BS Degree in Informatics Engineering Academic year: 2022/2023 - 2nd year, 2nd term Subject: File Structures and Datab ases Second Assignment's Report: DB development and Querying	uc3m uc3m | Universidad Carlos III de Madrid

Lecturer:	**FRANCISCO JAVIER CALLE GÓMEZ**		
Group:	**89**	**Lab User**	**fsdb253**
Student:	**ÁLVARO CABRERA NIETO**	**NIA:**	**100472152**
Student:	**GONZALO CARRETERO HERNÁNDEZ**	**NIA:**	**100472147**
Student:	**JIAHAO CHEN**	**NIA:**	**100472232**

1 Introduction

The task at hand involves designing Queries, Packages, Views, and Triggers for a given database. The objective is to accurately execute each component according to the given specifications. In order to meet this goal, it is necessary to provide detailed documentation for each element, including a thorough explanation of the solution, the accompanying code, and relevant tests that demonstrate its functionality. The document will be structured according to the following index:

1.1 QUERIES:

1.1.1 Self-sufficient:

- **% of tracks**
  - **Relational algebra:**

Band_members ≡ (INVOLVEMENT ϴ (band=performer) **ALBUMS)**

Recorded_songs ≡ (TRACKS ✳ (title, writer) **SONGS)**

Songs ≡ (Band_members ϴ ((musician=cowriter OR musician=writer) AND PAIR = PAIR) **Recorded_songs**

Songs_Grouped ≡ π (PAIR, performer, sequ) **Songs**

Song_Counts ≡ π (performer, COUNT(x) AS counts) **Ɠ** (performer) **Songs_Grouped**

Album_Totals ≡ π (COUNT(x) AS total, performer) **Ɠ** (performer) **(ALBUMS ✳** (PAIR) **TRACKS)**

Percentage ≡ π ROUND(counts/total, 2) **Song_Counts ✳** (performer) **Album_Totals**

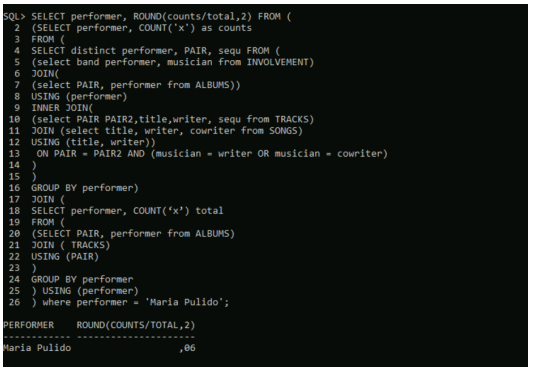
Result ≡ π name, COALESCE(ratio, 0) **Performers]* Percentage**

  - **SQL Implementation:**

```
SELECT name, COALESCE(ratio, 0) from Performers LEFT OUTER JOIN (
SELECT performer, ROUND(counts/total,2) ratio FROM (
(SELECT performer, COUNT('x') as counts
FROM (
        SELECT distinct performer, PAIR, sequ FROM (
                (select band performer, musician from INVOLVEMENT)
                JOIN(
                (select PAIR, performer from ALBUMS))
                USING (performer)
                INNER JOIN(
                        (select PAIR PAIR2,title,writer, sequ from TRACKS)
                        JOIN (select title, writer, cowriter from SONGS)
                        USING (title, writer))
                ON PAIR = PAIR2 AND (musician = writer OR musician = cowriter)
        )
```

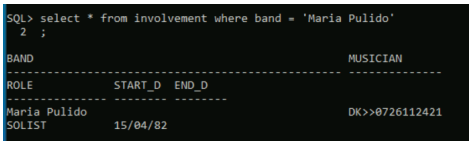
```
)
GROUP BY performer)
JOIN (
        SELECT performer, COUNT('x') total
                FROM (
                        (SELECT PAIR, performer from ALBUMS)
                        JOIN ( TRACKS)
                        USING (PAIR)
                )
                GROUP BY performer
) USING (performer)
)
)ON name = performer;
```

- **Testing:**

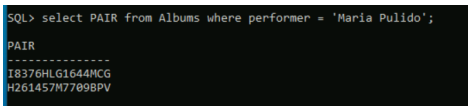
Action	Result obtained
Ratio of performer Maria Pulido	

```
SQL> SELECT performer, ROUND(counts/total,2) FROM (
  2  (SELECT performer, COUNT('x') as counts
  3  FROM (
  4  SELECT distinct performer, PAIR, sequ FROM (
  5  (select band performer, musician from INVOLVEMENT)
  6  JOIN(
  7  (select PAIR, performer from ALBUMS))
  8  USING (performer)
  9  INNER JOIN(
 10  (select PAIR PAIR2,title,writer, sequ from TRACKS)
 11  JOIN (select title, writer, cowriter from SONGS)
 12  USING (title, writer))
 13   ON PAIR = PAIR2 AND (musician = writer OR musician = cowriter)
 14  )
 15  )
 16  GROUP BY performer)
 17  JOIN (
 18  SELECT performer, COUNT('x') total
 19  FROM (
 20  (SELECT PAIR, performer from ALBUMS)
 21  JOIN ( TRACKS)
 22  USING (PAIR)
 23  )
 24  GROUP BY performer
 25  ) USING (performer)
 26  ) where performer = 'Maria Pulido';

PERFORMER    ROUND(COUNTS/TOTAL,2)
------------ ---------------------
Maria Pulido                   ,06
```

Expected Result: 1/16 = 0.0625 (worked as expected)
Reasoning:
She is a soloist and has two albums:

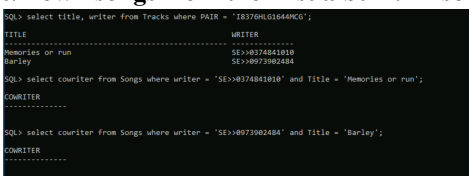


```
SQL> select * from involvement where band = 'Maria Pulido'
  2  ;

BAND                                                MUSICIAN
--------------------------------------------------- --------------
ROLE            START_D  END_D
--------------- -------- --------
Maria Pulido                                        DK>>0726112421
SOLIST          15/04/82
```

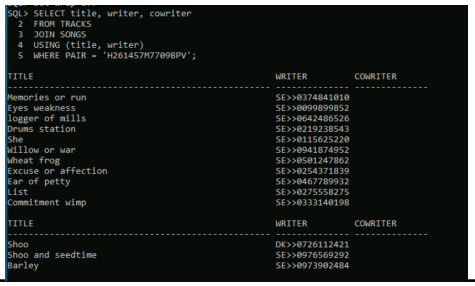


```
SQL> select PAIR from Albums where performer = 'Maria Pulido';

PAIR
---------------
I8376HLG1644MCG
H261457M7709BPV
```

0/2 own songs from the first album. 14 songs for the second album. Maria only participated in "Shoo":



```
SQL> select title, writer from Tracks where PAIR = 'I8376HLG1644MCG';

TITLE                                             WRITER
------------------------------------------------- -------------
Memories or run                                   SE>>0374841010
Barley                                            SE>>0973902484

SQL> select cowriter from Songs where writer = 'SE>>0374841010' and Title = 'Memories or run';

COWRITER
-------------


SQL> select cowriter from Songs where writer = 'SE>>0973902484' and Title = 'Barley';

COWRITER
-------------
```



```
SQL> SELECT title, writer, cowriter
  2  FROM TRACKS
  3  JOIN SONGS
  4  USING (title, writer)
  5  WHERE PAIR = 'H261457M7709BPV';

TITLE                                             WRITER         COWRITER
------------------------------------------------- -------------- --------------
Memories or run                                   SE>>0374841010
Eyes weakness                                     SE>>0009899852
Jogger of mills                                   SE>>0042486526
Drums station                                     SE>>0219238543
She                                               SE>>0115625220
Willow or war                                     SE>>0941874952
Wheat frog                                        SE>>0501147862
Excuse or affection                               SE>>0254371839
Ear of petty                                      SE>>0467789932
List                                              SE>>0275558275
Commitment wimp                                   SE>>0333140198

TITLE                                             WRITER         COWRITER
------------------------------------------------- -------------- --------------
Shoo                                              DK>>0726112421
Shoo and seedtime                                 SE>>0976569292
Barley                                            SE>>0973902484
```

Ratio of performer Reinfangoria

```
SQL> SELECT performer, ROUND(counts/total,2) FROM (
  2  (SELECT performer, COUNT('x') as counts
  3  FROM (
  4  SELECT distinct performer, PAIR, sequ FROM (
  5  (select band performer, musician from INVOLVEMENT)
  6  JOIN(
  7  (select PAIR, performer from ALBUMS))
  8  USING (performer)
  9  INNER JOIN(
 10  (select PAIR PAIR2,title,writer, sequ from TRACKS)
 11  JOIN (select title, writer, cowriter from SONGS)
 12  USING (title, writer))
 13   ON PAIR = PAIR2 AND (musician = writer OR musician = cowriter)
 14  )
 15  )
 16  GROUP BY performer)
 17  JOIN (
 18  SELECT performer, COUNT('x') total
 19  FROM (
 20  (SELECT PAIR, performer from ALBUMS)
 21  JOIN ( TRACKS)
 22  USING (PAIR)
 23  )
 24  GROUP BY performer
 25  ) USING (performer)
 26  ) where performer = 'Reinfangoria';

PERFORMER    ROUND(COUNTS/TOTAL,2)
------------ ---------------------
Reinfangoria                   ,55
```

Expected Result: (0+2+2+7)/(2+2+2+14) = 0.55 (worked as expected)
Reasoning:

```
SQL> select * from involvement where band = 'Reinfangoria';
se truncaran las filas

se truncaran las filas

BAND                                               MUSICIAN       ROLE
-------------------------------------------------- -------------- --------------
Reinfangoria                                       US>>0184635694 Percussion
Reinfangoria                                       US>>0255528753 Guitar
Reinfangoria                                       US>>0279650153 Voice
```

```
SQL> select PAIR from Albums where performer = 'Reinfangoria';

PAIR
---------------
Y269706V2124B09
U218997X8814TWB
C76962EF9249VZR
D810762J2671AEV
```

Band with 3 members and has 4 albums

```
SQL> select title, writer, cowriter
  2  FROM Tracks
  3  JOIN SONGS
  4  USING (title, writer)
  5  where PAIR = 'Y269706V2124B09'
  6  ;

TITLE                                              WRITER         COWRITER
-------------------------------------------------- -------------- --------------
Cap                                                SE>>0032567020
Seconds and neptune                                SE>>0941874952
```

```
SQL> select title, writer, cowriter
  2  FROM Tracks
  3  JOIN SONGS
  4  USING (title, writer)
  5  where PAIR = 'U218997X8814TWB'
  6  ;

TITLE                                              WRITER         COWRITER
-------------------------------------------------- -------------- --------------
Feeling smart                                      US>>0279650153
Dynamite                                           US>>0279650153
```

From 1º Album 0/2 own songs and From 2º Album 2/2 own songs written by one of the members as main writer

```
SQL> select title, writer, cowriter
  2  FROM Tracks
  3  JOIN SONGS
  4  USING (title, writer)
  5  where PAIR = 'C76962EF9249VZR'
  6  ;

TITLE                                              WRITER         COWRITER
-------------------------------------------------- -------------- --------------
Hermitages                                         US>>0255528753 US>>0279650153
Hermitages (ext. version)                          US>>0255528753 US>>0279650153
```

From 3º Album 2/2 own songs written by 2 members of the band together

```
SQL> select title, writer, cowriter
  2  FROM Tracks
  3  JOIN SONGS
  4  USING (title, writer)
  5  where PAIR = 'D810762J2671AEV'
  6  ;

TITLE                                              WRITER         COWRITER
-------------------------------------------------- -------------- --------------
Songs                                              SE>>0529934667
Devil of hurricane                                 US>>0279650153
Seconds and neptune                                SE>>0941874952
Station                                            US>>0184635694 US>>0279650153
Feeling smart                                      US>>0279650153
Heart                                              SE>>0901595934
Twist                                              SE>>0652715321
Affection                                          SE>>0575372923
Blame arena                                        SE>>0087648528
Goblin                                             US>>0279650153
Cap                                                SE>>0032567020

TITLE                                              WRITER         COWRITER
-------------------------------------------------- -------------- --------------
Hermitages                                         US>>0255528753 US>>0279650153
Metal puzzle                                       US>>0279650153
Sailboat                                           US>>0279650153

14 filas seleccionadas.
```

For the last Album: 7/14 own songs,

- 'Devil of hurricane', 'Feeling smart', 'Goblin', 'Metal puzzle' and 'Sailboat' are written by one of the members as main writer
- 'Station' is written by one of the members as cowriter
- 'Hermitages' is written by 2 members of the band together

- **% of Performances**
  - **Relational algebra:**

Recorded_performances ≡ (Performances ✳ (title, writer) **SONGS)**

Songs2 ≡ (Recorded_performances ϴ ((musician=cowriter OR musician=writer) AND performer = band) **Involvement**

Songs_Grouped2 ≡ π (performer, when, sequ) **Songs2**

Song_Counts2 ≡ π (performer, COUNT(x) AS counts) Ǵ (performer) **Songs_Grouped2**

Performances_Totals ≡ π (COUNT(x) AS total, performer) Ǵ (performer) **(Performances)**

Percentage2≡ π ROUND(counts/total, 2) **Song_Counts2** ✳ (performer) **Performances_Totals**

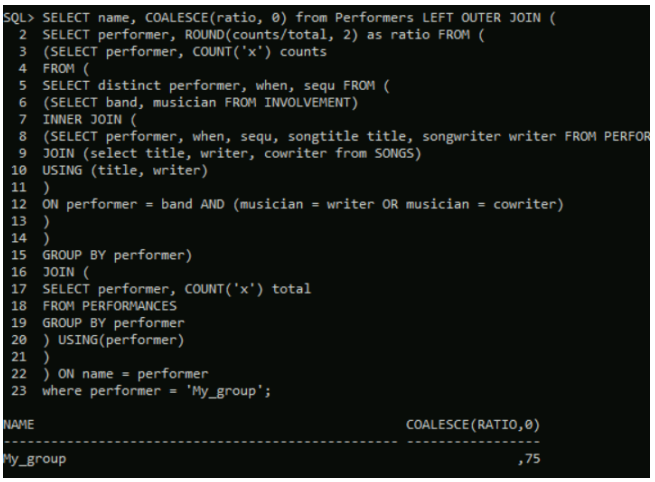
Result2 ≡ π name, e, COALESCE(ratio, 0) **Performers]* Percentage2**

  - **SQL Implementation:**

```
SELECT name, COALESCE(ratio, 0) from Performers LEFT OUTER JOIN (
SELECT performer, ROUND(counts/total, 2) as ratio FROM (
(SELECT performer, COUNT('x') counts
FROM (
        SELECT distinct performer, when, sequ FROM (
                (SELECT band, musician FROM INVOLVEMENT)
                INNER JOIN (
                        (SELECT performer, when, sequ, songtitle title, songwriter writer FROM PERFORMANCES)
                        JOIN (select title, writer, cowriter from SONGS)
                        USING (title, writer)
```

```
                    )
                    ON performer = band AND (musician = writer OR musician = cowriter)
                    )
          )
GROUP BY performer)
JOIN (
                    SELECT performer, COUNT('x') total
                    FROM PERFORMANCES
                    GROUP BY performer
          ) USING(performer)
)
) ON name = performer;
```

- **Testing:**

Action	**Result obtained**
Ratio of performer my_group (since there is not a existing data in the database with reasonable size to be tested, we decided to defined our own test)	

```
SQL> SELECT name, COALESCE(ratio, 0) from Performers LEFT OUTER JOIN (
  2  SELECT performer, ROUND(counts/total, 2) as ratio FROM (
  3  (SELECT performer, COUNT('x') counts
  4  FROM (
  5  SELECT distinct performer, when, sequ FROM (
  6  (SELECT band, musician FROM INVOLVEMENT)
  7  INNER JOIN (
  8  (SELECT performer, when, sequ, songtitle title, songwriter writer FROM PERFOR
  9  JOIN (select title, writer, cowriter from SONGS)
 10  USING (title, writer)
 11  )
 12  ON performer = band AND (musician = writer OR musician = cowriter)
 13  )
 14  )
 15  GROUP BY performer)
 16  JOIN (
 17  SELECT performer, COUNT('x') total
 18  FROM PERFORMANCES
 19  GROUP BY performer
 20  ) USING(performer)
 21  )
 22  ) ON name = performer
 23  where performer = 'My_group';

NAME                                               COALESCE(RATIO,0)
-------------------------------------------------- -----------------
My_group                                                         ,75
```

Expected Result: 3/4 = 0.75 (worked as expected)
Reasoning:

First, we design our test:
insert into Performers Values('My_group', 'Spanish', 'Spanish');

INSERT INTO Involvement Values('My_group', 'US>>0279650153', 'role1', TO_DATE('29/03/23', 'DD/MM/YY'), Null);
insert into Involvement Values('My_group',

'US>>0184635694', 'role2', TO_DATE('29/03/23', 'DD/MM/YY'), Null);
insert into Concerts Values('My_group',

TO_DATE('29/03/23', 'DD/MM/YY'), Null, 'Madrid', 'Madrid', 'Spain', 0, Null, 555963189);

insert into Performances Values('My_group', TO_DATE('29/03/23', 'DD/MM/YY'), 1, 'Hermitages', 'US>>0255528753',Null);

```
SQL> select writer, cowriter from songs where title = 'Hermitages' and writer = 'US>>0255528753';

WRITER         COWRITER
-------------- --------------
US>>0255528753 US>>0279650153
```

insert into Performances Values('My_group', TO_DATE('29/03/23', 'DD/MM/YY'), 2, 'Station', 'US>>0184635694',Null);

```
SQL> select writer, cowriter from songs where title = 'Station' and writer = 'US>>0184635694';

WRITER         COWRITER
-------------- --------------
US>>0184635694 US>>0279650153
```

insert into Performances Values('My_group', TO_DATE('29/03/23', 'DD/MM/YY'), 3, 'Feeling smart', 'US>>0279650153', Null);

```
SQL> select writer, cowriter from songs where title = 'Feeling smart' and writer = 'US>>0279650153';

WRITER         COWRITER
-------------- --------------
US>>0279650153
```

insert into Performances Values('My_group', TO_DATE('29/03/23', 'DD/MM/YY'), 4, 'Cap', 'SE>>0032567020',Null);

```
SQL> select writer, cowriter from songs where title = 'Cap' and writer = 'SE>>0032567020';

WRITER         COWRITER
-------------- --------------
SE>>0032567020
```

As we can observe, it is a band with 2 members and it has 1 concert with 4 performances, where:

- **the first performance song is written by one of the members as cowriter with another musician**
- **the second performance song is written by both members**
- **the third performance song is written by one member as writer**
- **the fourth performance song is written by other musicians**

We we want a single query for % of tracks and % of performances, we just have to define an alias for both and join them by the performer, we prefer to do it separately so it is more readable and easy to understand.

1.1.2 Revival:

- **Relational algebra:**

Performers_join ≡ (π (name AS performer) **Performers)** ✳ (performer) **(π** (PAIR, performer) **ALBUMS))**

Albums_join ≡ (π (PAIR, title, writer, rec_date) **Tracks)** ✳ (PAIR) **Performers_join**

Tracks_join ≡ σ (rec_date < when) **((π** (performer, songtitle as title, songwriter as writer, when, sequ) **Performances)** ✳ (performer, title, writer) **Albums_join)**

Counts ≡ π performer, when, sequ, min(rec_date) rec_date **Ǵ** (performer, when, sequ) **Tracks_join**

GroupedCounts ≡ π performer, count('x') as counts, avg(when -rec_date) as time **Ǵ** (performer) **Counts**

Total ≡ π performer, count('x') as total **Ǵ** (performer) **Performances**

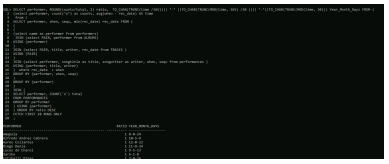
Ratio ≡ π performer, counts/total, time(YY-MM-DD) **σ** (first 10 rows) **ORDER BY** DESC **(GroupedCounts)** ✳ (performer) **(Total))**

- **SQL Implementation:**

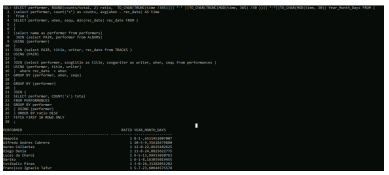
Implicit semantic comment: In case there are several recorded songs -> we take the one recorded at the very beginning:

```
SELECT performer, ROUND(counts/total, 2) ratio,  TO_CHAR(TRUNC(time /365))|| '-' ||TO_CHAR(TRUNC(MOD(time, 365) /30 ))||
'-'||TO_CHAR(MOD(time, 30)) Year_Month_Days FROM (
        (select performer, count('x') as counts, avg(when - rec_date) AS time
         from (
                SELECT performer, when, sequ, min(rec_date) rec_date FROM (
                (
                        (
                                (select name as performer from performers)
                                JOIN (select PAIR, performer from ALBUMS)
                                USING (performer)
                        )
                        JOIN (select PAIR, title, writer, rec_date from TRACKS )
                        USING (PAIR)
                )
                JOIN (select performer, songtitle as title, songwriter as writer, when, sequ from performances )
                USING (performer, title, writer)
                )  where rec_date  < when
                GROUP BY (performer, when, sequ)
        )
        GROUP BY (performer)
)
JOIN (
        SELECT performer, COUNT('x') total
                FROM PERFORMANCES
        GROUP BY performer
) USING (performer)
) ORDER BY ratio DESC
FETCH FIRST 10 ROWS ONLY;
```

- **Testing:**

Action	Expected result	Obtained result
We change the record day of a recorded song in the album owned by the performer 'Amapola'	**the age increases**	**Before** 

After



Process:

First, we look for a recorded song performed by Amapola and recorded:

```
select rec_date, PAIR, SEQU, writer from (
(
        (
                (select name as performer from performers)
                JOIN (select PAIR, performer from ALBUMS)
                USING (performer)
        )
        JOIN (select PAIR, title, writer, rec_date, Sequ from TRACKS )
        USING (PAIR)
)
JOIN (select performer, songtitle as title, songwriter as writer, when from performances )
USING (performer, title, writer)
)  where rec_date  < when and performer = 'Amapola';
```

```
REC_DATE PAIR                 SEQU WRITER
-------- --------------- ---------- --------------
01/10/17 V9731UDI1111EQ8         14 SE>>0572037422
23/10/89 T8035OSF7829PWT          1 SE>>0866705629
26/01/83 Q5168ZMK7665S4P          2 SE>>0870229861
05/05/82 G5957P9T4471XQG          6 SE>>0870229861
11/03/02 O17930L74069FYA          6 SE>>0059303488
25/01/86 S1203J2Y7215PJT          2 DE>>0030666064
06/08/85 D112354W7082A4X          3 DE>>0030666064
11/07/92 V769467U7356CNS          1 SE>>0845920845
27/04/91 K8380NBP995Y8I           1 SE>>0845920845
16/12/11 K3433AKA6089TXY          8 DE>>0284185302
04/05/16 F80852DR6537381          9 DE>>0284185302

REC_DATE PAIR                 SEQU WRITER
-------- --------------- ---------- --------------
25/01/86 S1203J2Y7215PJT          2 DE>>0030666064
06/08/85 D112354W7082A4X          3 DE>>0030666064
13/04/99 R6223IMS7982JSV          5 SE>>0529934667
26/09/17 V9731UDI1111EQ8          2 SE>>0880748275

4525 filas seleccionadas.
```

If we take the last one and change it rec_date using:

```
UPDATE Tracks
SET rec_date = TO_DATE('29/01/01', 'DD/MM/YY')
WHERE PAIR= 'V9731UDI1111EQ8'and SEQU = 2;
```

```
SQL> UPDATE Tracks
  2  SET rec_date = TO_DATE('29/01/01', 'DD/MM/YY')
  3  WHERE PAIR= 'V9731UDI1111EQ8'and SEQU = 2;

1 fila actualizada.
```

1.2 OPERATIVITY:

Package 'melopack':

```
Unset
-- ----------------------------------------
-- Package header
-- ----------------------------------------
create or replace package melopack as
    -- Define the public variable for current performer, identified by its name, of type
varchar2 and with 50-character limit.
    current_performer varchar2(50);

-- ----------------------------------------
-- Assign procedure
-- ----------------------------------------
    -- Define the procedure for assigning a value to the variable.
    PROCEDURE assign(name_in varchar2);

-- ----------------------------------------
-- Insert into album procedure
-- ----------------------------------------
    PROCEDURE create_album_track(
  p_pair IN CHAR,
  p_format IN CHAR,
  p_title IN VARCHAR2,
  p_rel_date IN DATE,
  p_publisher IN VARCHAR2,
  p_manager IN NUMBER,
  p_sequ IN NUMBER,
  p_track_title IN VARCHAR2,
  p_writer IN CHAR,
  p_duration IN NUMBER,
  p_rec_date IN DATE,
  p_studio IN VARCHAR2,
  p_engineer IN VARCHAR2
);

-- ----------------------------------------
-- Delete track from album procedure
-- ----------------------------------------
    -- Define the procedure for deleting a track from an album.
    PROCEDURE delete_track(
  p_pair IN CHAR,
```

```
  p_sequ IN NUMBER
  );


-- ----------------------------------------
-- Report statistics
-- ----------------------------------------
    -- PROCEDURE 3 GOES HERE

  -- Define the procedure for reporting statistics about the current performer and its
collaborators.
  PROCEDURE performer_report;

-- ----------------------------------------
-- Nullary function that returns the value
-- ----------------------------------------
    FUNCTION active RETURN varchar2;

end melopack;
/



-- ----------------------------------------
-- Package body
-- ----------------------------------------
create or replace package body melopack as

-- ----------------------------------------
-- Assign procedure (assigns a value to the variable)
-- ----------------------------------------
PROCEDURE assign(name_in varchar2) IS
BEGIN
    current_performer := name_in;
    DBMS_OUTPUT.PUT_LINE('Current performer is ' || current_performer);
END assign;

-- ----------------------------------------
-- Insert into album procedure
-- ----------------------------------------
PROCEDURE create_album_track(
  p_pair IN CHAR,
  p_format IN CHAR,
  p_title IN VARCHAR2,
  p_rel_date IN DATE,
  p_publisher IN VARCHAR2,
  p_manager IN NUMBER,
  p_sequ IN NUMBER,
  p_track_title IN VARCHAR2,
  p_writer IN CHAR,
  p_duration IN NUMBER,
  p_rec_date IN DATE,
  p_studio IN VARCHAR2,
  p_engineer IN VARCHAR2
)
```

```
IS
  v_album_count NUMBER;
BEGIN
  -- Check if album already exists
  SELECT COUNT('x') INTO v_album_count FROM ALBUMS WHERE PAIR = p_pair;

  IF v_album_count > 0 THEN
    -- Album already exists, insert new track only
    INSERT INTO TRACKS (PAIR, sequ, title, writer, duration, rec_date, studio, engineer)
    VALUES (p_pair, p_sequ, p_track_title, p_writer, p_duration, p_rec_date, p_studio,
p_engineer);
     -- Print success message to console
    DBMS_OUTPUT.PUT_LINE('New track created successfully into a existing Album!');

  ELSE
    -- Album does not exist, insert new album and new track
    INSERT INTO ALBUMS (PAIR, performer, format, title, rel_date, publisher, manager)
    VALUES (p_pair, current_performer, p_format, p_title, p_rel_date, p_publisher,
p_manager);

    INSERT INTO TRACKS (PAIR, sequ, title, writer, duration, rec_date, studio, engineer)
    VALUES (p_pair, p_sequ, p_track_title, p_writer, p_duration, p_rec_date, p_studio,
p_engineer);
    -- Print success message to console
    DBMS_OUTPUT.PUT_LINE('New album and track created successfully!');

  END IF;
END create_album_track;



-- ----------------------------------------
-- Delete track from album procedure
-- ----------------------------------------
PROCEDURE delete_track(
  p_pair IN CHAR,
  p_sequ IN NUMBER
)

IS
  v_track_count NUMBER;
BEGIN
  -- Check if track exists in the album
  SELECT COUNT('x') INTO v_track_count FROM TRACKS WHERE PAIR = p_pair AND sequ =
p_sequ;

  IF v_track_count > 0 THEN
    -- Delete the track
    DELETE FROM TRACKS WHERE PAIR = p_pair AND sequ = p_sequ;

    -- Check if album has any remaining tracks
    SELECT COUNT('x') INTO v_track_count FROM TRACKS WHERE PAIR = p_pair;

    IF v_track_count = 0 THEN
      -- Delete the album since it has no remaining tracks
```

```
      DELETE FROM ALBUMS WHERE PAIR = p_pair;
    END IF;

    -- Print success message to console
    DBMS_OUTPUT.PUT_LINE('Track deleted successfully!');
  ELSE
    -- Print error message to console
    DBMS_OUTPUT.PUT_LINE('Track not found in the album!');
  END IF;
END delete_track;



-- ----------------------------------------
-- Report statistics
-- ----------------------------------------
PROCEDURE performer_report IS

  TYPE album_stats IS RECORD (
    format      ALBUMS.format%TYPE,
    num_albums  NUMBER,
    avg_songs   NUMBER,
    avg_length  NUMBER,
    periodicity NUMBER
  );
  TYPE collaborator_stats IS RECORD (
    name        VARCHAR2(50),
    type        VARCHAR2(25),
    num_works   NUMBER,
    ratio       NUMBER,
    work_type   VARCHAR(25)
  );

  TYPE collaborator_stats_tab IS TABLE OF collaborator_stats INDEX BY BINARY_INTEGER;
  TYPE album_stats_tab IS TABLE OF album_stats INDEX BY BINARY_INTEGER;

  l_stats2 collaborator_stats_tab;
  l_stats2_2 collaborator_stats_tab;

  l_stats album_stats_tab;
BEGIN
  SELECT format,
         COUNT('x') AS num_albums,
         TRUNC(AVG(songs_per_album),2) AS avg_songs,
         TRUNC(AVG(album_length),2) AS avg_length,
         TRUNC(AVG(periodicity)*COUNT('x')/(COUNT('x')-0.9999999999999999),2) AS
periodicity
  BULK COLLECT INTO l_stats
  FROM (
    SELECT a.format,
       a.PAIR,
       COUNT(a.sequ) AS songs_per_album,
       SUM(a.duration) / 60 AS album_length,
       AVG(rel_date - NVL(prev_date, rel_date)) AS periodicity
FROM (
```

```
    SELECT a.PAIR,
           t.SEQU,
           a.rel_date,
           a.format,
           t.duration,
           MAX(a1.rel_date) AS prev_date
    FROM ALBUMS a
    INNER JOIN TRACKS t ON a.PAIR = t.PAIR
    LEFT OUTER JOIN ALBUMS a1 ON a1.performer = a.performer AND a1.rel_date < a.rel_date
AND a1.format = a.format
    WHERE a.performer = current_performer
    GROUP BY a.PAIR, t.SEQU, a.rel_date, a.format, t.duration
) a
GROUP BY a.format, a.PAIR, a.rel_date
  )
  GROUP BY format;

  FOR i IN l_stats.FIRST..l_stats.LAST LOOP
    DBMS_OUTPUT.PUT_LINE('Format: ' || l_stats(i).format ||
                         ', Number of Albums: ' || l_stats(i).num_albums ||
                         ', Average Songs (days): ' || l_stats(i).avg_songs ||
                         ', Average Length (minutes): ' || l_stats(i).avg_length ||
                         ', Periodicity (days): ' || l_stats(i).periodicity);
  END LOOP;


-- Insert statistics about publisher
  SELECT B.publisher,
         'PUBLISHER',
         B.counts,
         TRUNC(ratio, 3),
         'Albums'
  BULK COLLECT INTO l_stats2
  FROM (
    WITH A AS (
      SELECT COUNT('x') AS total
      FROM Albums
      WHERE performer = current_performer
      GROUP BY performer
    )
    SELECT publisher, COUNT('x')/(A.total) AS ratio, COUNT('x') as counts
    FROM Albums, A
    WHERE performer = current_performer
    GROUP BY publisher, A.total
  ) B;

-- Insert statistics about Studios
  SELECT C.studio,
         'STUDIO',
         C.counts,
         TRUNC(ratio, 3),
         'Tracks'
  BULK COLLECT INTO l_stats2_2
  FROM (
    SELECT *
```

```
    FROM (
      WITH A AS (
        SELECT *
        FROM Albums
        JOIN Tracks USING (PAIR)
        WHERE performer = current_performer
      ),
      B AS (
        SELECT COUNT('x') AS total
        FROM A
        GROUP BY performer
      )
      SELECT studio, COUNT('x') as counts, COUNT('x')/(B.total) AS ratio
      FROM A, B
      GROUP BY studio, B.total
    )
  ) C;

  FOR i IN 1..l_stats2_2.COUNT LOOP
    l_stats2(l_stats2.COUNT + 1) := l_stats2_2(i);
  END LOOP;

-- Insert statistics about Engineer
  SELECT C.engineer,
         'ENGINEER',
         C.counts,
         TRUNC(ratio, 3),
         'Tracks'
  BULK COLLECT INTO l_stats2_2
  FROM (
    SELECT *
    FROM (
      WITH A AS (
          SELECT *
          FROM Albums
          JOIN Tracks USING (PAIR)
          WHERE performer = current_performer
        ),
        B AS (
          SELECT COUNT('x') AS total
          FROM A
          GROUP BY performer
        )
        SELECT engineer, COUNT('x') counts, COUNT('x')/(B.total) AS ratio
        FROM A, B
        GROUP BY engineer, B.total
    )
  ) C;

  FOR i IN 1..l_stats2_2.COUNT LOOP
    l_stats2(l_stats2.COUNT + 1) := l_stats2_2(i);
  END LOOP;

-- Insert statistics about Manager-Albums
  SELECT B.name,
```

```
          'MANAGER',
          B.counts,
          TRUNC(ratio, 3),
          'Albums'
  BULK COLLECT INTO l_stats2_2
  FROM (
    WITH A AS (
        SELECT COUNT('x') AS total
        FROM Albums where performer = current_performer
        GROUP BY performer
        )
        SELECT name, manager, COUNT('x') counts, COUNT('x')/(A.total) AS ratio
        FROM Albums JOIN Managers ON manager = mobile, A
        where performer = current_performer
        GROUP BY name, manager, A.total
  ) B;

  FOR i IN 1..l_stats2_2.COUNT LOOP
    l_stats2(l_stats2.COUNT + 1) := l_stats2_2(i);
  END LOOP;

-- Insert statistics about Manager-Concerts
  SELECT B.name,
          'MANAGER',
          B.counts,
          TRUNC(ratio, 3),
          'Concerts'
  BULK COLLECT INTO l_stats2_2
  FROM (
    WITH A AS (
          SELECT COUNT('x') AS total
          FROM Concerts where performer = current_performer
          GROUP BY performer
        )
        SELECT name, manager, COUNT('x') counts, COUNT('x')/(A.total) AS ratio
        FROM Concerts JOIN Managers ON manager = mobile, A
        where performer = current_performer
        GROUP BY name, manager, A.total

  ) B;

  FOR i IN 1..l_stats2_2.COUNT LOOP
    l_stats2(l_stats2.COUNT + 1) := l_stats2_2(i);
  END LOOP;

  -- Iterate through each row in the result set and print values with headers
  FOR i IN 1..l_stats2.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE('Name: ' || l_stats2(i).name || ' Type: ' || l_stats2(i).type
|| ' N Works: ' || l_stats2(i).num_works || ' %works: ' || l_stats2(i).ratio || ' Work
Type: ' || l_stats2(i).work_type);
  END LOOP;

END performer_report;

-- ----------------------------------------
```

```
-- Nullary function that returns the value
-- ---------------------------------------
    FUNCTION active RETURN varchar2 is
    begin
        return current_performer;
    end;

end melopack;
/
```

Procedure 1. Assign:

- **Design:**

Input: a parameters name_in representing the name of the performer to be assigned

Output: a success message printed to the console, indicating the new value of the "current_performer" variable

Description:

- The procedure assigns the value of the "name_in" parameter to the global variable "current_performer" defined in the package
- A success message is printed to the console, indicating the new value of the "current_performer" variable.

- **SQL Implementation:**

Above in the package

- **Testing:**

Action	**Result**

begin
melopack.assign('Hola');
end;
	SQL> begin
	2 melopack.assign('Hola');
	3 end;
	4 /
	Current performer is Hola
	
	Procedimiento PL/SQL terminado correctamente.
	
	SQL> begin DBMS_OUTPUT.PUT_LINE(melopack.current_performer); end;
	2 /
	Hola
	
	Code for checking:

	begin DBMS_OUTPUT.PUT_LINE(melopack.current_performer); end;
Expected: current_performer = 'Hola **Reasoning:** we assigned a new value	

Procedure 2.1. Insert_album_track:

- **Design:**

Input: 12 parameters for defining an album and a track: p_pair, p_format, p_title, p_rel_date, p_publisher, p_manager, p_sequ, p_track_title, p_writer, p_duration, p_rec_date, p_studio, p_engineer

Output: a message in the screen indicating if the album or/and track is created Output. screen indicating if the album or/and track is created Logic

Description:

- The procedure first checks if the album already exists in the database by counting the number of occurrences of p_pair in the ALBUMS table.
- If the album already exists (i.e. v_album_count > 0), a new track is inserted into the TRACKS table with the given parameters and a success message is printed to the console.
- If the album does not exist, a new album is first inserted into the ALBUMS table with the given parameters, and then a new track is inserted into the TRACKS table with the given parameters. A success message is printed to the console.

- **SQL Implementation:**

Above in the package

- **Testing:**

Action	Result

```
begin
melopack.current_performer := 'Almudena Mostorino';
melopack.create_album_track(
'N0000CZU177572V',
'V',
'Girlfriend',
TO_DATE('18/12/69', 'DD/MM/YY'),
'Meloducto Records',
555034841,
1,
'Ad and promise',
'SE>>0575372923',
100,
TO_DATE('18/12/71', 'DD/MM/YY'),
'Jurado Studios',
'Agustin Andres Bardales'
);
end;
```

```
SQL> begin
  2  melopack.current_performer := 'Almudena Mostorino';
  3  melopack.create_album_track(
  4  'N0000CZU177572V',
  5  'V',
  6  'Girlfriend',
  7  TO_DATE('18/12/69', 'DD/MM/YY'),
  8  'Meloducto Records',
  9  555034841,
 10  1,
 11  'Ad and promise',
 12  'SE>>0575372923',
 13  100,
 14  TO_DATE('18/12/71', 'DD/MM/YY'),
 15  'Jurado Studios',
 16  'Agustin Andres Bardales'
 17  );
 18  end;
 19  /
New track created successfully into a existing Album!

Procedimiento PL/SQL terminado correctamente.
```

```
SQL> select pair, sequ, title from Tracks where pair = 'N0000CZU177572V' and sequ = 1;

PAIR                SEQU TITLE
--------------- --------- --------------------------------------------------
N0000CZU177572V         1 Ad and promise
```

Code for checking

select pair, sequ, title from Tracks where pair = 'N0000CZU177572V' and sequ = 1;

Expected: New track created successfully into a existing album

Reasoning: We take data from a existing Album, engineer, studio, manager and song to create a new track using a new sequ number

```
SQL> select a.pair, a.performer, a.format, a.title FROM ALBUMS a INNER JOIN TRACKS b ON(a.PAIR = b.PAIR) fetch first 1 rows only;

PAIR            PERFORMER                                          F TITLE
--------------- -------------------------------------------------- - -----------
N0000CZU177572V Almudena Mostorino                                 V Girlfriend
```

```
SQL> select a.rel_date, a.publisher, a.manager FROM ALBUMS a INNER JOIN TRACKS b ON(a.PAIR = b.PAIR) where a.PAIR = 'N0000CZU177572V';

REL_DATE PUBLISHER                    MANAGER
-------- ------------------------- ----------
18/12/69 Meloducto Records          555034841
```

```
SQL> select engineer from tracks order by engineer asc fetch first 1 rows only;

ENGINEER
--------------------------------------------------
Agustin Andres Bardales
```

```
SQL> select rel_date, publisher, manager from albums where pair = 'N0000CZU177572V';

REL_DATE PUBLISHER                    MANAGER
-------- ------------------------- ----------
18/12/69 Meloducto Records          555034841

SQL> select sequ from tracks where pair = 'N0000CZU177572V';

      SEQU
----------
         2
         3
         4
         5
         6
         7
         8
         9
        10
        11
        12

      SEQU
----------
        13
        14
```

```
SQL> select * from songs fetch first 1 rows only;

TITLE                                              WRITER         COWRITER
-------------------------------------------------- -------------- --------------
Ad and promise                                     SE>>0575372923
```

```
SQL> select * from studios fetch first 1 rows only;

NAME                                               ADDRESS
-------------------------------------------------- ----------------------------
Jurado Studios                                     Street Flood, N 104, VC-77707
```

```
begin
melopack.current_performer := 'Almudena Mostorino';
melopack.create_album_track(
'N0000CZU177572O',
'V',
'Girlfriend',
TO_DATE('18/12/69', 'DD/MM/YY'),
'Meloducto Records',
555034841,
1,
'Ad and promise',
'SE>>0575372923',
100,
TO_DATE('18/12/71', 'DD/MM/YY'),
'Jurado Studios',
'Agustin Andres Bardales'
);
end;
```

```
SQL> begin
  2  melopack.current_performer := 'Almudena Mostorino';
  3  melopack.create_album_track(
  4  'N0000CZU177572O',
  5  'V',
  6  'Girlfriend',
  7  TO_DATE('18/12/69', 'DD/MM/YY'),
  8  'Meloducto Records',
  9  555034841,
 10  1,
 11  'Ad and promise',
 12  'SE>>0575372923',
 13  100,
 14  TO_DATE('18/12/71', 'DD/MM/YY'),
 15  'Jurado Studios',
 16  'Agustin Andres Bardales'
 17  );
 18  end;
 19  /
New album and track created successfully!

Procedimiento PL/SQL terminado correctamente.

SQL> select pair, performer, format from albums where pair = 'N0000CZU177572O';

PAIR            PERFORMER                                          F
--------------- -------------------------------------------------- -
N0000CZU177572O Almudena Mostorino                                 V

SQL> select pair, sequ, title from tracks where pair = 'N0000CZU177572O';

PAIR                 SEQU TITLE
--------------- ---------- --------------------------------------------------
N0000CZU177572O          1 Ad and promise
```

Code for checking

select pair, performer, format from albums where pair = 'N0000CZU177572O';

select pair, sequ, title from tracks where pair = 'N0000CZU177572O';

Expected: New track and album created

Reasoning: We use the same data of the previous test besides a new PAIR to create a new album and a track

```
SQL> select pair, performer from albums where pair = 'N0000CZU177572O';

ninguna fila seleccionada
```

Procedure 2.2. delete_track:

- **Design:**

Input: two input parameters: p_pair representing PAIR and p_sequ representing the track sequence number.

Output: success or error message printed to the console,

Description:

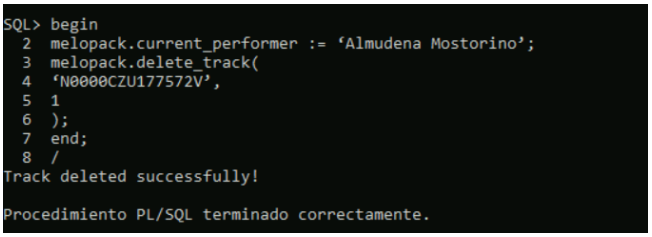
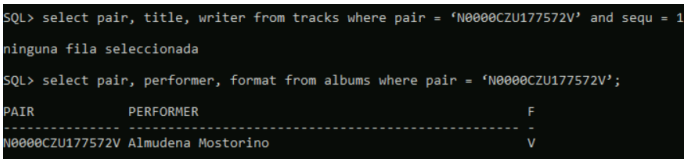
- The procedure first checks if the track exists in the database by counting the number of occurrences using p_pair and p_sequ in the TRACKS table.
- If the track exists (i.e. v_track_count > 0), it is deleted from the TRACKS table.
- The procedure then checks if the album has any remaining tracks by counting the number of occurrences of p_pair in the TRACKS table.
- If the album has no remaining tracks (i.e. v_track_count = 0), it is deleted from the ALBUMS table.
- A success message is printed to the console indicating whether the track was deleted successfully or not.

- **SQL Implementation:**

Above in the package

- **Testing:**

Action	**Result**

begin
 	
melopack.current_performer := 'Almudena Mostorino';
 	
melopack.delete_track(
 	
'N0000CZU177572V',
 	
1
 	
);
 	
end;
	
	SQL> begin
	2 melopack.current_performer := 'Almudena Mostorino';
	3 melopack.delete_track(
	4 'N0000CZU177572V',
	5 1
	6);
	7 end;
	8 /
	Track deleted successfully!
	
	Procedimiento PL/SQL terminado correctamente.
	
	 
	SQL> select pair, title, writer from tracks where pair = 'N0000CZU177572V' and sequ = 1
	
	ninguna fila seleccionada
	
	SQL> select pair, performer, format from albums where pair = 'N0000CZU177572V';
	
	PAIR PERFORMER F
	--- --- -
	N0000CZU177572V Almudena Mostorino V
	
	Code for checking:
	select pair, title, writer from tracks where pair = 'N0000CZU177572V' and sequ = 1;
	
	select pair, performer, format from albums where pair = 'N0000CZU177572V';

Expected: Track deleted successfully and the album still exists **Reasoning:** We delete the track inserted in the first test of the previous procedure. The album has more than 1 track.	

```
SQL> select pair, title, writer from tracks where pair = 'N0000CZU177572V';

PAIR           TITLE                                             WRITER
-------------- ------------------------------------------------- -------------
N0000CZU177572V Music of list                                    SE>>083851262
N0000CZU177572V Evenings                                         SE>>093487226
N0000CZU177572V Delirium gate                                    SE>>064036543
N0000CZU177572V Ad and promise                                   SE>>057537292
N0000CZU177572V Hermitages                                       ES>>003174819
N0000CZU177572V History of success                               SE>>037671043
N0000CZU177572V Loving                                           ES>>000223932
N0000CZU177572V Waitress and rock                                SE>>051420987
N0000CZU177572V Universe of holidays                             SE>>086160223
N0000CZU177572V Trimph                                           SE>>034708045
N0000CZU177572V Firs                                             ES>>000223932

PAIR           TITLE                                             WRITER
-------------- ------------------------------------------------- -------------
N0000CZU177572V Armor                                            SE>>016038345
N0000CZU177572V Grace or nice                                    SE>>008665713
N0000CZU177572V Church                                           SE>>046762880

14 filas seleccionadas.
```

```
begin

melopack.current_performer := 'Almudena Mostorino';

melopack.delete_track(

'N0000CZU177572O',

1

);

end;
```

```
SQL> begin
  2  melopack.current_performer := 'Almudena Mostorino';
  3  melopack.delete_track(
  4  'N0000CZU177572O',
  5  1
  6  );
  7  end;
  8  /
Track deleted successfully!

Procedimiento PL/SQL terminado correctamente.
```

```
SQL> select pair, title, writer from tracks where pair = 'N0000CZU177572O';

ninguna fila seleccionada

SQL> select pair, performer from albums where pair = 'N0000CZU177572O';

ninguna fila seleccionada
```

Code for checking:
select pair, title, writer from tracks where pair = 'N0000CZU177572O';

select pair, performer from albums where pair = 'N0000CZU177572O';

Expected: Track deleted successfully and the album also
Reasoning: We delete the track inserted in the second test of the previous procedure. The album only has 1 track.

Procedure 3. performer_report:

- **Design:**

Input: no parameter for input. However, the current_performer variable define inside the package is used to generate the report.

Output: a report of the performer:

- First line containing the statistics about Album and Tracks
- Following lines about collaborators

Description:

- Initialize variables (using IS RECORD and IS TABLE) to hold the statistics data like a table
- Define a query to retrieve statistics about Albums and Tracks and store them in the variable defined
- Print the content of the variable
- For each type of collaborator:
    - Define a query to retrieve statistics and store them in the variable defined
- Print the content of the variable row by row

- **Relational algebra:**

`For Album statistics:`

C ≡ σ(a.performer = current_performer) **Albums a Θ**(a.PAIR = t.PAIR) **Tracks t)]***(a1.performer = a.performer AND a1.rel_date < a.rel_date AND a1.format = a.format) **Albums a1**

B ≡ π (a.PAIR, t.SEQU, a.rel_date, a.format, t.duration, MAX(a1.rel_date) AS prev_date) **Ǵ** (a.PAIR, t.SEQU, a.rel_date, a.format, t.duration) **C**

A ≡ π (a.format, a.PAIR, COUNT(a.sequ) song_per_album, SUM(a.duration)/60 album_length, AVG(rel_date - NVL(prev_date, rel_date)) AS periodicity) **Ǵ** (a.format, a.PAIR, a.rel_date) **B** AS a

Results ≡ π(COUNT(x),AVG(songs_per_album),AVG(album_length),AVG(periodicity)*COUNT('x')/(COUNT('x')-0.99))) **Ǵ** (performer) **(A)**

`For publisher works on Albums`

A ≡ σ(performer = current_performer) **π** (COUNT('x') AS total) **Ǵ** (performer) **Albums**

B ≡ σ(performer = current_performer) **π** (publisher, COUNT('x')/(A.total) AS ratio, COUNT('x') as counts) **Ǵ** (publisher, A.total) **Albums x A**

Results ≡ π(B.publisher, 'PUBLISHER', B.counts, TRUNC(ratio, 3), 'Albums') **B**

`For Studios works on Tracks`

A ≡ σ(performer = current_performer) **Albums ***(PAIR) **Tracks**

B ≡ π (COUNT('x') AS total) **Ǵ** (performer) **A**

C ≡ π (studio, COUNT('x') as counts, COUNT('x')/(B.total) AS ratio) **Ǵ** (studio, B.total) **A x B**

Results ≡ π(C.studio,'STUDIO', C.counts, TRUNC(ratio, 3), 'Tracks') **C**

`For Engineer works on Tracks`

A ≡ σ(performer = current_performer) **Albums ***(PAIR) **Tracks**

B ≡ π (COUNT('x') AS total) $\mathcal{G}$ (performer) **A**

C ≡ π (engineer, COUNT('x') as counts, COUNT('x')/(B.total) AS ratio) $\mathcal{G}$ (engineer, B.total) **A x B**

Results ≡ π(C.engineer,'ENGINEER', C.counts, TRUNC(ratio, 3), 'Tracks') **C**

`For Managers works on Albums`

A ≡ σ(performer = current_performer) π (COUNT('x') AS total) $\mathcal{G}$ (performer) **Albums**

B ≡ σ(performer = current_performer) π (name, manager, COUNT('x') counts, COUNT('x')/(A.total) AS ratio) $\mathcal{G}$ (name, manager, A.total) **Albums** Θ(manager = mobile) **A**

Results ≡ π(B.name, 'MANAGER', B.counts, TRUNC(ratio, 3), 'Albums') **B**

`For Managers works on Concerts`

A ≡ σ(performer = current_performer) π (COUNT('x') AS total) $\mathcal{G}$ (performer) **Albums**

B ≡ σ(performer = current_performer) π (name, manager, COUNT('x') counts, COUNT('x')/(A.total) AS ratio) $\mathcal{G}$ (name, manager, A.total) **Concerts** Θ(manager = mobile) **Managers x A**

Results ≡ π(B.name,'MANAGER',B.counts,TRUNC(ratio, 3),'Concerts') **B**

- **SQL Implementation:**

Above in the package

- **Testing:**

Action	Result

Code to check report
 	
begin
 	
melopack.assign('Maria Pulido');
 	
melopack.performer_report;
 	
end;
 	
Code to insert new album/track
 	
begin
 	
melopack.current_performer := 'Maria Pulido';
 	
melopack.create_album_track(
 	
'N0000CZU177572I',
	Before
	
	SQL> begin
	2 melopack.assign('Maria Pulido');
	3 melopack.performer_report;
	4 end;
	5 /
	Current performer is Maria Pulido
	Format: V, Number of Albums: 1, Average Songs (days): 14, Average Length (minutes): 67,3, Periodicity (days): 0
	Format: S, Number of Albums: 1, Average Songs (days): 2, Average Length (minutes): 9,53, Periodicity (days): 0
	Name: Scorpio Type: PUBLISHER N Works: 2 %works: 1 Work Type: Albums
	Name: Rudito Inc. Type: STUDIO N Works: 16 %works: 1 Work Type: Tracks
	Name: Rudi Perez Type: ENGINEER N Works: 16 %works: 1 Work Type: Tracks
	Name: Rosa Alfonsina Type: MANAGER N Works: 2 %works: 1 Work Type: Albums
	
	Procedimiento PL/SQL terminado correctamente.

'V',

'Girlfriend',

TO_DATE('18/12/2000', 'DD/MM/YYYY'),

'Scorpio',

555034841,

1,

'Ad and promise',

'SE>>0575372923',

100,

TO_DATE('18/12/2000', 'DD/MM/YYYY'),

'Jurado Studios',

'Agustin Andres Bardales'

);

end;

```
SQL> begin
  2  melopack.current_performer := 'Maria Pulido';
  3  melopack.create_album_track(
  4  'N0000CZU177572I',
  5  'V',
  6  'Girlfriend',
  7  TO_DATE('18/12/2000', 'DD/MM/YYYY'),
  8  'Scorpio',
  9  555034841,
 10  1,
 11  'Ad and promise',
 12  'SE>>0575372923',
 13  100,
 14  TO_DATE('18/12/2000', 'DD/MM/YYYY'),
 15  'Jurado Studios',
 16  'Agustin Andres Bardales'
 17  );
 18  end;
 19  /
New album and track created successfully!

Procedimiento PL/SQL terminado correctamente.
```

After

```
SQL> begin
  2  melopack.assign('Maria Pulido');
  3  melopack.performer_report;
  4   end;
  5  /
Current performer is Maria Pulido
Format: V, Number of Albums: 2, Average Songs (days): 7,5, Average Length (minutes): 34,48, Periodicity (days): 6821,99
Format: S, Number of Albums: 1, Average Songs (days): 2, Average Length (minutes): 9,53, Periodicity (days): 0
Name: Scorpio Type: PUBLISHER N Works: 3 %works: 1 Work Type: Albums
Name: Jurado Studios Type: STUDIO N Works: 1 %works: ,058 Work Type: Tracks
Name: Rudito Inc. Type: STUDIO N Works: 16 %works: ,941 Work Type: Tracks
Name: Rudi Perez Type: ENGINEER N Works: 16 %works: ,941 Work Type: Tracks
Name: Agustin Andres Bardales Type: ENGINEER N Works: 1 %works: ,058 Work Type: Tracks
Name: Esther "Esthi" Type: MANAGER N Works: 1 %works: ,333 Work Type: Albums
Name: Rosa Alfonsina Type: MANAGER N Works: 2 %works: ,666 Work Type: Albums

Procedimiento PL/SQL terminado correctamente.
```

Row	Before/After
1	Before: Format: V, Number of Albums: 1, Average Songs (days): 14, Average Length (minutes): 67,3, Periodicity (days): 0 After: Format: V, Number of Albums: 2, Average Songs (days): 7,5, Average Length (minutes): 34,48, Periodicity (days): 6821,99 Reasoning: - the number of Albums went from 1 to 2, since we added a new Album of format V (worked as expected). - the average Songs went from 14 to 7.5, previously we have only 1 album so 14 is the total songs. Thus (14 + 1) / 2 = 7.5(worked as expected). - average length = (67,3 + 100/60 (length of the new track added))/2 = 34,48 (worked as expected).

	- Periodicity was previously 0 because there is only 1 album of this format. So we can't calculate the average time between two albums of this format because there is only 1. But after the new album is added, the periodicity will be in this case the difference in days between the rel_date of the first album and the added one. We see 15/04/1982 - 18/12/2000 (this is not possible, but for this exercise we do not consider it) is approximately 18.5 years so the result is round **6.752,5**, which is pretty close to 6821,99 (worked as expected).
	`SQL> select pair, format, to_char(rel_date, 'YYYY-MM-DD') from albums where performer = 'Maria Pulido';`
	`PAIR F TO_CHAR(RE`
	`---------- - ----------`
	`I8376HLG1644MCG S 1982-06-17`
	`H261457M7709BPV V 1982-04-15`
	`N0000CZU177572I V 2000-12-18`

2	The second row remains the same (worked as expected)
3	Before: Name: Scorpio Type: PUBLISHER N Works: 2 %works: 1 Work Type: Albums After: Name: Scorpio Type: PUBLISHER N Works: 3 %works: 1 Work Type: Albums The new album is done with Publisher 'Scorpio' (worked as expected)
4º in the report after	Before: nothing After: Name: Jurado Studios Type: STUDIO N Works: 1 %works: ,058 Work Type: Tracks A new studio is added due to the last track. (worked as expected)
4º before and 5' after	Before: Name: Rudito Inc. Type: STUDIO N Works: 16 %works: 1 Work Type: Tracks After: Name: Rudito Inc. Type: STUDIO N Works: 16 %works: ,941 Work Type: Tracks the %Works dropped 1/17 * 100% due to the new studio (worked as expected)
6º after	Before: nothing

	After: Name: Agustin Andres Bardales Type: ENGINEER N Works: 1 %works: ,058 Work Type: Tracks New engineer is added (worked as expected)
5º before and 7º after	Before: Name: Rudi Perez Type: ENGINEER N Works: 16 %works: 1 Work Type: Tracks After Name: Rudi Perez Type: ENGINEER N Works: 16 %works: ,941 Work Type: Tracks Percentage of previous engineer dropped 1/17 also(worked as expected)
8º after	Before: nothing After: Name: Rudi Perez Type: ENGINEER N Works: 16 %works: 1 Work Type: Tracks New manager is added (worked as expected)
6% before and 9º after	Before: Name: Rosa Alfonsina Type: MANAGER N Works: 2 %works: 1 Work Type: Albums After: Name: Rosa Alfonsina Type: MANAGER N Works: 2 %works: ,666 Work Type: Albums Percetange dropped ⅓ due to the new manager.(worked as expected)

Code to check report

```
begin
melopack.assign('Maria Pulido');
melopack.performer_report;
 end;
```

Code to delete album/track

```
begin
melopack.assign('Maria Pulido');
melopack.delete_track(
'N0000CZU177572I',
1); end;
```

```
SQL> begin
  2  melopack.assign('Maria Pulido');
  3  melopack.delete_track(
  4  'N0000CZU177572I',
  5  1); end;
  6  /
Current performer is Maria Pulido
Track deleted successfully!

Procedimiento PL/SQL terminado correctamente.

SQL> begin
  2  melopack.assign('Maria Pulido');
  3  melopack.performer_report;
  4   end;
  5  /
Current performer is Maria Pulido
Format: V, Number of Albums: 1, Average Songs (days): 14, Average Length (minutes): 67,3, Periodicity (days): 0
Format: S, Number of Albums: 1, Average Songs (days): 2, Average Length (minutes): 9,53, Periodicity (days): 0
Name: Scorpio Type: PUBLISHER N Works: 2 %works: 1 Work Type: Albums
Name: Rudito Inc. Type: STUDIO N Works: 16 %works: 1 Work Type: Tracks
Name: Rudi Perez Type: ENGINEER N Works: 16 %works: 1 Work Type: Tracks
Name: Rosa Alfonsina Type: MANAGER N Works: 2 %works: 1 Work Type: Albums

Procedimiento PL/SQL terminado correctamente.
```

We get our original result before doing the insertion of the new track. (worked as expected)

1.3 EXTERNAL DESIGN

1.3.1 First view

- **Relational algebra:**

my_albums ≡ π (PAIR, SUM(T.duration) as album_duration) **Ǧ** (PAIR) **σ**(A.performer = current_performer) (**Albums A** ✳ (PAIR) **Tracks T)**

- **SQL Implementation:**

```
Unset
-- ----------------------------------------
-- my_albums view (read only)
-- ----------------------------------------
CREATE OR REPLACE VIEW my_albums AS (
      SELECT PAIR, SUM(T.duration) as album_duration
      FROM ALBUMS A
      JOIN TRACKS T USING (PAIR)
      WHERE A.performer = melopack.active
      GROUP BY (PAIR)
) WITH READ ONLY;
```

- **Testing:**

Action	Result
Set current_performer: begin melopack.assign('Vigilia'); end; **Get view:** select * from my_albums; **Insert track:** insert into TRACKS(PAIR, sequ, title, writer, rec_date, engineer, duration) values('W68348XD82327T4', 3, 'Absence',	SQL> begin melopack.assign('Vigilia'); end; 2 / Current performer is Vigilia Procedimiento PL/SQL terminado correctamente. SQL> select * from my_albums; PAIR ALBUM_DURATION --- --- W68348XD82327T4 570 M3631ZNM24148LB 3878 SQL> insert into TRACKS (PAIR, sequ, title, writer, rec_date, engineer, duration) values ('W68348XD82327T4',3,'Absence', 'SE>>0736344207','21/06/71','Charles',147); 1 fila creada.

'SE>>0736344207', '21/06/71', 'Charles', 147);

```
SQL> insert into TRACKS(PAIR, sequ, title, writer, rec_date, engineer, duration) values('W68348XD82327
T4', 3, 'Absence', 'SE>>0736344207', '21/06/71', 'Charles', 147);

1 fila creada.

SQL> select * from my_albums;

PAIR            ALBUM_DURATION
--------------- --------------
W68348XD82327T4            717
M3631ZNM24148L8           3878
```

Reasoning:

After inserting a track into one of the available albums, its duration should be updated by the length of the track inserted. We tried on album with PAIR = 'W68348XD82327T4'.

There are several things we need to do before inserting, such as:

- Look at the amount of tracks already in that album so as not to repeat a sequ value.
- Select a title and writer that exist in table Songs.

After doing so, we now insert a new track into that album:

Expected result: duration of album with PAIR = 'W68348XD82327T4' is increased from 570 to 570 + 147 = 717 (worked as expected)

Set current_performer:

begin melopack.assign('Maria Pulido'); end;

Get view:

select * from my_albums;

delete track:

DELETE FROM TRACKS WHERE PAIR = 'I8376HLG1644MCG' AND sequ = 1;

```
SQL> DELETE FROM TRACKS WHERE PAIR = 'I8376HLG1644MCG' AND sequ = 1;

1 fila suprimida.
```

```
SQL> begin melopack.assign('Maria Pulido'); end;
  2  /
Current performer is Maria Pulido

Procedimiento PL/SQL terminado correctamente.

SQL> select * from my_albums;

PAIR            ALBUM_DURATION
--------------- --------------
I8376HLG1644MCG            225
H261457M7709BPV           4038
```

Reasoning We delete a the album'I8376HLG1644MCG' from the performer Maria Pulido and delete the first track of it

Expected Result: 225 (worked as expected)

```
SQL> select pair from albums where performer = 'Maria Pulido';

PAIR
---------------
I8376HLG1644MCG
H261457M7709BPV

SQL> select pair, sequ, duration from tracks where PAIR = 'I8376HLG1644MCG';

PAIR                 SEQU   DURATION
--------------- ---------- ----------
I8376HLG1644MCG          1        347
I8376HLG1644MCG          2        225
```

Set current_performer:
 	
begin melopack.assign('Maria Pulido'); end;
 	
Insert
 	
insert into my_albums values('I8376HLG1644MCG', 10000);
 	
Update
 	
update my_albums set album_duration = 200 where PAIR = 'I8376HLG1644MCG';
 	
Delete
 	
delete from my_albums where PAIR = 'I8376HLG1644MCG';
	SQL> insert into my_albums values('I8376HLG1644MCG', 10000);
	insert into my_albums values('I8376HLG1644MCG', 10000)
	*
	ERROR en linea 1:
	ORA-42399: no se puede realizar una operacion DML en una vista de solo lectura
	
	SQL> update my_albums set album_duration = 200 where PAIR = 'I8376HLG1644MCG';
	update my_albums set album_duration = 200 where PAIR = 'I8376HLG1644MCG'
	*
	ERROR en linea 1:
	ORA-01732: operacion de manipulacion de datos no valida en esta vista
	
	SQL> delete from my_albums where PAIR = 'I8376HLG1644MCG';
	delete from my_albums where PAIR = 'I8376HLG1644MCG'
	*
	ERROR en linea 1:
	ORA-01732: operacion de manipulacion de datos no valida en esta vista

Reasoning We try to insert/update/delete a row in a read only view. **Expected Result:** Error (worked as expected)	

1.3.2 Second view

- **Relational algebra:**

A ≡ π(TO_CHAR(when, 'YYYY-MM') when, COUNT('x') AS num_performances) Ǧ (TO_CHAR(when, 'YYYY-MM')) **PERFORMANCES**

B ≡ π(TO_CHAR(when, 'YYYY-MM') when, COUNT(*) AS attendees) Ǧ (TO_CHAR(when, 'YYYY-MM')) **ATTENDANCES**

Result ≡ π (when, total_concerts, AVG_duration, num_performances/total_concerts AVG_performances, attendees) ⊤(when) **(**π (COUNT('x') total_concerts, AVG(duration) AVG_duration, TO_CHAR(when, 'YYYY-MM') when) Ǧ (TO_CHAR(when, 'YYYY-MM')) **CONCERTS** Θ(when) **A** Θ(when) **B)**

- **SQL Implementation:**

```
Unset
-- ----------------------------------------
-- Events view (read only)
-- ----------------------------------------
CREATE OR REPLACE VIEW events AS
```

```
WITH A AS (SELECT TO_CHAR(when, 'YYYY-MM') when, COUNT('x') AS num_performances FROM PERFORMANCES
        GROUP BY TO_CHAR(when, 'YYYY-MM')
),
B AS (
(SELECT TO_CHAR(when, 'YYYY-MM') when, COUNT(*) AS attendees FROM ATTENDANCES GROUP BY TO_CHAR(when, 'YYYY-MM'))
)
SELECT when, total_concerts, AVG_duration, num_performances/total_concerts AVG_performances, attendees FROM (
(SELECT COUNT('x') total_concerts, AVG(duration) AVG_duration, TO_CHAR(when, 'YYYY-MM') when FROM CONCERTS
GROUP BY TO_CHAR(when, 'YYYY-MM'))
JOIN A USING (when) JOIN B USING (when)
)ORDER BY when ASC
WITH READ ONLY;
```

Implicit semantic comment: we suppose that the same person can go to several concerts in the same day.

Implicit semantic comment: we suppose that the duration of the concert is updated and coherent with the sum of the duration of the performances

Implicit semantic comment: If a concert has not performance, we consider that the concert does not exist.

Implicit semantic comment: If a concert has not attendees, it is not a concert but a music rehearsal.

- **Testing:**

To test this view, we will insert a new concert with one performance, in a specific date, as well as an attendant to that concert, and all the values of the view should change in the following way in the row with the month of the inserted concert:

- new total_concert = previous total_concert + 1
- new total_attendees = previous total_attendees + 1
- new avg_duration = (previous avg_duration * previous total_concert + new performance duration) / new total concert
- new avg_num_performances = (previous avg_num_performances * previous total_concert + new extra performances) / new total_concert

Before each insertion, the values have been checked in order to maintain coherence of foreign relations.

When executing **select * from events** just after creating the view:

```
SQL> select * from events;

WHEN    TOTAL_CONCERTS AVG_DURATION AVG_PERFORMANCES  ATTENDEES
------- -------------- ------------ ---------------- ----------
2009-11             30   150,133333       11,6333333          4
2018-05             51   127,705882       11,6666667       1722
2018-06             77   125,831169       11,7532468       2624
2018-07             88        128,5       11,7727273       2982
2018-08            100       126,82             11,7       3271
2018-09            105   126,209524       11,6857143       3465
2018-10             66        127,5       11,6818182       2217
2018-11             28          125       11,5357143       1017
2019-05             47   125,829787       11,5531915       1641
2019-06             50       129,88            11,64       1703
2019-07             62   125,580645       11,6612903       2136

WHEN    TOTAL_CONCERTS AVG_DURATION AVG_PERFORMANCES  ATTENDEES
------- -------------- ------------ ---------------- ----------
2019-08             82   127,792683       11,6829268       2778
2019-09             67   128,716418       11,6268657       2294
2019-10             50       129,32             11,7       1661
2019-11             17   121,058824       11,3529412        620
2020-05             22   127,454545       11,8181818        772
2020-06             26   126,307692       11,6538462        873
2020-07             26   127,269231       11,6538462        984
2020-08             27   128,259259       11,7407407        945
2020-09             29   130,137931       11,8965517       1021
2020-10             27   128,851852       11,6666667        895

21 filas seleccionadas.
```

Now we insert the following (**notice the trigger for update the concert duration is not implemented here**):

- A concert with one performance and one attendee in 2020-10 (already some concerts)

insert into concerts (performer, manager, when, country, municipality, address, attendance, duration) values ('Lazarte', 555000555, TO_DATE('01/10/2020', 'DD/MM/YYYY'), 'Spain', 'Madrid', 'calle', 1, 40);

insert into attendances (client, performer, when, RFID, purchase) values ('joelpelaez@clients.vinylinc.com', 'Lazarte' , TO_DATE('01/10/2020', 'DD/MM/YYYY'), 1234, '20/09/20');

insert into performances (performer, when, sequ, songtitle, songwriter, duration) values ('Lazarte', TO_DATE('01/10/2020', 'DD/MM/YYYY'), 1, 'Absence', 'SE>>0736344207', 40);

- **A concert with one performance and one attendee in 2020-10 (no concerts yet in this month)**

insert into concerts (performer, manager, when, country, municipality, address, attendance, duration) values ('Lazarte', 555000555, TO_DATE('01/10/2021', 'DD/MM/YYYY'), 'Spain', 'Madrid', 'calle', 1, 40);

insert into attendances (client, performer, when, RFID, purchase) values ('joelpelaez@clients.vinylinc.com', 'Lazarte' ,TO_DATE('01/10/2021', 'DD/MM/YYYY'), 1234, '20/09/20');

insert into performances (performer, when, sequ, songtitle, songwriter, duration) values ('Lazarte', TO_DATE('01/10/2021', 'DD/MM/YYYY'), 1, 'Absence', 'SE>>0736344207', 40);

TO DELETE (for convenience, not part of the test)

delete from attendances where performer = 'Lazarte' and when = TO_DATE('01/10/2021', 'DD/MM/YYYY') and client = 'joelpelaez@clients.vinylinc.com';

delete from performances where performer = 'Lazarte' and when = TO_DATE('01/10/2021', 'DD/MM/YYYY') and sequ = 1;

delete from concerts where performer = 'Lazarte' and when = TO_DATE('01/10/2021', 'DD/MM/YYYY');

delete from attendances where performer = 'Lazarte' and when = TO_DATE('01/10/2020', 'DD/MM/YYYY') and client = 'joelpelaez@clients.vinylinc.com';

delete from performances where performer = 'Lazarte' and when = TO_DATE('01/10/2020', 'DD/MM/YYYY') and sequ = 1;

delete from concerts where performer = 'Lazarte' and when = TO_DATE('01/10/2020', 'DD/MM/YYYY');

```
SQL> insert into concerts (performer, manager, when, country, municipality, address, attendance, duration) values ('Lazarte', 555000555, TO_DATE('01/10/2020', 'DD/MM/YYYY'), 'Spain', 'Madri
d', 'calle', 1, 40);

1 fila creada.

SQL>
SQL> insert into attendances (client, performer, when, RFID, purchase) values ('joelpelaez@clients.vinylinc.com', 'Lazarte' , TO_DATE('01/10/2020', 'DD/MM/YYYY'), 1234, '20/09/20');

1 fila creada.

SQL>
SQL> insert into performances (performer, when, sequ, songtitle, songwriter, duration) values ('Lazarte', TO_DATE('01/10/2020', 'DD/MM/YYYY'), 1, 'Absence', 'SE>>0736344207', 40);

1 fila creada.

SQL> insert into concerts (performer, manager, when, country, municipality, address, attendance, duration) values ('Lazarte', 555000555, TO_DATE('01/10/2021', 'DD/MM/YYYY'), 'Spain', 'Madri
d', 'calle', 1, 40);

1 fila creada.

SQL>
SQL> insert into attendances (client, performer, when, RFID, purchase) values ('joelpelaez@clients.vinylinc.com', 'Lazarte' ,TO_DATE('01/10/2021', 'DD/MM/YYYY'), 1234, '20/09/20');

1 fila creada.

SQL>
SQL> insert into performances (performer, when, sequ, songtitle, songwriter, duration) values ('Lazarte', TO_DATE('01/10/2021', 'DD/MM/YYYY'), 1, 'Absence', 'SE>>0736344207', 40);

1 fila creada.
```

After running again **select * from events**

```
SQL> select * from events
  2  ;

WHEN    TOTAL_CONCERTS AVG_DURATION AVG_PERFORMANCES  ATTENDEES
------- -------------- ------------ ---------------- ----------
2009-11             30   150,133333       11,6333333          4
2018-05             51   127,705882       11,6666667       1722
2018-06             77   125,831169       11,7532468       2624
2018-07             88        128,5       11,7727273       2982
2018-08            100       126,82             11,7       3271
2018-09            105   126,209524       11,6857143       3465
2018-10             66        127,5       11,6818182       2217
2018-11             28          125       11,5357143       1017
2019-05             47   125,829787       11,5531915       1641
2019-06             50       129,88            11,64       1703
2019-07             62   125,580645       11,6612903       2136

WHEN    TOTAL_CONCERTS AVG_DURATION AVG_PERFORMANCES  ATTENDEES
------- -------------- ------------ ---------------- ----------
2019-08             82   127,792683       11,6829268       2778
2019-09             67   128,716418       11,6268657       2294
2019-10             50       129,32             11,7       1661
2019-11             17   121,058824       11,3529412        620
2020-05             22   127,454545       11,8181818        772
2020-06             26   126,307692       11,6538462        873
2020-07             26   127,269231       11,6538462        984
2020-08             27   128,259259       11,7407407        945
2020-09             29   130,137931       11,8965517       1021
2020-10             28   125,678571       11,2857143        896
2021-10              1           40                1          1

22 filas seleccionadas.
```

For 2020-10.

Expected total_concerts: 27 + 1 = 28 (worked as expected)

Expected total_attendees: 895 + 1 = 896 (worked as expected)

Expected avg_duration: (128.85 * 27 + 40)/ 28 = 125,6 (worked as expected)

Expected avg_num_performances: (11,66 * 27 + 1) / 28 = 11,27 (worked as expected)

For 2021-10.

Expected total_concerts: 0 + 1 = 1(worked as expected)

Expected total_attendees: 0 + 1 = 896 (worked as expected)

Expected avg_duration: (0 * 0 + 40)/ 1= 40 (worked as expected)

Expected avg_num_performances: 1 (worked as expected)

- **Insert**

insert into events values(TO_DATE('2020-01', 'YYYY-MM'), 1, 1, 1, 1);

```
SQL> insert into events values(TO_DATE('2020-01', 'YYYY-MM'), 1, 1, 1, 1);
insert into events values(TO_DATE('2020-01', 'YYYY-MM'), 1, 1, 1, 1)
*
ERROR en linea 1:
ORA-01779: no se puede modificar una columna que se corresponde con una tabla
no reservada por clave
```

1.3.3 Third view:

For this view, we have focus on three main aspects:

- Selecting the number of concerts of current performer to check wether they should or should not have fans
- Checking the number of attendances of each attendant to see if they are qualified to be a fan or not
- Get the information of the fans from client table to be displayed.

After that, we created the table banned to store information from clients and their corresponding performer from which they are banned.

One of the parts with which we have had more trouble is with the relationship between the view and its corresponding base tables. We had some problems when trying to test with own created simpler tables, and in the end we have not been able to accomplish some of the requirements from this view. Deleting from view but leaving intact the base tables was especially challenging.

- **Relational algebra:**

fans ≡ π(client, name, surname1, surname2, age) **(**

(σ(count > 1) π(performer, count as num_concerts) Ġ (performer)

(π(performer, when) σ(performer = current_performer)

CONCERTS))

✳ (performer)

(σ(count > 1) π(client, performer, count as num_attend) Ġ (client, performer) σ(performer = current_performer) **ATTENDANCES)**

✳ (client)

(π(email as client, name, surname1, surname2, (currentdate-birthdate)/365.2422) as age **CLIENTS)**

)

- **SQL Implementation:**

```
Unset
-- ----------------------------------------
-- Fans view (full operativity)
-- ----------------------------------------
CREATE OR REPLACE VIEW fans AS (
select client, name, surn1, surn2, age from(
(select performer, count('x') as num_concerts from (
    select performer, when from concerts
    where performer = melopack.active)
    group by performer HAVING count('x') > 1)
    JOIN
      (select client, performer, count('x') as num_atten from
        ATTENDANCES where performer = melopack.active GROUP BY client, performer HAVING
count('x') > 1)
    USING (performer)
    JOIN
       (select e_mail as client, name, surn1, surn2, trunc((sysdate - birthdate) /
365.2422, 0) as age
    from CLIENTS)
     USING(client)
     )
```

```
) WITH CHEK OPTION;

-- ----------------------------------------
-- Additional table that records which clients are banned
-- for which performers
-- ----------------------------------------
CREATE TABLE BANNED (
e_mail      VARCHAR2(100),
performer VARCHAR2(100),
CONSTRAINT PK_BANNED PRIMARY KEY (e_mail),
CONSTRAINT FK_BANNED1 FOREIGN KEY (e_mail) REFERENCES CLIENTS,
CONSTRAINT FK_BANNED2 FOREIGN KEY (performer) REFERENCES PERFORMERS
);
```

- **Testing:**

1.4 Triggers.

1º Trigger

- **Design:**

- **Associated Table**: PERFORMANCES
- **Events**: When there is insertion
- **Temporality**: After
- **Granularity**: Row
- **Condition**:
- **Action**: Create a trigger that uses 'after insert on PERFORMANCES' and updates the column 'duration' in 'CONCERTS' by using SQL's 'UPDATE' statement.

- **SQL:**

```
Unset
create or replace trigger update_duration
```

```
after insert on PERFORMANCES
for each row
begin
    update CONCERTS
    set duration = duration + :new.duration
    where performer = :new.performer
    and when = :new.when;
end update_duration;
/
```

```
Unset
create or replace trigger delete_duration
after delete on PERFORMANCES
for each row
begin
    update CONCERTS
    set duration = duration - :old.duration
    where performer = :old.performer
    and when = :old.when;
end delete_duration;
/
```

We can delete the trigger by using the following command:

```
Unset
drop trigger update_duration;
```

Same code above but with comments (can be ignored):

-- Create or replace the trigger with the label 'update_duration'.

create or replace trigger update_duration

-- The trigger is executed after insertion on the table 'PERFORMANCES'.

after insert on PERFORMANCES

-- Checks every row.

for each row

begin

```
    -- Update the table 'CONCERTS' by adding the new duration for the matching row.
    update CONCERTS
    set duration = duration + :new.duration
    -- If no 'WHERE' clause is included, all records in the table will be updated.
    where performer = :new.performer
    and when = :new.when;
end update_duration;
/
```

- **Testing:**

Originally, there are 76348 rows in 'CONCERTS'.

We observe that there is no null value in the inserted ones in both 'duration' columns for 'CONCERTS' and 'PERFORMANCES', despite the fact that the attribute is optional in both tables.

In order to do the tests correctly, we use the following existing data taken from the first row of 'CONCERTS':

- 'FK_CONCERTS1'/Performer: 'Barba'.
- 'FK_CONCERTS2'/Manager: '555761116'.
- 'FK_CONCERTS3'/Tour: 'Barba', ''09 sauces Tour ', '555761116'
- 'FK_PERFORMANCES1'/Concert for which we have used 'Barba', '07-11-09'.
- 'FK_PERFORMANCES2'/Track: 'Sabia tomillo', 'SE>>0637396189'.

We observe that there is no row in 'performances' where 'performer' is 'Barba', 'when' is '07-11-09' and 'song' is 'Sabia tomillo', so we will use these values in the test cases.

We can reset the duration of the modified concert for testing by using the following command:

```
Unset
update CONCERTS
set duration = 104
where performer = 'Barba' and when = '07-11-09';
```

The following testing cases have been considered:

1. Inserting one performance (worked as expected):

```
Unset
-- ----------------------------------------
-- Checking the duration of the concert
-- ----------------------------------------
select duration from concerts where performer = 'Barba' and when = '07-11-09';
/*
DURATION
----------
104
*/

-- ----------------------------------------
-- Inserting a performance
-- ----------------------------------------
insert into performances values ('Barba', '07-11-09', '999', 'Sabia tomillo',
'SE>>0637396189', '562');
-- 1 row created.

-- ----------------------------------------
-- Checking the duration of the concert after the insertion
-- ----------------------------------------
select duration from concerts where performer = 'Barba' and when = '07-11-09';
/*
DURATION
----------
666
*/
```

```
SQL> select duration from concerts where performer = 'Barba' and when = '07-11-09';

  DURATION
----------
       104

SQL> /*
SQL> DURATION
SQL> ----------
SQL> 104
SQL> */
SQL>
SQL> -- ----------------------------------------
SQL> -- Inserting a performance
SQL> -- ----------------------------------------
SQL> insert into performances values ('Barba', '07-11-09', '999', 'Sabia tomillo', 'SE>>0637396189', '562');

1 fila creada.

SQL> -- 1 row created.
SQL>
SQL> -- ----------------------------------------
SQL> -- Checking the duration of the concert after the insertion
SQL> -- ----------------------------------------
SQL> select duration from concerts where performer = 'Barba' and when = '07-11-09';

  DURATION
----------
       666
```

We delete first the inserted row before doing the second test:

```
Unset
delete from performances where performer = 'Barba' and when = '07-11-09' and sequ =
'999';
```

```
SQL> delete from performances where performer = 'Barba' and when = '07-11-09' and sequ = '999';

1 fila suprimida.
```

2. Inserting multiple performances (worked as expected):

```
Unset
-- ----------------------------------------
-- Checking the duration of the concert
-- ----------------------------------------
select duration from concerts where performer = 'Barba' and when = '07-11-09';
/*
DURATION
----------
104
*/
```

```
-- ----------------------------------------
-- Inserting many performances (104+229+167+277=777)
-- ----------------------------------------
insert all
into performances (performer, when, sequ, songtitle, songwriter, duration) values
('Barba', '07-11-09', '998', 'Sabia tomillo', 'SE>>0637396189', '229')
into performances (performer, when, sequ, songtitle, songwriter, duration) values
('Barba', '07-11-09', '997', 'Sabia tomillo', 'SE>>0637396189', '167')
into performances (performer, when, sequ, songtitle, songwriter, duration) values
('Barba', '07-11-09', '996', 'Sabia tomillo', 'SE>>0637396189', '277')
select 1 from dual;
-- 3 rows created


-- ----------------------------------------
-- Checking the duration of the concert
-- ----------------------------------------
select duration from concerts where performer = 'Barba' and when = '07-11-09';
/*
DURATION
----------
777


If we do not delete the previous test insertion we get ->
*/
```

```
SQL> -- ----------------------------------------
SQL> -- Checking the duration of the concert
SQL> -- ----------------------------------------
SQL> select duration from concerts where performer = 'Barba' and when = '07-11-09';

  DURATION
----------
       104

SQL> /*
SQL> DURATION
SQL> ----------
SQL> 104
SQL> */
SQL>
SQL> -- ----------------------------------------
SQL> -- Inserting many performances (104+229+167+277=777)
SQL> -- ----------------------------------------
SQL> insert all
  2  into performances (performer, when, sequ, songtitle, songwriter, duration) values ('Barba', '07-11-09', '998', 'Sabia tomillo', 'SE>>0637396189', '229')
  3  into performances (performer, when, sequ, songtitle, songwriter, duration) values ('Barba', '07-11-09', '997', 'Sabia tomillo', 'SE>>0637396189', '167')
  4  into performances (performer, when, sequ, songtitle, songwriter, duration) values ('Barba', '07-11-09', '996', 'Sabia tomillo', 'SE>>0637396189', '277')
  5  select 1 from dual;

3 filas creadas.

SQL> -- 3 rows created
SQL>
SQL> -- ----------------------------------------
SQL> -- Checking the duration of the concert
SQL> -- ----------------------------------------
SQL> select duration from concerts where performer = 'Barba' and when = '07-11-09';

  DURATION
----------
       777
```

2° Trigger

- **Design:**

- **Associated Table**: ATTENDANCES
- **Events**: when there is insertion
- **Temporality**: Before
- **Granularity**: Row
- **Condition**: If 'birthdate' column in 'CLIENTS' is smaller than 18
- **Action**: Create a trigger that uses 'before insert on ATTENDANCES' and checks if the 'birthdate' column in 'CLIENTS' is smaller than 18. If the client is underaged, an error is raised.

- **SQL Implementation:**

```
Unset
create or replace trigger reject_age
before insert on ATTENDANCES
for each row
declare client_birth date;
begin
    select birthdate into client_birth
    from CLIENTS where e_mail= :new.client;

    if client_birth is null
        then RAISE_APPLICATION_ERROR(-20001, 'The client birthdate is missing.');
    end if;

    if client_birth > add_months(sysdate, -12*18)
        then RAISE_APPLICATION_ERROR(-20002, 'The client must be of legal age.');
```

```
    end if;

end reject_age;
/
```

We can delete the trigger by using the following command:

```
Unset
drop trigger reject_age;
```

Same code above but with comments (can be ignored):

```
-- Create or replace the trigger with the label 'reject_age'.
create or replace trigger reject_age
-- The trigger is executed before insertion on the table 'ATTENDANCES'.
before insert on ATTENDANCES
-- Checks every row.
for each row
-- Declare the variable 'client_birth' of type 'date'.
declare client_birth date;
begin
   -- Store the query result into the aforementioned variable.
   select birthdate into client_birth
   -- Must use new.client (foreign key).
   from CLIENTS where e_mail= :new.client;

   -- If the birthdate of the client is missing, raise an error.
   if client_birth is null
      then RAISE_APPLICATION_ERROR(-20001, 'The client birthdate is missing.');
   end if;

   -- If the client is underaged, raise a different error.
   if client_birth > add_months(sysdate, -12*18)
```

```
        then RAISE_APPLICATION_ERROR(-20002, 'The client must be of legal age.');
    end if;

end reject_age;
/
```

- **Testing:**

Originally, there are 35621 rows in 'ATTENDANCES'.

In the given schematic design, table 'ATTENDANCES' has a column called 'artist', but the code uses 'performer' instead. Also, 'purchase' is used instead of 'purchasedatetime'.

We observe that there is no null value in the inserted ones in the 'birthdate' column, despite the fact that the attribute is optional.

In order to do the tests correctly, we must respect the constraint:

- 'FK_ATTENDANCES1' of 'ATTENDANCES' that references an existing tuple composed of 'performer' and 'when', for which we have used 'Barba', '07-11-2009' which belongs to the first row of 'CONCERTS' as its rownum is equal to 1.
- The DNI of each client and RFID of every ticket must be unique due to the 'UK_CLIENTS' and 'UK_ATTENDANCES' constraints, respectively.

We can delete every created row for testing by using the following commands:

```
Unset
delete attendances where client like 'example2%';
delete clients where e_mail like 'example2%';
```

The following testing cases have been considered:

1. The client is of legal age: (worked as expected)

```
Unset
-- ----------------------------------------
-- Creating the client of legal age
```

```
-- ----------------------------------------

insert into clients values ('example@example.com', 'John', 'Doe', 'Doe', '01-01-2000', '666666666', 'Street', '12345678');
-- 1 row created.


-- ----------------------------------------

-- Buying the ticket for a valid client

-- ----------------------------------------

insert into attendances values ('example@example.com', 'Barba', '07-11-2009', '4JL4W8F7T6R9D2K0S3P1Q5NMYX9CZ0E7V4B1U8I2O6A3H5G7', '01-01-2023');

-- 1 row created.


-- ----------------------------------------

-- Checking the ticket is correctly created

-- ----------------------------------------

select * from attendances where client = 'example@example.com';

/*
CLIENT                 PERFORMER                 WHEN              RFID                 PURCHASE
----------------------------------------------------------

example@example.com   Barba   07/11/09   4JL4W8F7T6R9D2K0S3P1Q5NMYX9CZ0E7V4B1U8I2O6A3H5G7
01/01/23

*/
```

```
SQL> insert into clients values ('example@example.com', 'John', 'Doe', 'Doe', '01-01-2000', '666666666', 'Street', '12345678');
insert into clients values ('example@example.com', 'John', 'Doe', 'Doe', '01-01-2000', '666666666', 'Street', '12345678')
*
ERROR en linea 1:
ORA-00001: restriccion unica (FSDB253.PK_CLIENTS) violada


SQL> -- 1 row created.
SQL>
SQL> -- ----------------------------------------
SQL> -- Buying the ticket for a valid client
SQL> -- ----------------------------------------
SQL> insert into attendances values ('example@example.com', 'Barba', '07-11-2009', '4JL4W8F7T6R9D2K0S3P1Q5NMYX9CZ0E7V4B1U8I2O6A3H5G7', '01-01-20
insert into attendances values ('example@example.com', 'Barba', '07-11-2009', '4JL4W8F7T6R9D2K0S3P1Q5NMYX9CZ0E7V4B1U8I2O6A3H5G7', '01-01-2023')
*
ERROR en linea 1:
ORA-00001: restriccion unica (FSDB253.PK_ATTENDANCES) violada


SQL> -- 1 row created.
SQL>
SQL> -- ----------------------------------------
SQL> -- Checking the ticket is correctly created
SQL> -- ----------------------------------------
SQL> select * from attendances where client = 'example@example.com';

CLIENT                                                                                               PERFORMER
---------------------------------------------------------------------------------------------------- ----------------------------------------
example@example.com                                                                                  Barba

SQL> /*
```

2. The client is underaged: (worked as expected)

Unset

```
-- ----------------------------------------

-- Creating the underaged client

-- ----------------------------------------

insert into clients values ('example2@example2.com', 'John', 'Doe', 'Doe', '01-01-2010',
'666666666', 'Street', '23456781');
-- 1 row created.


-- ----------------------------------------

-- Buying the ticket for an underaged client

-- ----------------------------------------

insert into attendances values ('example2@example2.com', 'Barba', '07-11-2009',
'7KX5U8E2I1N6M3Y0R9D4L1O6V2CZ0P5A3H8B7T4G9S1Q5F7', '01-01-2023');

/*

insert into attendances values ('example2@example2.com', 'Barba', '07-11-2009',
'7KX5U8E2I1N6M3Y0R9D4L1O6V2CZ0P5A3H8B7T4G9S1Q5F7', '01-01-2023')


ERROR at line 1:

ORA-20001: The client must be of legal age.

ORA-06512: at "FSDB253.REJECT_AGE", line 11

ORA-04088: error during execution of trigger 'FSDB253.REJECT_AGE'

*/

-- ----------------------------------------

-- Checking the ticket is correctly rejected

-- ----------------------------------------


select * from attendances where client like 'example%';


/*

CLIENT                PERFORMER               WHEN               RFID               PURCHASE
----------------------------------------------------------

example@example.com   Barba   07/11/09   4JL4W8F7T6R9D2K0S3P1Q5NMYX9CZ0E7V4B1U8I2O6A3H5G7
01/01/23

*/
```

```
SQL> insert into clients values ('example2@example2.com', 'John', 'Doe', 'Doe', '01-01-2010', '666666666', 'Street', '23456781');

1 fila creada.

SQL> -- 1 row created.
SQL>
SQL> -- ----------------------------------------
SQL> -- Buying the ticket for an underaged client
SQL> -- ----------------------------------------
SQL> insert into attendances values ('example2@example2.com', 'Barba', '07-11-2009', '7KX5U8E2I1N6M3Y0R9D4L1O6V2CZ0P5A3H8B7T4G9S1Q5F7', '01-01-2023');
insert into attendances values ('example2@example2.com', 'Barba', '07-11-2009', '7KX5U8E2I1N6M3Y0R9D4L1O6V2CZ0P5A3H8B7T4G9S1Q5F7', '01-01-2023')
*
ERROR en linea 1:
ORA-20002: The client must be of legal age.
ORA-06512: en "FSDB253.REJECT_AGE", linea 11
ORA-04088: error durante la ejecucion del disparador 'FSDB253.REJECT_AGE'


SQL> /*
SQL> insert into attendances values ('example2@example2.com', 'Barba', '07-11-2009', '7KX5U8E2I1N6M3Y0R9D4L1O6V2CZ0P5A3H8B7T4G9S1Q5F7', '01-01-2023')
SQL>
SQL> ERROR at line 1:
SQL> ORA-20001: The client must be of legal age.
SQL> ORA-06512: at "FSDB253.REJECT_AGE", line 11
SQL> ORA-04088: error during execution of trigger 'FSDB253.REJECT_AGE'
SQL> */
SQL> -- ----------------------------------------
SQL> -- Checking the ticket is correctly rejected
SQL> -- ----------------------------------------
SQL>
SQL> select * from attendances where client like 'example%';

CLIENT                                                                                                                           PERFORMER
-------------------------------------------------------------------------------------------------------------------------------- --------------------------------------------------------------------------------
example@example.com                                                                                                              Barba
```

This example also represents the rejection of future birthdates, which have not yet occurred knowing that sysdate is the current date, like '01-01-2999'.

3. A birthdate is null: (worked as expected)

```
Unset
-- ----------------------------------------
-- Creating the client with null birthdate
-- ----------------------------------------
insert into clients values ('example3@example3.com', 'John', 'Doe', 'Doe', null, '666666666', 'Street', '34567812');


-- ----------------------------------------
-- Buying the ticket for the non-valid client.
-- ----------------------------------------
insert into attendances values ('example3@example3.com', 'Barba', '07-11-2009', '3RJ5D7V8N9X1F2G0K4S6B1U8M0Z5P7Q2H6T4L3A9E1C7Y2I', '01-01-2023');
/*
insert into attendances values ('example3@example3.com', 'Barba', '07-11-2009', '3RJ5D7V8N9X1F2G0K4S6B1U8M0Z5P7Q2H6T4L3A9E1C7Y2I', '01-01-2023')
*
ERROR at line 1:
ORA-20001: The client must be of legal age.
ORA-06512: at "FSDB253.REJECT_AGE", line 6
ORA-04088: error during execution of trigger 'FSDB253.REJECT_AGE'
*/
```

```
-- ----------------------------------------
-- Checking the ticket is correctly rejected
-- ----------------------------------------
select * from attendances where client like 'example%';
/*
CLIENT                PERFORMER               WHEN               RFID                PURCHASE
-------------------------------------------------------------
example@example.com   Barba   07/11/09   4JL4W8F7T6R9D2K0S3P1Q5NMYX9CZ0E7V4B1U8I2O6A3H5G7
01/01/23
*/
```

```
SQL> -- Creating the client with null birthdate
SQL> -- ----------------------------------------
SQL> insert into clients values ('example3@example3.com', 'John', 'Doe', 'Doe', null, '666666666', 'Street', '34567812');

1 fila creada.

SQL>
SQL> -- ----------------------------------------
SQL> -- Buying the ticket for the non-valid client.
SQL> -- ----------------------------------------
SQL> insert into attendances values ('example3@example3.com', 'Barba', '07-11-2009', '3RJ5D7V8N9X1F2G0K4S6B1U8M0Z5P7Q2H6T4L3A9E1C7Y2I', '01-01-2023');
insert into attendances values ('example3@example3.com', 'Barba', '07-11-2009', '3RJ5D7V8N9X1F2G0K4S6B1U8M0Z5P7Q2H6T4L3A9E1C7Y2I', '01-01-2023')
            *
ERROR en linea 1:
ORA-20001: The client birthdate is missing.
ORA-06512: en "FSDB253.REJECT_AGE", linea 7
ORA-04088: error durante la ejecucion del disparador 'FSDB253.REJECT_AGE'


SQL> /*
SQL> insert into attendances values ('example3@example3.com', 'Barba', '07-11-2009', '3RJ5D7V8N9X1F2G0K4S6B1U8M0Z5P7Q2H6T4L3A9E1C7Y2I', '01-01-2023')
SQL> *
SQL> ERROR at line 1:
SQL> ORA-20001: The client must be of legal age.
SQL> ORA-06512: at "FSDB253.REJECT_AGE", line 6
SQL> ORA-04088: error during execution of trigger 'FSDB253.REJECT_AGE'
SQL> */
SQL>
SQL>
SQL> -- ----------------------------------------
SQL> -- Checking the ticket is correctly rejected
SQL> -- ----------------------------------------
SQL> select * from attendances where client like 'example%';

CLIENT                                                                                  PERFORMER
--------------------------------------------------------------------------------------- ----------------------------------------------------------------
example@example.com                                                                     Barba
```

Even though the original statement only specifies the rejection of underaged customers, a null birthdate check is included, since it is an optional attribute in the 'CLIENTS' design but is not nullable in the trigger. Otherwise, a client with 'null' birthdate would be able to acquire a ticket.

4. Buying multiple tickets (worked as expected)

```
Unset
-- ----------------------------------------
-- Creating the valid clients
-- ----------------------------------------
```

```
insert all
into clients (e_mail, name, surn1, surn2, birthdate, phone, address, DNI) values
('example4@example4.com', 'John', 'Doe', 'Doe', '01-01-2000', '666666666', 'Street',
'45678123')
into clients (e_mail, name, surn1, surn2, birthdate, phone, address, DNI) values
('example5@example5.com', 'John', 'Doe', 'Doe', '01-01-2000', '666666666', 'Street',
'56781234')
into clients (e_mail, name, surn1, surn2, birthdate, phone, address, DNI) values
('example6@example6.com', 'John', 'Doe', 'Doe', '01-01-2000', '666666666', 'Street',
'67812345')
select 1 from dual;
-- 3 rows created.


-- ----------------------------------------
-- Creating the underaged clients
-- ----------------------------------------
insert all
into clients (e_mail, name, surn1, surn2, birthdate, phone, address, DNI) values
('example7@example7.com', 'John', 'Doe', 'Doe', '01-01-2010', '666666666', 'Street',
'78123456')
into clients (e_mail, name, surn1, surn2, birthdate, phone, address, DNI) values
('example8@example8.com', 'John', 'Doe', 'Doe', '01-01-2015', '666666666', 'Street',
'81234567')
into clients (e_mail, name, surn1, surn2, birthdate, phone, address, DNI) values
('example9@example9.com', 'John', 'Doe', 'Doe', '01-01-2020', '666666666', 'Street',
'01234567')
select 1 from dual;
-- 3 rows created.


-- ----------------------------------------
-- Checking the inserted clients
-- ----------------------------------------
select * from clients where e_mail like 'example%';
/*
E_MAIL NAME SURN1 SURN2 BIRTHDAT PHONE ADDRESS DNI
-----------------------------------------------------------
example2@example2.com John Doe Doe 01/01/10  666666666 Street 23456781
example3@example3.com John Doe Doe 666666666 Street 34567812
example4@example4.com John Doe Doe 01/01/00  666666666 Street 45678123
example5@example5.com John Doe Doe 01/01/00  666666666 Street 56781234
```

```
example6@example6.com John Doe Doe 01/01/00  666666666 Street 67812345
example7@example7.com John Doe Doe 01/01/10  666666666 Street 78123456
example8@example8.com John Doe Doe 01/01/15  666666666 Street 81234567
example9@example9.com John Doe Doe 01/01/20  666666666 Street 01234567
example@example.com John Doe Doe 01/01/00  666666666 Street 12345678

9 rows selected.
*/

-- ----------------------------------------
-- Checking the bought tickets
-- ----------------------------------------
select * from attendances where client like 'example%';

/*
CLIENT PERFORMER WHEN RFID PURCHASE
------------------------------------------------------------
example@example.com   Barba   07/11/09   4JL4W8F7T6R9D2K0S3P1Q5NMYX9CZ0E7V4B1U8I2O6A3H5G7
01/01/23
*/

-- ----------------------------------------
-- Buying three valid tickets
-- ----------------------------------------
insert all
into    attendances    (client,    performer,    when,    RFID,    purchase)    values
('example4@example4.com',                    'Barba',                    '07-11-2009',
'YRLIQASHY0K8ON20V09PHZN9X7J6GCG566NQHWVH1P3EB0T', '01-01-2023')
into    attendances    (client,    performer,    when,    RFID,    purchase)    values
('example5@example5.com',                    'Barba',                    '07-11-2009',
'1DFT9J71HDDZIOCS92Q28ICKM3P6NMAEY6FSATYT3Z1B8QR', '01-01-2023')
into    attendances    (client,    performer,    when,    RFID,    purchase)    values
('example6@example6.com',                    'Barba',                    '07-11-2009',
'GQ2F1IPWUAY5LL2RIP5DZLMNH8T4NNXZ0HRZ0LQ7LJN3L08', '01-01-2023')
select 1 from dual;
-- 3 rows created.

-- ----------------------------------------
```

```
-- Buying tickets for underaged clients
-- ----------------------------------------
insert all
into    attendances    (client,    performer,    when,    RFID,    purchase)    values
('example7@example7.com',                    'Barba',                    '07-11-2009',
'3ZNPHLWBMGE0N0IEJ1A2NPAIOJZOU1CDSVGZOM5NKNCUR6T', '01-01-2023')
into    attendances    (client,    performer,    when,    RFID,    purchase)    values
('example8@example8.com',                    'Barba',                    '07-11-2009',
'VMVTB9WSRG44BVUYJNWD3S6JLGJFP6CD6OXCKMFFYYQ58NH', '01-01-2023')
into    attendances    (client,    performer,    when,    RFID,    purchase)    values
('example9@example9.com',                    'Barba',                    '07-11-2009',
'GD1L33ALQE0N7DRTMCK8QLRWWIOMWU0MFVK2567733CT8O6', '01-01-2023')
select 1 from dual;
/*
into    attendances    (client,    performer,    when,    RFID,    purchase)    values
('example7@example7.com',                    'Barba',                    '07-11-2009',
'3ZNPHLWBMGE0N0IEJ1A2NPAIOJZOU1CDSVGZOM5NKNCUR6T', '01-01-2023')
     *
ERROR at line 2:
ORA-20002: The client must be of legal age.
ORA-06512: at "FSDB253.REJECT_AGE", line 11
ORA-04088: error during execution of trigger 'FSDB253.REJECT_AGE'
*/

-- ----------------------------------------
-- Checking the bought tickets
-- ----------------------------------------
select * from attendances where client like 'example%';

/*
CLIENT PERFORMER WHEN RFID PURCHASE
------------------------------------------------------------
example4@example4.com   Barba   07/11/09   YRLIQASHY0K8ON20V09PHZN9X7J6GCG566NQHWVH1P3EB0T
01/01/23
example5@example5.com   Barba   07/11/09   1DFT9J71HDDZIOCS92Q28ICKM3P6NMAEY6FSATYT3Z1B8QR
01/01/23
example6@example6.com   Barba   07/11/09   GQ2F1IPWUAY5LL2RIP5DZLMNH8T4NNXZ0HRZ0LQ7LJN3L08
01/01/23
example@example.com   Barba   07/11/09   4JL4W8F7T6R9D2K0S3P1Q5NMYX9CZ0E7V4B1U8I2O6A3H5G7
01/01/23
```

```
*/
```

```
SQL>
SQL> /*
SQL> CLIENT PERFORMER WHEN RFID PURCHASE
SQL> ----------------------------------------------------------
SQL> example@example.com Barba 07/11/09 4JL4W8F7T6R9D2K0S3P1Q5NMYX9CZ0E7V4B1U8I2O6A3H5G7 01/01/23
SQL> */
SQL>
SQL> -- ----------------------------------------
SQL> -- Buying three valid tickets
SQL> -- ----------------------------------------
SQL> insert all
  2  into attendances (client, performer, when, RFID, purchase) values ('example4@example4.com', 'Barba', '07-11-2009', 'YRLIQASHY0K8ON20V09PHZN9X7J6GCG566NQHWVH1P3EB0T', '01-01-2023')
  3  into attendances (client, performer, when, RFID, purchase) values ('example5@example5.com', 'Barba', '07-11-2009', '1DFT9J71HDDZIOCS92Q28ICKM3P6NMAEY6FSATYT3Z1B8QR', '01-01-2023')
  4  into attendances (client, performer, when, RFID, purchase) values ('example6@example6.com', 'Barba', '07-11-2009', 'GQ2F1IPWUAY5LL2RIPSDZLMNH8T4NNXZ0HRZ0LQ7LJN3L08', '01-01-2023')
  5  select 1 from dual;

3 filas creadas.

SQL> -- 3 rows created.
SQL>
SQL> -- ----------------------------------------
SQL> -- Buying tickets for underaged clients
SQL> -- ----------------------------------------
SQL> insert all
  2  into attendances (client, performer, when, RFID, purchase) values ('example7@example7.com', 'Barba', '07-11-2009', '3ZNPHLWBMGE0N0IEJ1A2NPAIOJZOU1CDSVGZOM5NKNCUR6T', '01-01-2023')
  3  into attendances (client, performer, when, RFID, purchase) values ('example8@example8.com', 'Barba', '07-11-2009', 'VMVTB9WSRG44BVUYJNWD3S6JLGJFP6CDGOXCKMFFYYQ58NH', '01-01-2023')
  4  into attendances (client, performer, when, RFID, purchase) values ('example9@example9.com', 'Barba', '07-11-2009', 'GD1L33ALQE0N7DRTMCK8QLRWWIOMWU0MFVK2567733CT8O6', '01-01-2023')
  5  select 1 from dual;
into attendances (client, performer, when, RFID, purchase) values ('example7@example7.com', 'Barba', '07-11-2009', '3ZNPHLWBMGE0N0IEJ1A2NPAIOJZOU1CDSVGZOM5NKNCUR6T', '01-01-2023')
     *
ERROR en linea 2:
ORA-20002: The client must be of legal age.
ORA-06512: en "FSDB253.REJECT_AGE", linea 11
ORA-04088: error durante la ejecucion del disparador 'FSDB253.REJECT_AGE'
```

```
SQL> /*
SQL> into attendances (client, performer, when, RFID, purchase) values ('example7@example7.com', 'Barba', '07-11-2009', '3ZNPHLWBMGE0N0IEJ1A2NPAIOJZOU1CDSVGZOM5NKNCUR6T', '01-01-2023')
SQL>      *
SQL> ERROR at line 2:
SQL> ORA-20002: The client must be of legal age.
SQL> ORA-06512: at "FSDB253.REJECT_AGE", line 11
SQL> ORA-04088: error during execution of trigger 'FSDB253.REJECT_AGE'
SQL> */
SQL>
SQL> -- ----------------------------------------
SQL> -- Checking the bought tickets
SQL> -- ----------------------------------------
SQL> select * from attendances where client like 'example%';

CLIENT                                                                                                   PERFORMER
-------------------------------------------------------------------------------------------------------- ----------------------------------------------------------
example4@example4.com                                                                                    Barba
example5@example5.com                                                                                    Barba
example6@example6.com                                                                                    Barba
example@example.com                                                                                      Barba

SQL>
SQL> /*
SQL> CLIENT PERFORMER WHEN RFID PURCHASE
SQL> ----------------------------------------------------------
SQL> example4@example4.com Barba 07/11/09 YRLIQASHY0K8ON20V09PHZN9X7J6GCG566NQHWVH1P3EB0T 01/01/23
SQL> example5@example5.com Barba 07/11/09 1DFT9J71HDDZIOCS92Q28ICKM3P6NMAEY6FSATYT3Z1B8QR 01/01/23
SQL> example6@example6.com Barba 07/11/09 GQ2F1IPWUAY5LL2RIPSDZLMNH8T4NNXZ0HRZ0LQ7LJN3L08 01/01/23
SQL> example@example.com Barba 07/11/09 4JL4W8F7T6R9D2K0S3P1Q5NMYX9CZ0E7V4B1U8I2O6A3H5G7 01/01/23
SQL> */
SQL>
```

3º Trigger

- **Design:**
- **Associated Table**: SONGS
- **Events**: when there is insertion
- **Temporality**: Before
- **Granularity**: Statement
- **Condition**: when writer and co-writer are reversed exists with the same title as another existing one with the same writer and co-writer.
- **Action**: Create a trigger that uses 'before insert on SONGS' and checks if a song where writer and co-writer are reversed exists with the same title as another existing one with the same writer and co-writer. If we create a simple trigger we get the error 'ORA-4091' (mutating table), so we must use a compound trigger.

- **SQL Implementation:**

```
Unset
CREATE OR REPLACE TRIGGER reject_reversed
FOR INSERT ON songs
COMPOUND TRIGGER

  TYPE song_type IS RECORD (
    title songs.title%TYPE,
    writer songs.writer%TYPE,
    cowriter songs.cowriter%TYPE
  );

  TYPE song_list_type IS TABLE OF song_type INDEX BY BINARY_INTEGER;

  songs_to_insert song_list_type;

  AFTER EACH ROW IS
  BEGIN
    songs_to_insert(songs_to_insert.COUNT+1).title := :new.title;
    songs_to_insert(songs_to_insert.COUNT).writer := :new.writer;
    songs_to_insert(songs_to_insert.COUNT).cowriter := :new.cowriter;
  END AFTER EACH ROW;

  AFTER STATEMENT IS
    repeated_songs int;
  BEGIN
    FOR i IN 1..songs_to_insert.COUNT LOOP
      SELECT COUNT(*) INTO repeated_songs
      FROM songs
      WHERE title = songs_to_insert(i).title
        AND writer = songs_to_insert(i).cowriter
        AND cowriter = songs_to_insert(i).writer;

      IF repeated_songs > 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'This song already exists.');
      END IF;
    END LOOP;
  END AFTER STATEMENT;
END reject_reversed;
/
```

We can delete the trigger by using the following command:

```
Unset
drop trigger reject_reversed;
```

Same code above but with comments (can be ignored):

-- Create or replace the compound trigger with the label 'reject_reversed'. Parts 1 and 2 of the compound trigger (BEFORE STATEMENT and BEFORE EACH ROW) are omitted as they were not needed.

```
CREATE OR REPLACE TRIGGER reject_reversed
FOR INSERT ON songs
COMPOUND TRIGGER

-- Declare a record type named 'song_type' with three fields: 'title', 'writer', 'cowriter'.
  TYPE song_type IS RECORD (
    title songs.title%TYPE,
    writer songs.writer%TYPE,
    cowriter songs.cowriter%TYPE
  );

-- Declare a user-defined table type named 'song_list_type' based on the aforementioned
'song_type' record type. Used to hold a collection of instances of a record type (song_type).
  TYPE song_list_type IS TABLE OF song_type INDEX BY BINARY_INTEGER;

-- Declare a variable named 'song_to_insert' of the aforementioned 'song_list_type' record
type. Used to hold a collection of instances of a record type ('song_type') to insert.
  songs_to_insert song_list_type;

-- Third part of the compound trigger, which is executed after each row is processed by the
trigger. It populates the collection 'songs_to_insert'.
  AFTER EACH ROW IS
  BEGIN
    songs_to_insert(songs_to_insert.COUNT+1).title := :new.title;
    songs_to_insert(songs_to_insert.COUNT).writer := :new.writer;
    songs_to_insert(songs_to_insert.COUNT).cowriter := :new.cowriter;
  END AFTER EACH ROW;
```

```
-- Fourth part of the compound trigger, which is executed after all the rows have been
processed by the trigger. It loops through the songs_to_insert collection, checks if the current
song already exists in the table, and if it does, it raises an error.

  AFTER STATEMENT IS

    repeated_songs int;

  BEGIN

    FOR i IN 1..songs_to_insert.COUNT LOOP

      SELECT COUNT(*) INTO repeated_songs

      FROM songs

      WHERE title = songs_to_insert(i).title

        AND writer = songs_to_insert(i).cowriter

        AND cowriter = songs_to_insert(i).writer;


      IF repeated_songs > 0 THEN

        RAISE_APPLICATION_ERROR(-20003, 'This song already exists.');

      END IF;

    END LOOP;

  END AFTER STATEMENT;

END reject_reversed;

/
```

- **Testing:**

Originally, there are 123698 rows in 'SONGS'.

We observe that 'writer' cannot be null, as it composes the primary key for 'SONGS' like 'title' does.

In order to do the tests correctly, we must respect the constraint:

- 'FK_SONGS1' of 'SONGS' that references an existing musician, for which we have used 'SE>>0572457878', which belongs to the first row of 'MUSICIANS' as its rownum is equal to 1.

- - The constraint 'FK_SONGS2' also references an existing musician, for which we have used 'SE>>0483834065', which is the second row of 'MUSICIANS'.

We can delete every created row for testing by using the following command:

```
Unset
delete songs where title like 'example%';
-- CAUTION: This command and or any 'delete songs where
...' freezes sqlplus.
```

The following testing cases have been considered:

1. The writers have one song with the same title: (worked as expected)

```
Unset
-- ----------------------------------------
-- Creating the song
-- ----------------------------------------
insert into songs values ('example 199', 'SE>>0572457878', 'SE>>0483834065');
-- 1 row created.


-- ----------------------------------------
-- Checking the song is correctly created
-- ----------------------------------------
select * from songs where title like 'example%';
/*
TITLE WRITER COWRITER
--------------------------------------------------
example 199 SE>>0572457878 SE>>0483834065
*/


-- ----------------------------------------
-- Creating the already existing song
-- ----------------------------------------
insert into songs values ('example 199', 'SE>>0572457878', 'SE>>0483834065');
/*
insert into songs values ('example 199', 'SE>>0572457878', 'SE>>0483834065')
```

```
*
ERROR at line 1:
ORA-00001: unique constraint (FSDB253.PK_SONGS) violated
*/


-- ----------------------------------------
-- Creating the same song with reversed writers
-- ----------------------------------------
insert into songs values ('example 199', 'SE>>0483834065', 'SE>>0572457878');
/*
insert into songs values ('example 199', 'SE>>0483834065', 'SE>>0572457878')
            *
ERROR at line 1:
ORA-20003: This song already exists.
ORA-06512: at "FSDB253.REJECT_REVERSED", line 31
ORA-04088: error during execution of trigger 'FSDB253.REJECT_REVERSED'
*/
```

```
SQL> insert into songs values ('example 199', 'SE>>0572457878', 'SE>>0483834065');
insert into songs values ('example 199', 'SE>>0572457878', 'SE>>0483834065')
*
ERROR en linea 1:
ORA-00001: restriccion unica (FSDB253.PK_SONGS) violada


SQL> /*
SQL> insert into songs values ('example 199', 'SE>>0572457878', 'SE>>0483834065')
SQL> *
SQL> ERROR at line 1:
SQL> ORA-00001: unique constraint (FSDB253.PK_SONGS) violated
SQL> */
SQL>
SQL> -- ----------------------------------------
SQL> -- Creating the same song with reversed writers
SQL> -- ----------------------------------------
SQL> insert into songs values ('example 199', 'SE>>0483834065', 'SE>>0572457878');
insert into songs values ('example 199', 'SE>>0483834065', 'SE>>0572457878')
            *
ERROR en linea 1:
ORA-20003: This song already exists.
ORA-06512: en "FSDB253.REJECT_REVERSED", linea 31
ORA-04088: error durante la ejecucion del disparador 'FSDB253.REJECT_REVERSED'


SQL> /*
SQL> insert into songs values ('example 199', 'SE>>0483834065', 'SE>>0572457878')
SQL>              *
SQL> ERROR at line 1:
SQL> ORA-20003: This song already exists.
SQL> ORA-06512: at "FSDB253.REJECT_REVERSED", line 31
SQL> ORA-04088: error during execution of trigger 'FSDB253.REJECT_REVERSED'
SQL> */
SQL>
```

2. The writers have many songs with the same title: (worked as expected)

```
Unset
-- ----------------------------------------
-- Creating the songs
-- ----------------------------------------
insert all
into songs (title, writer, cowriter) values ('example 1199', 'SE>>0572457878', 'SE>>0483834065')
into songs (title, writer, cowriter) values ('example 1299', 'SE>>0572457878', 'SE>>0483834065')
into songs (title, writer, cowriter) values ('example 1399', 'SE>>0572457878', 'SE>>0483834065')
select 1 from dual;
-- 3 rows created.
```

```
-- ----------------------------------------
-- Creating the same songs with reversed writers
-- ----------------------------------------
insert all
into songs (title, writer, cowriter) values (‘example 1199’, 'SE>>0483834065', 'SE>>0572457878')
into songs (title, writer, cowriter) values (‘example 1299’, 'SE>>0483834065', 'SE>>0572457878')
into songs (title, writer, cowriter) values (‘example 1399’, 'SE>>0483834065', 'SE>>0572457878')
select 1 from dual;
/*
into songs (title, writer, cowriter) values ('example 1199', 'SE>>0483834065', 'SE>>0572457878')
     *
ERROR at line 2:
ORA-20003: This song already exists.
ORA-06512: at "FSDB253.REJECT_REVERSED", line 31
ORA-04088: error during execution of trigger 'FSDB253.REJECT_REVERSED'

*/

select * from songs where title = ‘example 1199’ or title = ‘example 1299’ or title = ‘example 1399’;

/*
TITLE                                                WRITER         COWRITER
---------------------------------------------------- -------------- --------------
example 1199                                         SE>>0572457878 SE>>0483834065
example 1299                                         SE>>0572457878 SE>>0483834065
example 1399                                         SE>>0572457878 SE>>0483834065
*/
```

```
SQL> */
SQL> -- ----------------------------------------
SQL> -- Creating the songs
SQL> -- ----------------------------------------
SQL> insert all
  2  into songs (title, writer, cowriter) values ('example 1199', 'SE>>0572457878', 'SE>>0483834065')
  3  into songs (title, writer, cowriter) values ('example 1299', 'SE>>0572457878', 'SE>>0483834065')
  4  into songs (title, writer, cowriter) values ('example 1399', 'SE>>0572457878', 'SE>>0483834065')
  5  select 1 from dual;

3 filas creadas.

SQL> -- 3 rows created.
SQL>
SQL> -- ----------------------------------------
SQL> -- Creating the same songs with reversed writers
SQL> -- ----------------------------------------
SQL> insert all
  2  into songs (title, writer, cowriter) values ('example 1199', 'SE>>0483834065', 'SE>>0572457878')
  3  into songs (title, writer, cowriter) values ('example 1299', 'SE>>0483834065', 'SE>>0572457878')
  4  into songs (title, writer, cowriter) values ('example 1399', 'SE>>0483834065', 'SE>>0572457878')
  5  select 1 from dual;
into songs (title, writer, cowriter) values ('example 1199', 'SE>>0483834065', 'SE>>0572457878')
     *
ERROR en linea 2:
ORA-20003: This song already exists.
ORA-06512: en "FSDB253.REJECT_REVERSED", linea 31
ORA-04088: error durante la ejecucion del disparador 'FSDB253.REJECT_REVERSED'


SQL> /*
SQL> into songs (title, writer, cowriter) values ('example 1199', 'SE>>0483834065', 'SE>>0572457878')
SQL>      *
SQL> ERROR at line 2:
SQL> ORA-20003: This song already exists.
SQL> ORA-06512: at "FSDB253.REJECT_REVERSED", line 31
SQL> ORA-04088: error during execution of trigger 'FSDB253.REJECT_REVERSED'
SQL>
SQL> */
```

```
SQL> */
SQL> select * from songs where title = 'example 1199' or title = 'example 1299' or title = 'example 1399';

TITLE                                              WRITER         COWRITER
-------------------------------------------------- -------------- --------------
example 1199                                       SE>>0572457878 SE>>0483834065
```

2 Concluding Remarks

As a start, the three members of the group believe that this has been one of the most difficult projects we have ever had, which has been even more complicated with the time we had. While the delivery was announced in plenty of time, we did not acquire and (more importantly) hone the knowledge necessary to develop the work until weeks later. Nonetheless, we believe that this is not due to our ineptitude, nor lack of effort on our part or on the part of the teachers, but because of the high difficulty inherent in the subject.

Regarding the queries, the most difficult problem we had was the different types of compilation problem while coding with SQL. Regarding the Procedures, we struggled particularly with the 3º one due to its size. And for the views, we were not able to complete the last one and we

couldn't find a way to update the view and now the table or viceverda. Finally, for the trigger, we had to faced the mutating table error and we used compound triggers to solve it.

With respect to future improvements, we think it could be consider a good idea to divide the projects into several smaller assignments. Finally, without a shadow of a doubt, this work will help us in the future, not only to better understand the subject, but also to better understand the digital world around us.