Universidad Carlos III

Files and databases 2022-23

Course 2022-23

LabReport P1

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

Index

1 Introduction

This document will deal with the creation of a new database using SQL+ for a musical events production and management company called Melomaniacs. To this database, we will carry out the insertion of data coming from a less structured database provided by the professors containing tables (fsdb.recordings, fsdb.artists, fsdb.melomaniacs and fsdb.livesingings). The idea is to achieve a correct insertion without any errors arising as along the way to create the database and insert the data, several errors will appear. These errors may occur because we did a poor design or because specific errors have been introduced in the fsdb database for us to find them. To perform this work, we first designed an indicative relational graph that we changed a lot in the following section to have a basis from which we could start creating the database in sql. As we went on, problems related with sql started arising so we had to do a big modification of the previous relational graph. After creating the tables, we started inserting although when inserting more errors occurred so we had to keep modifying the tables that we created before. Our objective for this work was to do a database that simulates a professional database used for events organized by these types of companies.

In addition to this document, 2 additional files can be found on the zip file which are creation.sql and upload.sql containing the tables and insertions respectively.

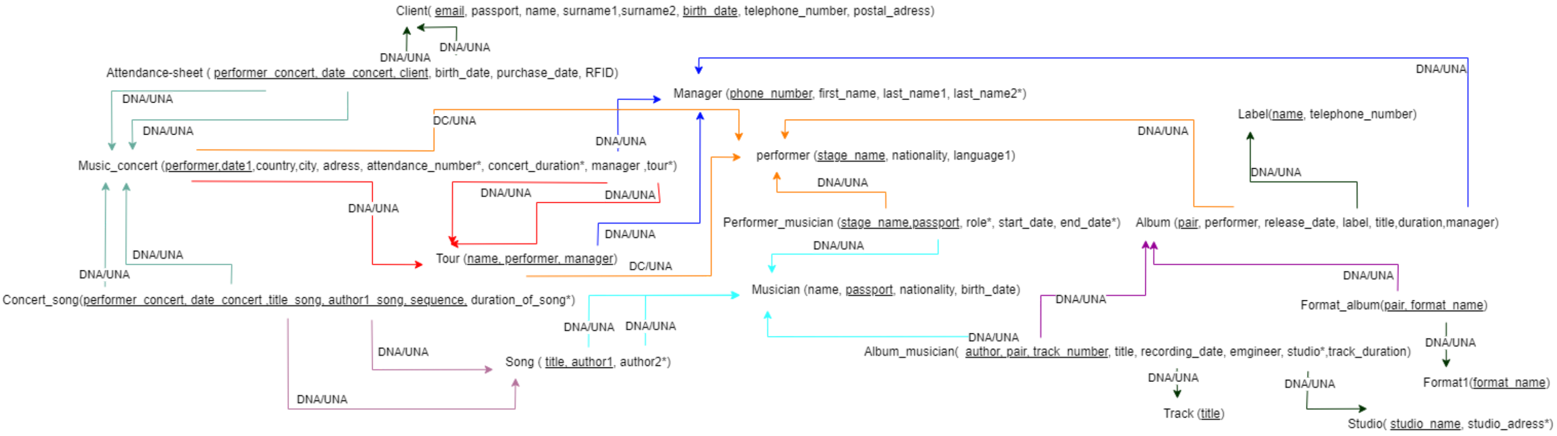
Finally, this document will consist of a first section containing all information related to the relational design, including the updated relational graph, implicit semantics and non observed explicit semantics.

The second section will contain all the information related to the implementation. It has 3 tables, one containing the reincorporated semantics (consisting of those semantics that we were not able to implement in the relational design), another with the implicit semantics and the excluded semantics.

Finally, a description of the insertion of data into tables takes place. In this section we set the order of insertion of data and the problems we found while inserting the data.

2 Relational Design

- **Relational Schema:**


Client(email, passport, name, surname1,surname2, birth_date, telephone_number, postal_adress)
DNA/UNA
DNA/UNA
Attendance-sheet (performer_concert, date_concert, client, birth_date, purchase_date, RFID)
DNA/UNA
DNA/UNA
DC/UNA
DNA/UNA
Manager (phone_number, first_name, last_name1, last_name2*)
Label(name, telephone_number)
DNA/UNA
DNA/UNA
Music_concert (performer,date1,country,city, adress, attendance_number*, concert_duration*, manager ,tour*)
performer (stage_name, nationality, language1)
DNA/UNA
DNA/UNA
DNA/UNA
DNA/UNA
DNA/UNA
Performer_musician (stage_name,passport, role*, start_date, end_date*)
Album (pair, performer, release_date, label, title,duration,manager)
DNA/UNA
DNA/UNA
DNA/UNA
Tour (name, performer, manager)
DC/UNA
DNA/UNA
Concert_song(performer_concert, date_concert ,title_song, author1_song, sequence, duration_of_song*)
Musician (name, passport, nationality, birth_date)
DNA/UNA
Format_album(pair, format_name)
DNA/UNA
DNA/UNA
DNA/UNA
DNA/UNA
DNA/UNA
Song (title, author1, author2*)
Album_musician(author, pair, track_number, title, recording_date, emgineer, studio*,track_duration)
DNA/UNA
DNA/UNA
DNA/UNA
Format1(format_name)
DNA/UNA
Track (title)
Studio(studio_name, studio_adress*)

- **Implicit semantics:**

Presp_id	Stage	Mechanism	Description
I_1	Design	Primary key	author1 has been included as a primary key in table song
I_2	Design	Relation	Performer_musician relation is created as a middle relation between performer and musician because a performer can have several musicians or one and a musician can be on several bands although not at the same time.
I_3	Design	Relation	Album_musician relation is created as a middle relation between album and musician as a musician can have a song of his own in several albums and an album can have songs of different musicians from the performer of the album.
I_4	Design	Attribute	The attribute role, start_date and end_date has been included on performer_musican because a musician could have been on multiple bands/soloist with different roles,start dates and end_date.

I_5	Design	Attribute	We put the birth_date on Client and Attendance_sheet mandatory because what the client tells us is that on Attendance_sheet we must extract the birth_date and email from Client to later check if they are older than 18 years to buy a concert ticket. (there is a contradiction on the pdf because on Client it says the birthdate is optional but later on the Attendance_sheet it says it is mandatory).
I_6	Design	Attribute	We put studio optional on album_musician because the client wants that if we remove an studio that information is removed from the rows affected from album_musician but we don't delete that row.
I_8	Design	Primary key	We added manager as a primary key on Tour because we want to check that the manager on concert is the same as on tour if possible (tour not null)
I_9	Design	Primary key/ attribute	We put on the table Concert_song the attribute sequence as a primary key because a performer on a concert could sing the same song multiple times with different son durations
I_{10}	Design	Primary key/ attribute	On the table album_musician we added a new attribute as a primary key track_number because it could be the case that the same song could be repeated on the same album
I_{11}	Design	Relation	We created a middle relation between Format1 and album called Format_album because one album could have multiple formats

Table 1: Implicit semantics incorporated into the relational graph

- **Non-observed explicit semantics:**

Presp_id	Description
S_1	We cannot verify that the client that purchased a ticket is older than 18 years old.
S_2	We cannot verify that a performer has two or more tours with the same name.
S_3	We cannot verify that a musician has two songs with the same name
S_4	We cannot verify the data types of all the attributes
S_5	We cannot verify that the track duration is less than 90 minutes
S_6	We cannot check the number_of_attendance_number is bigger or equal than 0
S_7	We cannot verify the duration of a concert is equal to the sum of the song_duration of all the songs performed on that concert
S_8	We cannot verify that a band is a formation of more than 1 musician
S_9	We cannot verify that a tour has more than 1 concert (not included)
S_{10}	We cannot verify the duration of an album is equal to the sum of the track of all the songs performed on that concert
	We cannot verify that the count (track_num) on an album corresponds to the range of that album. e.g the album has 10 tracks but one of them says that its track num is 15.

Table 2: Non-observed explicit semantics

3 Relational Statics Implementation in SQL (DDL)

- **Re-incorporated semantics:**
  (identifiers referred to those assigned in table 1)

Presp_id	Solution Description
S_1	There is no age lower than 18 for the purchase of a ticket (attribute: purchase_date) in table attendance sheet.
S_2	The fact that two tours of the same performer cannot have the same name has been included. In class Tour, we said that performer and name were primary keys, therefore, there can not be 2 tours with the same name and performer although there can be 2 tours with the same name but different performers.
S_3	No musician gives the same name to 2 of his songs. This has been taken into account as in the class song (for livesingings), title and author are primary keys so happens that the same combination of title and author cannot appear.
S_4	For every attribute, we defined its datatype. Later on the insert, we made sure that all the data types were transformed into the correct one.
S_5	On the album_musician table we included a check that the duration of a track must be shorter than 90 minutes. As in the original fsdb database, the duration of songs was declared in seconds, we said that the duration must be shorter than 5400 seconds (which is equal to 90 minutes).
S_6	We have checked that the number of attendees is greater or equal to 0.

Table 3: re-incorporated explicit semantics

- **Incorporated implicit semantics:**
  (numbering continues where ended in table 2)

Presp id	Stage	Mechanism	Description
I_{12}	Implem	Check	There is no negative number of attendees in the music_concert table.
I_{13}	Implem	Check	The start_date should be older than the end_date on the performer_musician table.

Table 1(cont.): implicit semantics incorporated in the definition of each table

- **Excluded semantics:**

Presp_id	Description	Cause	Explicit/Implicit
E_1	Contracts are automatically updated with the company's update (integrity option UC on the FK referencing *Companies*).	PL/SQL does not observe this integrity option	Implicit
E_2	We cannot verify that a band is a formation of more than 1 musician	We are not sure if SQL allow to do this kind of operations	explicit
E_3	We cannot verify the duration of an album is equal to the sum of the track of all the songs performed on that concert	We are not sure if SQL allow to do this kind of operations	explicit
E_4	We cannot verify that a tour has more than 1 concert (not included)	We are not sure if SQL allow to do this kind of operations	explicit
E_5	We cannot verify the duration of a concert is equal to the sum of the song_duration of all the songs performed on that concert	We are not sure if SQL allow to do this kind of operations	implicit
E_6	We cannot verify that the count (track_num) on an album corresponds to the range of that album. e.g the album has 10 tracks but one of them says that its track num is 15.	We are not sure if SQL allow to do this kind of operations	implicit

Table 5: explicit semantics excluded in the creation of each table

4 Workload (DML)

- **Order of insertion:**

The order we have selected to insert the data is the following:

- The first insertion will occur in table **format1** as it is not referencing any other table, (that will tell us that it cannot be the first one) and is only referenced by one table which is format_album.
- Second insertion is **label** as it is not referencing any table and only one other table has a reference to it which is album.
- Third insertion is **studio** because of the same reason as the previous two.
- Fourth insert is **Manager** which does not reference anything and is referenced by 2 other tables.
- Fifth insert is **Manager** this happens, as it will be explained in the problems that arise, because the previous insert was done from fsdb.recordings, but there when we wanted to insert concerts and tours from livesingings, there was a manager that was in fsdb.livesingings but not in fsdb.recordings.
- Sixth insert is **Client** as it is not referencing any table and is being referenced only once.
- Seventh insert is **Performer** as it is not referencing any table and is being referenced by three other tables.
- Eighth insert is **Musician** as it is not referencing any table and is being referenced by four other tables.
- Ninth insert is **Performer_musician** which references performer and musician, which have already been created.
- Tenth insert is **Album** which references the label, performer and manager. All of these tables have already been created so no problem should arise. Moreover, the Album table is being referenced by format_album which will be created right before this one and album_musician which will be created later.
- Eleventh insert is **Format_album** which references format1 and album acting as a middle relation among these two.
- The twelfth insert is **Track** which is referenced by album_musician that will be created later.
- The thirteenth insert is **Album_musician** which references Album, Musician, Track and Studio. All the mentioned tables had to be created before album_musician.
- The fourteenth insert is **Song** which references a musician. It is referenced by Concert_song.

- The fifteenth insert is **Tour** which references Performer and Manager. It is referenced by music_concert
- The sixteenth insert is **Music_concert** which references the tour manager and performer. It is referenced by concert_song and attendance_sheet that will be created later.
- The seventeenth insert is **Concert_song** which references a song and music concert as it is a middle relation between these two tables.
- The last insert is **Attendance_sheet** which references the client and music_concert.

Errors found on the insertion of data:

The errors that we have found are specified in this section.

The first error was that there were (performer,date) tuples (concert) on fsdb.melomaniacs that were not in fsdb.livesingings. There was not enough data such as the city where the concert was going to be held or address or number of attendees,etc . That data that was absent, was necessary for the insertion in the table music concert. As those attributes were missing, the rows for those concerts in melomaniacs were not created.

There was a manager in fsdb.livesinging that was not included in fsdb.recordings. But we had enough information to create a row in the table manager including the one in fsdb.livesingings to all those managers that were already inserted from fsdb.recordings.

There was a case where a performer in a band had an end_date older than the start_date (on performer_musician) .