of all the tracks of an album) of an album, distinguishing by the format of the albums.

We select the pair and format of all albums of the `current_performer` and left join it with a grouping (by album pair) of the tracks getting the sum of the durations of the songs of each album.

Naturally, the left join is done using the pair of each album and all that selection is finally grouped by format to calculate the average duration for each format.

- **Periodicity:** here we calculate the frequency with which the performer airs an album. For that aim, we get the time elapsed between the release of the first and last album and we divide that subtraction by the total number of albums released by the artist.

- **Publishers Report:** here we group the albums of our performer by the publishers, and the count gives us the number of albums in which each publisher has collaborated with `current_performer`. We use the variable `total_albums` to store the total number of albums that our performer has released, which is needed to get the demanded percentage.

- **Studios Report:** we get the number of tracks each studio has worked on, using an IN statement to make sure those tracks belong to the `current_performer`. We also calculate the total number of tracks belonging to our performer using the same kind of statement and store the value in the variable `total_tracks`.

Again, we use this to calculate the percentage of studio tracks with respect to the total performer tracks. Notice that the primary key of Studio is already the name of the studio, so we do not need to join the selection with another query in order to get the studios' names.

- **Engineers Report:** the functioning of this block is the same as the previous one, but using engineers instead of studio.

- **Managers Report:** here, we select the name and number of albums done in collaboration with each performer from a left join between a table of managers plus albums done by each of them (so

obtained from albums and grouped by manager) and the managers table (so that we can get the name of each manager instead of their phone numbers).

On the other hand, we do a similar select but this time looking for the number of concerts, not albums. When we have both selects done, we full join them and set to 0 the null values obtained, as in some cases there are managers that collaborated in many albums but in zero concerts and the other way around.

## B. Implementation

```
CREATE OR REPLACE PACKAGE melopack AS
    current_performer varchar2(50);
    PROCEDURE AssignPerformer(performer varchar2);
    PROCEDURE InsertAlbum(AlbumPair varchar2, TrackTitle varchar2, TrackWriter char,
        TrackDuration number, TrackRecDate date, TrackStudio varchar, TrackEngineer
        varchar2,
        AlbumFormat char default NULL, AlbumTitle varchar2 default NULL, AlbumRelDate date
        default NULL,
        AlbumPublisher varchar2 default NULL, AlbumManager number default NULL);
    FUNCTION get_performer RETURN varchar2;
    PROCEDURE DeleteTrack (AlbumPair char, TrackTitle varchar2);
    PROCEDURE PerformerReport;
END melopack;
/
CREATE OR REPLACE PACKAGE BODY melopack AS
    PROCEDURE AssignPerformer ( performer varchar2) IS
        BEGIN
            current_performer := performer;
        END AssignPerformer;

    PROCEDURE InsertAlbum(AlbumPair varchar2, TrackTitle varchar2, TrackWriter char,
        TrackDuration number, TrackRecDate date, TrackStudio varchar,
```

```
TrackEngineer varchar2, AlbumFormat char default NULL, AlbumTitle varchar2 default  
NULL, AlbumRelDate date default NULL,
```

```
AlbumPublisher varchar2 default NULL, AlbumManager number default NULL)
```

```
IS
```

```
aux number;
```

```
sequ number;
```

```
BEGIN
```

```
    SELECT COUNT(*) INTO aux FROM albums WHERE PAIR = AlbumPair;
```

```
    IF aux=0 THEN
```

```
        INSERT INTO ALBUMS(PAIR,performer,format,title,rel_date,publisher,manager)
```

```
        VALUES(AlbumPair, current_performer, AlbumFormat, AlbumTitle, AlbumRelDate,  
AlbumPublisher, AlbumManager);
```

```
        sequ := 1;
```

```
    ELSE
```

```
        SELECT COUNT(*) INTO aux FROM tracks WHERE PAIR = AlbumPair;
```

```
        IF aux=0 THEN
```

```
            sequ := 1;
```

```
        ELSE select max(sequ)+1 into sequ from TRACKS where pair = AlbumPair group by  
pair;
```

```
        END IF;
```

```
    END IF;
```

```
    INSERT INTO TRACKS(PAIR, sequ,title,writer, duration,rec_date,studio,engineer)
```

```
    VALUES(AlbumPair, sequ, TrackTitle, TrackWriter, TrackDuration, TrackRecDate,  
TrackStudio, TrackEngineer);
```

```
END;
```

```
FUNCTION get_performer RETURN varchar2 IS
```

```
    BEGIN
```

```
        RETURN current_performer;
```

```
    END;
```

```
PROCEDURE DeleteTrack (AlbumPair char, TrackTitle varchar2) IS
```

```
aux number;

BEGIN

DELETE FROM tracks WHERE PAIR = AlbumPair AND title = TrackTitle;

SELECT COUNT('x') INTO aux FROM tracks WHERE PAIR = AlbumPair;

IF aux=0 THEN

    DELETE FROM albums WHERE PAIR = AlbumPair;

END IF;

END;

PROCEDURE PerformerReport IS

    total_albums number;

    total_tracks number;

    total_concerts number;

BEGIN

    dbms_output.put_line('***REPORT ON ' || current_performer || '***');

    dbms_output.put_line('-----');

    -- NUMBER OF ALBUMS OF EACH TYPE

    for myrow in (select (CASE format WHEN 'V' THEN 'vynil' WHEN 'S' THEN 'single' WHEN 'C'
    THEN 'CD' WHEN 'M' THEN 'audio file' ELSE 'streaming' END) F , count(*) C from albums
    where performer = current_performer group by format)

        LOOP

            dbms_output.put_line('The number of ' || myrow.F || ' albums is ' || myrow.C);

        END LOOP;

    dbms_output.put_line('-----');

    -- AVERAGE NUMBER OF SONG OF EACH TYPE

    for myrow in (select (CASE format WHEN 'V' THEN 'vynil' WHEN 'S' THEN 'single' WHEN 'C'
    THEN 'CD' WHEN 'M' THEN 'audio file' ELSE 'streaming' END) F, round(avg(num_songs),4) A
    from (select pair, format from albums where performer = current_performer) left join
    (select pair, count(*) as num_songs from tracks group by pair) using (pair) group by
    format)

        LOOP
```

```

dbms_output.put_line('The average number of songs of ' || myrow.F || ' albums is ' ||
myrow.A);

        END LOOP;

        dbms_output.put_line('-----');

-- AVERAGE ALBUM DURATION DEPENDING ON EACH TYPE

FOR myrow IN (select (CASE format WHEN 'V' THEN 'vynil' WHEN 'S' THEN 'single' WHEN 'C'
THEN 'CD' WHEN 'M' THEN 'audio file' ELSE 'streaming' END) F, round(avg(dur),4) D from
((select pair, format from albums where performer = current_performer) left join (select
pair, sum(duration) dur from tracks group by pair) using (pair)) group by format)

        LOOP

dbms_output.put_line('The average duration of ' || myrow.F || ' albums is ' || myrow.D
|| ' minutes');

        END LOOP;

        dbms_output.put_line('-----');

-- PERIODICITY

for myrow in (select round((max(rel_date) - min(rel_date))/count('x'),0) P from albums
where performer = current_performer)

        LOOP

dbms_output.put_line('The periodicity with which ' || current_performer || ' airs an album
is ' || myrow.P || ' days');

        END LOOP;

        dbms_output.put_line('-----');

-- PUBLISHERS REPORT

select count('x') into total_albums from albums where performer = current_performer;

for myrow in (select publisher P, count('x') A from albums where performer =
current_performer group by publisher)

        LOOP

dbms_output.put_line('Publisher ' || myrow.P || ' collaborated in ' || myrow.A || ' albums
(' || round((myrow.A/total_albums)*100,2) || '%)');

        END LOOP;

        dbms_output.put_line('-----');

```



-- STUDIOS REPORT

```
select count(pair) into total_tracks from tracks where pair in (select pair from albums
where performer = current_performer);
```

```
for myrow in (select studio S, count('x') T from (select distinct pair, studio from
tracks where pair in (select pair from albums where performer = current_performer))
group by studio)
```

LOOP

```
dbms_output.put_line('Studio '||myrow.S||' collaborated in '||myrow.T||' tracks
('||round((myrow.T/total_tracks)*100,2)||'%'));;
```

END LOOP;

```
dbms_output.put_line('-----');
```

-- ENGINEERS REPORT

```
for myrow in (select engineer E, count('x') T from (select distinct pair, engineer from
tracks where pair in (select pair from albums where performer = current_performer))
group by engineer)
```

LOOP

```
dbms_output.put_line('Engineer '||myrow.E||' collaborated in '||myrow.T||' tracks
('||round((myrow.T/total_tracks)*100,2)||'%'));;
```

END LOOP;

```
dbms_output.put_line('-----');
```

-- MANAGERS REPORT

```
select count('x') into total_concerts from concerts where performer = current_performer;
```

```
for myrow in (select name N, (CASE WHEN A IS NULL THEN 0 ELSE A END) AL, (CASE WHEN C
IS NULL THEN 0 ELSE C END) CO from (select name, A from (select manager M, count('x') A
from albums where performer = current_performer group by manager) left join (select
name, mobile M from managers) using (M)) full join (select name, C from (select manager
M, count('x') C from concerts where performer = current_performer group by manager) left
join (select name, mobile M from managers) using (M)) using (name))
```

LOOP

```
dbms_output.put_line('Manager '||myrow.N||' collaborated in '||myrow.AL||' albums
('||round((myrow.AL/total_albums)*100,2)||'%'));;
```

```
dbms_output.put_line(' and '||myrow.CO||' concerts
('||round((myrow.CO/total_concerts)*100,2)||'%'));;
```

```
        END LOOP;  
    END;  
END melopack;  
  
/
```

## C. Test

Now we will test that all the procedures work correctly.

- **AssignPerformer:** To test this function we add the function get\_performer to the package

```
FUNCTION get_performer  
RETURN varchar2  
IS  
BEGIN  
RETURN current_performer;  
END;  
  
/  
  
SET SERVEROUTPUT ON;  
EXEC melopack.AssignPerformer('Eva');  
BEGIN dbms_output.put_line(melopack.get_performer);END;/
```

```
PL/SQL procedure successfully completed.
```

```
Eva
```

```
PL/SQL procedure successfully completed.
```

- **InsertAlbum:**

We make sure that the album and tracks we are going to insert do not exist

```
SELECT * FROM tracks WHERE pair = 'A0011LC45433FEA';
```

```
SELECT * FROM albums WHERE pair = 'A0011LC45433FEA';
```

PAIR	SEQU	TITLE	WRITER	DURATION	REC_DATE	STUDIO	ENGINEER
------	------	-------	--------	----------	----------	--------	----------

```
EXEC melopack.InsertAlbum('A0011LC45433FEA', 'Abetos', 'SE>>0467628803',7, '3/04/1999',
'Jurado Studios','JUAN', 'C', 'City of discussion', '17/09/96', 'Mandaluna', 555299599);
```

```
select * from tracks where pair = 'A0011LC45433FEA';
```

PAIR	SEQU	TITLE	WRITER	DURATION	REC_DATE	STUDIO	ENGINEER
1 A0011LC45433FEA	1	Abetos	SE>>0467628803	7	03/04/99	Jurado Studios	JUAN

```
SELECT * FROM albums WHERE pair = 'A0011LC45433FEA';
```

PAIR	PERFORMER	FORMAT	TITLE	REL_DATE	PUBLISHER	MANAGER
1 A0011LC45433FEA	Eva	C	City of discussion	17/09/96	Mandaluna	555299599

As we can see both the track and the album have been correctly inserted. Now we try to insert another track into this album, since the album already exists we only need to specify the parameters for the track

```
EXEC melopack.InsertAlbum('A0011LC45433FEA', 'Fruit gate', 'SE>>0206964155',8,
'3/04/1999', 'Jurado Studios','JUAN');
```

```
select * from tracks where pair = 'A0011LC45433FEA';
```

PAIR	SEQU	TITLE	WRITER	DURATION	REC_DATE	STUDIO	ENGINEER
1 A0011LC45433FEA	1	Abetos	SE>>0467628803	7	03/04/99	Jurado Studios	JUAN
2 A0011LC45433FEA	2	Fruit gate	SE>>0206964155	8	03/04/99	Jurado Studios	JUAN

We can be convinced that our procedure works.

### • DeleteTrack:

```
-- TEST 1: delete track from album with many songs
```

```
select * from tracks where pair = 'X9622WJZ171C21 ';
```

```
--Show that the album actually has many tracks
```

PAIR	SEQU	TITLE	WRITER	DURATION	REC_DATE	STUDIO	ENGINEER
4 X9622WJZ171C21	4	Walk	SE>>0744017163	282	07/08/80	Stretchers Studios Milagros Francia Cuadra	
5 X9622WJZ171C21	5	Fortune and girlfriend	SE>>0427489972	273	29/05/80	Stretchers Studios Milagros Francia Cuadra	
6 X9622WJZ171C21	6	Venom	SE>>0475763296	358	19/05/80	Stretchers Studios Milagros Francia Cuadra	
7 X9622WJZ171C21	7	Key	SE>>0589180598	283	02/09/80	Stretchers Studios Milagros Francia Cuadra	
8 X9622WJZ171C21	8	Montains of balconies	SE>>0652478417	290	15/05/80	Stretchers Studios Milagros Francia Cuadra	
9 X9622WJZ171C21	9	Taste or sailboat	GB>>0374760762	168	26/07/80	Stretchers Studios Milagros Francia Cuadra	
10 X9622WJZ171C21	10	Quixote and mercury	SE>>0880748275	376	16/07/80	Stretchers Studios Milagros Francia Cuadra	
11 X9622WJZ171C21	11	Rock of grape harvest	SE>>0414837052	371	07/07/80	Stretchers Studios Milagros Francia Cuadra	

```
select * from tracks where title = 'Songs' and pair = 'X9622WJZ171C21';
```

--Show track "Songs" from album

PAIR	SEQU	TITLE	WRITER	DURATION	REC_DATE	STUDIO	ENGINEER
1 X9622WJZ171C21	13	Songs	SE>>0914103746	198	12/05/80	Stretchers Studios	Milagros Francia Cuadra

```
EXEC melopack.DeleteTrack('X9622WJZ171C21', 'Songs');
```

-- Delete track with procedure

```
PL/SQL procedure successfully completed.
```

```
select * from tracks where title = 'Songs' and pair = 'X9622WJZ171C21';
```

-- Check that it has actually been deleted

PAIR	SEQU	TITLE	WRITER	DURATION	REC_DATE	STUDIO	ENGINEER
------	------	-------	--------	----------	----------	--------	----------

-- TEST 2: delete last track of album

We managed to select an album with only two tracks:

TITLE	PAIR
1 Twist	A0058ZEF6365P8
2 Finish how	A0058ZEF6365P8

```
EXEC melopack.DeleteTrack('A0058ZEF6365P8 ', 'Twist');
```

-- We delete a track:

TITLE	PAIR
1 Finish how	A0058ZEF6365P8

```
EXEC melopack.DeleteTrack('A0058ZEF6365P8 ', 'Finish how');
```

-- We delete the other one:

TITLE	PAIR
-------	------

```
select pair from albums where pair='A0058ZEF6365P8 ';
```

--We can see the album was deleted too:



- **PerformerReport:**

```
set serveroutput on;
```

```
exec melopack.AssignPerformer('Grapas');
```

```
exec melopack.PerformerReport;
```

```
***REPORT ON Grapas***
-----
The number of single albums is 15
The number of vinyl albums is 12
-----
The average number of songs of single albums is 2
The average number of songs of vinyl albums is 13,8333
-----
The average duration of vinyl albums is 3871,25 minutes
The average duration of single albums is 531,5333 minutes
-----
The periodicity with which Grapas airs an album is 212 days
-----
Publisher Fuerteventura Inc. collaborated in 27 albums (100%)
-----
Studio Estudios de Grabaci?n Pablo Lopez collaborated in 2 tracks (1,02%)
Studio Walnut Tree Recordings collaborated in 3 tracks (1,53%)
Studio Lonely Recordings collaborated in 1 tracks (,51%)
Studio Doctor Recordings collaborated in 3 tracks (1,53%)
Studio Coli Inc. collaborated in 2 tracks (1,02%)
Studio Estudios de Grabaci?n Moon collaborated in 3 tracks (1,53%)
Studio Estudios Madonna Lily collaborated in 2 tracks (1,02%)
Studio Estudios de Grabaci?n Nasas collaborated in 2 tracks (1,02%)
Studio Agustico Inc. collaborated in 2 tracks (1,02%)
Studio Martinez Recordings collaborated in 2 tracks (1,02%)
Studio Baker Recordings collaborated in 1 tracks (,51%)
Studio Crowdy Studios collaborated in 2 tracks (1,02%)
Studio Ruiz Recordings collaborated in 6 tracks (3,06%)
-----
Engineer Erika Valencia collaborated in 3 tracks (1,53%)
Engineer Nabal Molina Perez collaborated in 2 tracks (1,02%)
Engineer Agustin Enrique Guzman collaborated in 2 tracks (1,02%)
Engineer Ramon Ruiz Bastidas collaborated in 6 tracks (3,06%)
```

## BS DEGREE IN INFORMATICS ENGINEERING

Academic year: 2022/2023 - 2<sup>nd</sup> year, 2<sup>nd</sup> term

Subject: File Structures and Databases

Second Assignment's Report: DB development and Querying

Engineer Janet Mamani Oca?a collaborated in 1 tracks (,51%)  
Engineer Rosa Chavela Martinez Martinez collaborated in 2 tracks (1,02%)  
Engineer Maria del Henar Sanz Calzado collaborated in 2 tracks (1,02%)  
Engineer Eduardo Herbada Colorado collaborated in 2 tracks (1,02%)  
Engineer Luis Ceballos collaborated in 2 tracks (1,02%)  
Engineer Agustin Andres Bardales collaborated in 3 tracks (1,53%)  
Engineer Felipe Villazan collaborated in 1 tracks (,51%)  
Engineer Pablo Garcia Lopez collaborated in 2 tracks (1,02%)  
Engineer Amable Lopez Caverro collaborated in 3 tracks (1,53%)

-----  
Manager Ada collaborated in 0 albums (0%)  
and 1 concerts (,83%)  
Manager Alicia "Alicica" collaborated in 0 albums (0%)  
and 1 concerts (,83%)  
Manager Antonio Fernando collaborated in 0 albums (0%)  
and 1 concerts (,83%)  
Manager Armando collaborated in 4 albums (14,81%)  
and 17 concerts (14,05%)  
Manager Aurea collaborated in 2 albums (7,41%)  
and 12 concerts (9,92%)  
Manager Candidiano collaborated in 0 albums (0%)  
and 1 concerts (,83%)  
Manager Cesarea collaborated in 0 albums (0%)  
and 2 concerts (1,65%)  
Manager Derfuta collaborated in 3 albums (11,11%)  
and 14 concerts (11,57%)  
Manager Dulce collaborated in 0 albums (0%)  
and 1 concerts (,83%)  
Manager Edgar collaborated in 0 albums (0%)  
Manager Esteban collaborated in 0 albums (0%)  
and 1 concerts (,83%)  
Manager Felix collaborated in 0 albums (0%)  
and 1 concerts (,83%)  
Manager Grazia collaborated in 0 albums (0%)  
and 1 concerts (,83%)  
Manager Jorge "Jorge" collaborated in 11 albums (40,74%)  
and 46 concerts (38,02%)  
Manager Maikel collaborated in 0 albums (0%)  
and 1 concerts (,83%)  
Manager Manuel collaborated in 0 albums (0%)  
and 1 concerts (,83%)  
Manager Maria de la Inocencia collaborated in 0 albums (0%)  
and 1 concerts (,83%)  
Manager Maria de los Dolores collaborated in 0 albums (0%)  
and 2 concerts (1,65%)  
Manager Mariluz collaborated in 2 albums (7,41%)  
and 11 concerts (9,09%)  
Manager Petra Patricia collaborated in 0 albums (0%)  
and 1 concerts (,83%)  
Manager Rosa Cayetana collaborated in 0 albums (0%)  
and 1 concerts (,83%)  
Manager Simon collaborated in 0 albums (0%)  
and 1 concerts (,83%)  
Manager Tina collaborated in 0 albums (0%)  
and 1 concerts (,83%)  
Manager Tomaso collaborated in 0 albums (0%)  
and 1 concerts (,83%)  
Manager Maria de la Almudena collaborated in 5 albums (18,52%)  
and 0 concerts (0%)

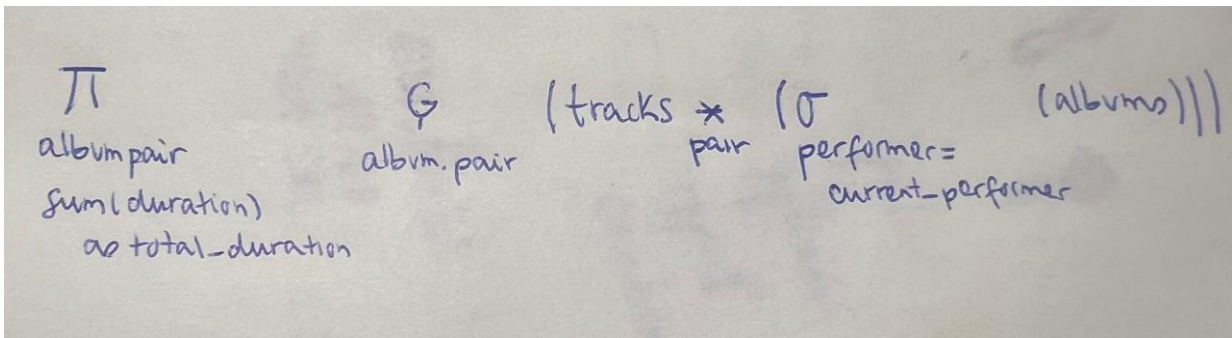
PL/SQL procedure successfully completed.



## 4. EXTERNAL DESIGN (views)

### First View:

### Relational Algebra Design



### SQL Implementation

```
CREATE OR REPLACE VIEW my_albums AS
(SELECT pair, sum(duration) total_duration FROM
tracks JOIN albums USING (pair)
WHERE albums.performer=melopack.get_performer
GROUP BY pair) WITH READ ONLY;
```

### TEST

When creating this view, we should get a list of all the albums of our current performer and the corresponding duration of each of them. We will test this with the artist “Grapas”.

```
exec melopack.assignperformer('Grapas');
```

Now, if we create the view, we obtain this:



# BS DEGREE IN INFORMATICS ENGINEERING

Academic year: 2022/2023 - 2<sup>nd</sup> year, 2<sup>nd</sup> term

Subject: File Structures and Databases

Second Assignment's Report: DB development and Querying

	PAIR	TOTAL_DURATION
1	H19911TH75267AC	459
2	U8613X4X7395YK0	368
3	Q0896Q7U6891BRQ	3783
4	N3273GIR5450VHP	3932
5	B9526FEA71798WW	4243
6	O3970K7J766PS7	657
7	Z3039K25876177Y	652
8	U07821FN9862EYS	495
9	E254440T5670LCU	3780
10	V2077M1F4417DY5	3700
11	A3349PGM3342KQG	526
12	I9075ZRM3525420	501
13	O3846SG36015656	744
14	X8647VSR7360XDO	4123
15	V9896NRT656022R	467
16	E4683SE01983U1	3753
17	N4516MZA2405Y44	381
18	L2478HZI1212UGV	609
19	W2733UYR4932FGM	625
20	A0892BVH7237WTB	3697
21	T7408O505274ZEN	561
22	K9925NYV3639MKL	3807
23	R3976FP962155P8	3912
24	K9758JJM2972STJ	4098
25	V7576SY51558DUJ	623
26	O1321KMI3551NMQ	305
27	W4174ULT6058I4G	3627

All the albums pair are different and, with a query, we can check that Grapas has in fact 27 albums

```
select count('x') from albums where performer = 'Grapas';
```

Now, if we update the albums table, for example deleting an album of Grapas, that should be changed also in the view automatically.

```
delete from albums where pair = 'H19911TH75267AC';
```

## BS DEGREE IN INFORMATICS ENGINEERING

Academic year: 2022/2023 - 2<sup>nd</sup> year, 2<sup>nd</sup> term

Subject: File Structures and Databases

Second Assignment's Report: DB development and Querying

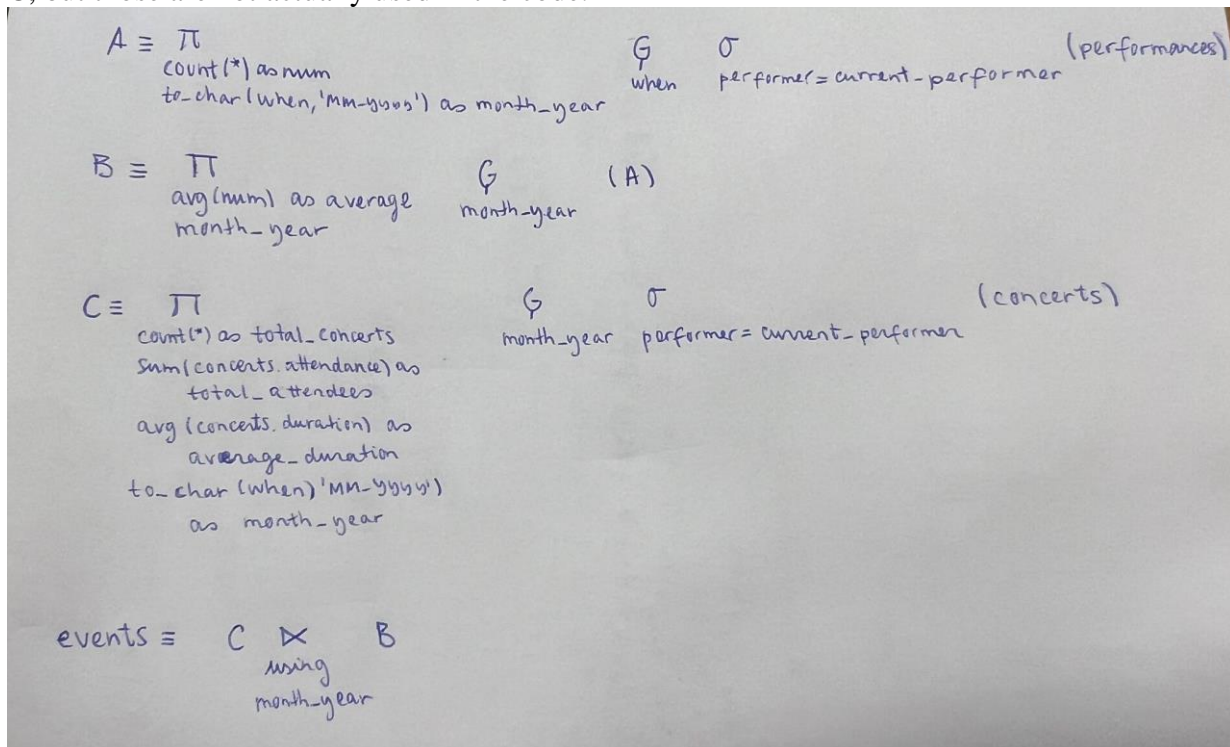
PAIR	TOTAL_DURATION
1 U8613X4X7395YK0	368
2 N3273GIR5450VHP	3932
3 Q0896Q7U6891BRQ	3783
4 B9526FEA71798WW	4243
5 O3970K7J766PS7	657
6 Z3039K25876177Y	652
7 U07821FN9862EYS	495
8 E254440T5670LCU	3780
9 V2077M1F4417DY5	3700
10 A3349PGM3342KQG	526
11 I9075ZRM3525420	501
12 V9896NRT6560Z2R	467
13 O3846SG36015656	744
14 X8647VSR7360XDO	4123
15 E4683SE01983U1	3753
16 N4516MZA2405Y44	381
17 W2733UYR4932FGM	625
18 A0892BVH7237WTB	3697
19 L2478HZI1212UGV	609
20 T7408O505274ZEN	561
21 K9925NYV3639MKL	3807
22 R3976FP962155P8	3912
23 V7576SY51558DUJ	623
24 K9758JMJ2972STJ	4098
25 O1321KMI3551NMQ	305
26 W4174ULT6058I4G	3627

As we can see, the view was actually updated.

## Second View:

### Relational Algebra Design

In order to simplify the visual representation of the relational algebra design, we use labels A, B and C, but those are not actually used in the code.



### SQL Implementation

```
CREATE OR REPLACE VIEW events AS
(select month_year, total_concerts, total_attendees, average
FROM
(select count(*) total_concerts, sum(concerts.attendance) total_attendees,
avg(concerts.duration) average_duration, to_char(when, 'MM/YYYY') month_year
FROM concerts WHERE performer=melopack.get_performer
GROUP BY to_char(when, 'MM/YYYY')) tabla1
LEFT JOIN
(select avg(num) average, month_year FROM ((select count(*) num,
to_char(when, 'MM/YYYY') month_year FROM performances WHERE
performer=melopack.get_performer
GROUP BY when))
GROUP BY month_year) tabla2
USING (month_year)) WITH READ ONLY;
```

## TEST

When creating this view, we should get a list of all the concert activity per month and year, displaying concerts, total attendees, average duration, and average number of performances. We will test this with the artist “Grapas”.

```
exec melopack.assignperformer('Grapas');
```

This are the first 33 rows from what we get, although we actually get 66 rows:

	MONTH_YEAR	TOTAL_CONCERTS	TOTAL_ATTENDEES	AVERAGE
1	08/1992	2	0	11
2	06/1997	1	0	12
3	08/1993	1	0	11
4	10/1997	1	0	12
5	09/1989	3	0	11
6	09/1995	1	0	12
7	08/1996	2	0	12
8	06/1986	2	0	10
9	11/1985	1	0	10
10	05/1995	1	0	12
11	09/1987	1	0	11
12	06/1996	1	0	12
13	06/1991	1	0	11
14	07/1992	2	0	11
15	10/1992	3	0	11
16	10/1985	1	0	9
17	08/1990	3	0	11
18	05/1985	1	0	10
19	05/1997	1	0	12
20	05/1989	1	0	11
21	09/1990	1	0	11
22	10/1986	2	0	11
23	05/1986	3	0	10
24	06/1987	2	0	11
25	09/1986	1	0	11
26	10/1989	1	0	11
27	10/1993	1	0	11
28	09/1994	1	0	12
29	08/1987	5	0	11
30	08/1989	1	0	11
31	05/1987	3	0	10
32	06/1993	2	0	11
33	09/1996	1	0	12

Now, to test it, we will delete the concerts occurred in 08/1992 using the following code:

delete from concerts where to\_char(when,'MM/YYYY') = '08/1992' and performer = 'Grapas';

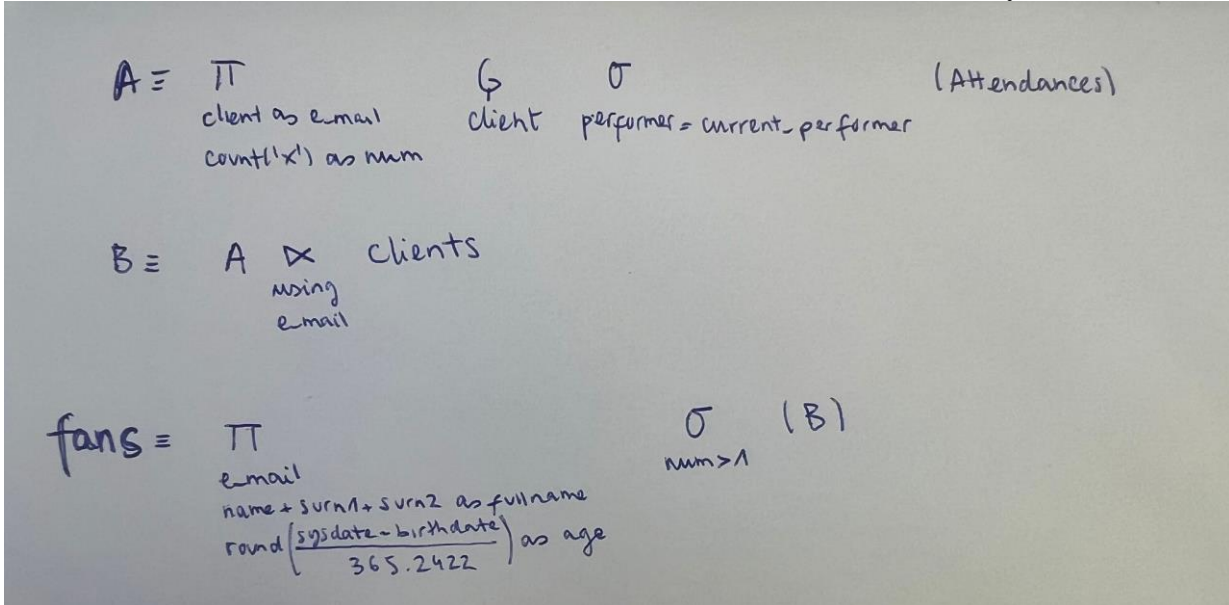
Now, we can see that in the view there is not any concert in that period:

	MONTH_YEAR	TOTAL_CONCERTS	TOTAL_ATTENDEES	AVERAGE
1	06/1997	1	0	12
2	08/1993	1	0	11
3	10/1997	1	0	12
4	09/1989	3	0	11
5	09/1995	1	0	12
6	08/1996	2	0	12
7	06/1986	2	0	10
8	11/1985	1	0	10
9	05/1995	1	0	12
10	09/1987	1	0	11
11	06/1996	1	0	12
12	06/1991	1	0	11
13	07/1992	2	0	11
14	10/1992	3	0	11
15	10/1985	1	0	9
16	08/1990	3	0	11
17	05/1985	1	0	10
18	05/1997	1	0	12
19	05/1989	1	0	11
20	09/1990	1	0	11
21	10/1986	2	0	11
22	05/1986	3	0	10
23	06/1987	2	0	11
24	09/1986	1	0	11
25	10/1989	1	0	11
26	10/1993	1	0	11
27	09/1994	1	0	12
28	08/1987	5	0	11
29	08/1989	1	0	11
30	05/1987	3	0	10
31	06/1993	2	0	11
32	09/1996	1	0	12
33	09/1992	1	0	11

### Third View:

### Relational Algebra Design

As well as in the second view, we use labels A and B, but those are not actually used in the code.



Handwritten Relational Algebra expressions:

$$A \equiv \pi_{\text{client as e\_mail, count('x') as num}}(\sigma_{\text{performer = current\_performer}}(\text{Attendances}))$$

$$B \equiv A \bowtie_{\text{using e\_mail}} \text{clients}$$

$$\text{fans} \equiv \pi_{\text{e\_mail, name + surn1 + surn2 as fullname, round}\left(\frac{\text{sysdate - birthdate}}{365.2422}\right) \text{ as age}}(\sigma_{\text{num > 1}}(B))$$

### SQL Implementation

Here is the implementation of a read only version of the view fans. We were not able to achieve the full operativity of this view given several problems when implementing the necessary triggers to do so.

```
create or replace view fans as
  select e_mail, name||' '||surn1||' '||surn2 as fullname, round(((SYSDATE -
  birthdate)/ 365.2422)) as age
  from
    ((select client as e_mail, count('x') num
      from attendances
      where performer = melopack.get_performer
      group by client)
  left join
    (select * from clients)
    using (e_mail))
  where num > 1;
```



## TEST

If we create the view taking as current performer “Amapola”, we get the following. These are the first eleven rows, but there are many others.

```
exec melopack.assignperformer('Amapola');
```

	E_MAIL	FULLNAME	AGE
1	torda@clients.vinylinc.com	Belen Lopez Diaz	48
2	alejandro@clients.vinylinc.com	Diego Alejandro Perez	38
3	martino@clients.vinylinc.com	Martin "Martino" Vazquez	39
4	ernestinarojas@clients.vinylinc.com	Ernestina Rojas	55
5	diaz@clients.vinylinc.com	Agustin Diaz-Marin	28
6	fidelagomez@clients.vinylinc.com	Fidela Lopez Gomez	37
7	agerolaque@clients.vinylinc.com	Jeronimo AgDero Laque	46
8	tomasortiz@clients.vinylinc.com	Tomas Jesus Alvarado Ortiz	50
9	iulenecalixto@clients.vinylinc.com	Iulene Gayoso Calixto	43
10	ngelesesquiche@clients.vinylinc.com	Maria de los Ongeles Chuan	74
11	tallest@clients.vinylinc.com	Mario Ramon Medina	41

Now, if for example we delete Torda from clients, we will get the view updated as well:

```
delete from attendances where client = 'torda@clients.vinylinc.com';
```

	E_MAIL	FULLNAME	AGE
1	alejandro@clients.vinylinc.com	Diego Alejandro Perez	38
2	martino@clients.vinylinc.com	Martin "Martino" Vazquez	39
3	ernestinarojas@clients.vinylinc.com	Ernestina Rojas	55
4	diaz@clients.vinylinc.com	Agustin Diaz-Marin	28
5	fidelagomez@clients.vinylinc.com	Fidela Lopez Gomez	37
6	agerolaque@clients.vinylinc.com	Jeronimo AgDero Laque	46
7	tomasortiz@clients.vinylinc.com	Tomas Jesus Alvarado Ortiz	50
8	iulenecalixto@clients.vinylinc.com	Iulene Gayoso Calixto	43
9	ngelesesquiche@clients.vinylinc.com	Maria de los Ongeles Chuan	74



## 5. EXPLICITLY REQUIRED TRIGGERS

### First Trigger:

The first trigger consists on updating the duration of a concert after the insertion of a song in the concert. It is performed **after inserting** in the table *PERFORMANCES* and takes the duration of the song and adds it to the current duration of the concert, this trigger is done **row by row**.

### CODE:

```
CREATE OR REPLACE TRIGGER inserting_song
AFTER INSERT ON performances
FOR EACH ROW
DECLARE
BEGIN
    UPDATE CONCERTS SET duration=duration+:NEW.duration WHERE performer=:NEW.performer
    and when=:NEW.when;
END inserting_song;
/
```

### TESTS:

If we check the state of the concert before inserting any songs we get:

```
select * from concerts where performer = 'Abdi' and when='12/10/85';
```

	PERFORMER	WHEN	TOUR	MUNICIPALITY	ADDRESS	COUNTRY	ATTENDANCE	DURATION	MANAGER
1	Abdi	12/10/85	(null)	Lemmonvalley	46 Saint Francis Street	Gabon	0	124	555448432

As we can see, the duration of the concert is initially 124.

```
insert into performances(performer, when, sequ, songtitle, songwriter, duration)
VALUES ('Abdi','12/10/85', 12, 'Abetos', 'SE>>0467628803', 10);
```

```
select * from concerts where performer = 'Abdi' and when='12/10/85';
```

	PERFORMER	WHEN	TOUR	MUNICIPALITY	ADDRESS	COUNTRY	ATTENDANCE	DURATION	MANAGER
1	Abdi	12/10/85	(null)	Lemmonvalley	46 Saint Francis Street	Gabon	0	134	555448432

After inserting the duration is 134.

We can also try to insert multiple rows at the same time, for this we have created an auxiliary table with the same structure as *PERFORMANCES*, as we can see all the rows are inserted correctly.

```
insert into aux(performer, when, sequ, songtitle, songwriter, duration)
VALUES ('Abdi','12/10/85', 420, 'Abetos', 'SE>>0467628803', 10);
insert into aux(performer, when, sequ, songtitle, songwriter, duration)
VALUES ('Abdi','12/10/85', 421, 'Abetos', 'SE>>0467628803', 10);
```

```
insert into performances(performer, when, sequ, songtitle, songwriter, duration)
(SELECT * from xd);
```

PERFORMER	WHEN	TOUR	MUNICIPALITY	ADDRESS	COUNTRY	ATTENDANCE	DURATION	MANAGER
1 Abdi	12/10/85	(null)	Lemmonvalley	46 Saint Francis Street	Gabon	0	154	555448432

Now the duration has increased by 20 units

We also check that the songs have been inserted:

1	Abdi	12/10/85	12	Abetos	SE>>0467628803	10
2	Abdi	12/10/85	420	Abetos	SE>>0467628803	10
3	Abdi	12/10/85	421	Abetos	SE>>0467628803	10

## Second Trigger:

The second trigger consists on checking the age of the attendee **before being inserted** in the table *CLIENTS*, it checks the age **row by row**, doing an if statement that subtracts the birth date of the person from the date of the day and converts it to years, it is then checked if it is less than 18. If the person is underage the insertion is not performed and an exception is executed.

## CODE:

```
CREATE OR REPLACE TRIGGER underage_client
BEFORE INSERT ON clients
FOR EACH ROW
DECLARE
    underage exception;
BEGIN
    IF (SYSDATE-:NEW.birthdate)/ 365.242199 < 18
    THEN RAISE underage;
    END IF;
EXCEPTION
    WHEN underage THEN RAISE_APPLICATION_ERROR (-20001,'CLIENT IS UNDERAGE CANNOT ADD!');
END underage_client;
/
```

## TESTS:

We try inserting a client who is underage:

```
insert into clients(e_mail, name, surn1, surn2, birthdate, phone, address, dni)
VALUES ('aaaa', 'Pia Feliciano', 'Byrne', 'Lopez', '1/04/2005', 555965130, 'Avenue
Fortress, N 68, PA-71762, Panama', 9);
```

```
Error starting at line : 16 in command -
insert into clients(e_mail, name, surn1, surn2, birthdate, phone, address, dni)
VALUES ('aaaa', 'Pia Feliciano', 'Byrne', 'Lopez', '1/04/2005', 555965130, 'Avenue Fortress, N 68, PA-71762, Panama', 9)
Error report -
ORA-20001: CLIENT IS UNDERAGE CANNOT ADD!
ORA-06512: at "FSDB265.UNDERAGE_CLIENT", line 8
ORA-04088: error during execution of trigger 'FSDB265.UNDERAGE_CLIENT'
```

We can see that we get the expected error

Now try to insert a client who is not underage to see if the trigger accepts the insertion:

```
insert into clients(e_mail, name, surn1, surn2, birthdate, phone, address, dni)
VALUES ('aaaa', 'Pia Feliciano', 'Byrne', 'Lopez', '30/03/2005', 555965130, 'Avenue
Fortress, N 68, PA-71762, Panama', 9);

select * from clients where e_mail='aaaa';
```

E_MAIL	NAME	SURN1	SURN2	BIRTHDATE	PHONE	ADDRESS	DNI
1 aaaa	Pia Feliciano	Byrne	Lopez	30/03/05	555965130	Avenue Fortress, N 68, PA-71762, Panama	9

We can see that the row has been correctly inserted.

## Third Trigger:

The third trigger consists on preventing the insertion of a row in the table *SONGS* if a row with the fields writer and cowriter already exists on the table. A check is performed **before the insertion of each row**, if the row already exists an exception is raised.

We had a lot of problems developing this trigger because of the mutating table error and we reached a solution so it is possible that the solution not as efficient as it could have been, still we are proud to have archived a working trigger.

First we create an auxiliary table and copy the data from songs onto it using a loop, then comparing with every row, we check that the swapped values are not in the table using another loop.

## CODE:

```
CREATE OR REPLACE TRIGGER cowriter_swap
FOR INSERT ON songs
COMPOUND TRIGGER
    TYPE aux_type IS RECORD(
        title songs.title%TYPE,
        writer songs.writer%TYPE,
        cowriter songs.cowriter%TYPE
    );
    TYPE t_aux_type IS TABLE OF aux_type index by simple_integer;
    t_aux t_aux_type;

BEFORE STATEMENT IS
BEGIN
    FOR i in(
        select rownum, title, writer, cowriter from songs order by rownum) loop
        t_aux(i.rownum).title := i.title;
        t_aux(i.rownum).writer := i.writer;
        t_aux(i.rownum).cowriter := i.cowriter;
    end loop;
END BEFORE STATEMENT;
BEFORE EACH ROW IS
    wri songs.writer%TYPE;
    writer_cowriter_swap exception;
BEGIN
    FOR i in 1..t_aux.COUNT loop
        IF t_aux(i).title = :NEW.title and t_aux(i).writer = :NEW.cowriter and
t_aux(i).cowriter = :NEW.writer
            THEN RAISE writer_cowriter_swap;
        END IF;
    END LOOP;
EXCEPTION
    WHEN writer_cowriter_swap THEN RAISE_APPLICATION_ERROR(-20002,'SONG ALREADY
INSERTED WITH FIELDS SWAPPED!');
END BEFORE EACH ROW;
END cowriter_swap;
/
```

## TESTS:

We try to insert:

```
insert into songs(title, writer, cowriter)
VALUES ('Abetos acabar', 'FI>>0357833358', 'ES>>0695626316');
insert into songs(title, writer, cowriter)
VALUES ('Abetos acabar', 'ES>>0695626316', 'FI>>0357833358');
```

```

Error starting at line : 38 in command -
insert into songs(title, writer, cowriter)
  VALUES ('Abetos acabar', 'ES>>0695626316', 'FI>>0357833358')
Error report -
ORA-20002: SONG ALREADY INSERTED WITH FIELDS SWAPPED!
ORA-06512: at "FSDB265.COWRITER_SWAP", line 29
ORA-04088: error during execution of trigger 'FSDB265.COWRITER_SWAP'

```

We can see that the expected error is raised

Now try to insert multiple rows at a time: (This is the point in which we obtained a mutating table error before)

We create an auxiliary table to do so:

```

CREATE TABLE aux(
  title      VARCHAR2(50),
  writer     CHAR(14),
  cowriter   CHAR(14)
);

insert into aux(title, writer, cowriter)
  VALUES ('Bag', 'CA>>0192153627', 'ES>>0695626316');
insert into aux(title, writer, cowriter)
  VALUES ('Bag', 'FI>>0357833358', 'CA>>0192153627');



insert into songs (select * from aux);

select * from songs where title='Bag' and (writer='CA>>0192153627' or
writer='FI>>0357833358');

```

	TITLE	WRITER	COWRITER
1	Bag	CA>>0192153627	ES>>0695626316
2	Bag	FI>>0357833358	CA>>0192153627

In this case the data is correctly inserted.

BS DEGREE IN INFORMATICS ENGINEERING Academic year: 2022/2023 - 2 <sup>nd</sup> year, 2 <sup>nd</sup> term Subject: File Structures and Databases Second Assignment's Report: DB development and Querying	 
--	---

## 6. CONCLUDING REMARKS

We have achieved a satisfactory result as we have been able to perform almost all of the structures assigned, excluding the triggers needed to make view 3 fully-operable. We consider that the semantic coverage reached is very complete taking into account the different limitations of Oracle and the relational design of the first assignment.

About the documentation, we found the videos on Aula Global really helpful. However, in some cases we also needed to look for information on the internet, in the Oracle documentation or webs as Stack Overflow and others.

As a remark, we found it interesting to use the tool of SQL Developer to better understand the functioning of the structures of our database, since we are beginners in SQL we found it much easier to only have to focus on the sql problems and not have to deal with a very difficult to manage interface that would have made the experience much more frustrating than it has already been.

Although a large amount of hours was required to perform this lab work, we acquired strong knowledge on managing the main structures of SQL querying, triggering, packaging, and Oracle environment.

For further editions, we recommend to change the delivering format, for example giving smaller weekly assignments instead of a huge assignment for a single deadline so that the students can receive some feedback on the correct or more efficient way to do the development after trying to do a component.

We think maybe our solutions have not been very efficient and maybe this way we would have been able to achieve better results.