BS DEGREE IN INFORMATICS ENGINEERING Academic year: 2018/2019 - 2nd year, 2nd term Subject: File Structures and Databases First Assignment's Report: Relational Design and Impl.	uc3m | Universidad Carlos III de Madrid

Lecturers:	Francisco Javier Calle Gómez & Pedro Miguel Suja Goffin		
Group:	**88**	**Lab User**	**FSDB284**
Student:	**María del Carmen Díaz de Mera Gómez-Limón**	**NIA:**	**100383384**
Student:	**Marina Torelli Postigo**	**NIA:**	**100383479**

1 Introduction

The problem to solve is the creation of a new database for a vinyl shop, to keep a record of all vinyls and the information related, as well as information about the songs played in the radio and about the clients and their purchases.

The already existing database had a lack of coverage, as it consisted of three tables with all the information in them and no added constraints, such as primary keys and foreign keys, therefore retrieving information could become complicated.

We were tasked with creating a new relational design for the database and doing a massive import of the information from the old database to the new one. The new database would keep all the relevant information while also having a better relational design between all fields of information.

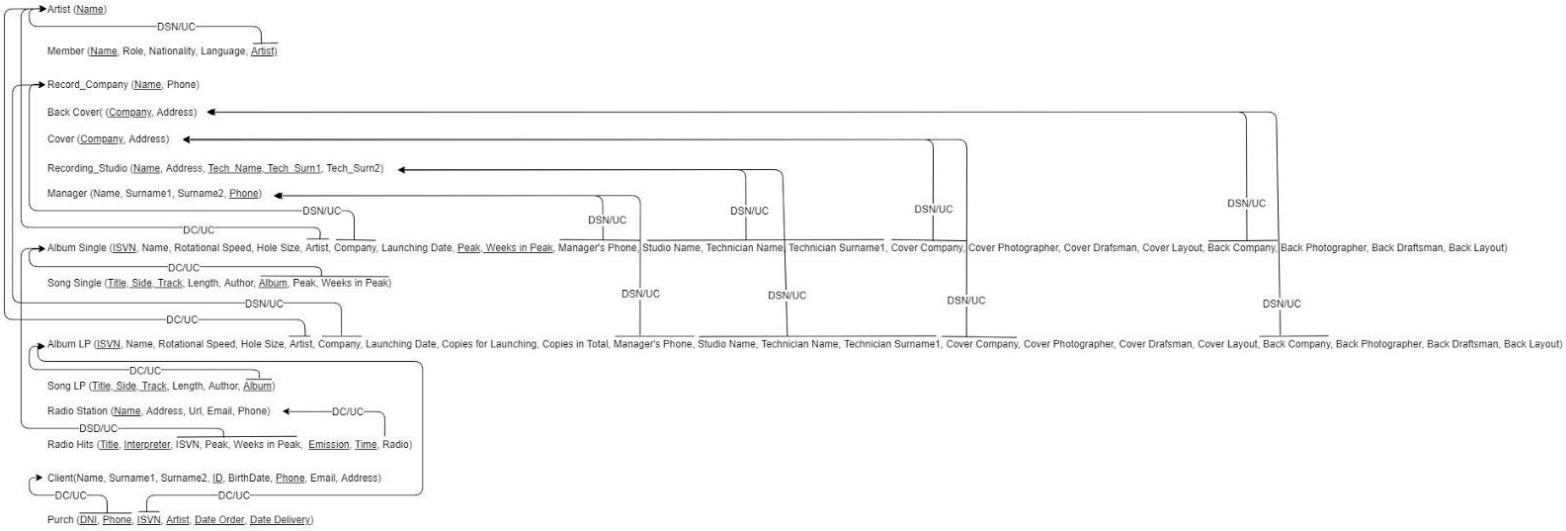
In order to implement this design in SQL, we have created two sql scripts will all the commands:

· To create the tables we have the *creation.sql* script

· To insert all the information from the old tables to the new ones we have the *upload.sql* script

2 Relational Design

- Relational Schema:



We have uploaded the relational schema in the zip file for better quality, *relational_schema.jpg*

We divided the given three tables in several different new tables with all the attributes divided.

To deal with the fact that, depending on the format of the vinyl, different attributes are stored we have created a different table for each format. Thus, there are two tables for album as well as for song (songs in LP don't have a peak and weeks in peak).

To dynamically implement the members of an artist, we created a separate table to store the members, with a reference to the artist they belong to, which can be changed easily when they leave a group or join a new one.

- Implicit semantics:

Presp id	Stage	Mechanism	Description
I_1	Design	Primary key	In addition to the ISVN, singles are identified by their peak and number of weeks in the peak, as two singles could be part of the same album (one on each side) and have different peaks.
I_2	Design	Primary key	An artist is identified by its name
I_3	Design	Primary key	The members of an artist are identified by their name and the role in their group/solo artist
I_4	Design	Primary key	The recording studio is identified by its name and the name and surname of the technician that was in charge of the album (The table recording_studio actually represents technicians)
I_5	Design	Primary key	All songs (single and LP) are identified by their title, the album they belong to and the side and track we can find them on.
I_6	Design	Primary key	A hit is identified by its title, interpreter, album it belongs to and the peak it reached and number of weeks it stayed there as well as the date and time of emission in the radio, as the same song could be played at different times
I_7	Design	Primary key	Clients are identified by their DNI and phone, as some of them have a repeated DNI as it only includes the number and not the letter
I_8	Design	Updates	All foreign keys are implemented using an update cascade, as if we change a value in one of the fields, we would want all the related values updated too.
I_9	Design	Relation	The information about nationality and language is associated to each member instead of the artist, as different members could come from different countries.
	Design	Relation	Hits will always be singles, purchases can only be of LP, as they are in different tables.

Table 1: Implicit semantics incorporated into the relational graph

- Non-observed explicit semantics:

Presp_id	Description
S_1	A group can only have up to 5 members
S_2	Vinyls only have two sides: 'Side A' and 'Side B'.
S_3	In a purchase, the date of delivery has to be later or equal to the date of the order
S_4	Clients can only do an order a day
S_5	You can only cancel orders if they haven't been delivered
S_6	When songs are deleted and a vinyl remains with no songs on any of its sides, the vinyl is deleted.

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_3	CONSTRAINT CQ_DATE CHECK (date_del >= date_order) in table Purch_
S_2	CONSTRAINT CQ_SIDE CHECK (side IN ('Side A', 'Side B')) in both Song tables.

Table 3: re-incorporated explicit semantics

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

Presp_id	Stage	Mechanism	Description
I_{10}	Implem.	Field size	The peak value is a number of 3 digits, as we consider the Top 100 Billboard
I_{11}	Implem.	Field size	No song has stayed in the peak more than 15 weeks, so the weeks in peak value is a number of 2 digits
I_{12}	Implem.	Field size	The record holder longest title song in the world has almost 500 characters, so we put that number as a limit in the name of album field
I_{13}	Implem.	Check	Phone numbers (In manager table and client table) always have 9 digit: We add a constraint CHECK (phone>=100000000 AND phone<1000000000)
I_{14}	Implem.	Field Size	DNI has 8 numbers, as we don't include the letter.

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

- Excluded semantics:

Presp_id	Description	Cause	Explicit/ Implicit
E_1	Artists have up to 5 members	The way we defined the relational design allows for as many members as they want to join an artist	Explicit
E_2	Vinyls with no songs on any of their sides should be deleted	Not being able to implement this constraint in SQL	Explicit
E_3	When the artist of a hit is deleted, instead of deleting the hit the default value is established	SQL does not allow the implementation of on delete set default	Explicit
E_4	You can only cancel orders if they haven't been delivered	This check should be done before deleting the rows, not in the implementation of the relational design.	Explicit

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

4 Workload (DML)

To create and dump data in the tables we first create the ones with no foreign keys, as they are independent from all the other tables and then the tables that depend on them.

The implemented order of the tables is:

- Artist, Record_Company, Manager, Recording_Studio, Cover and Back_Cover
- Member (Depends only on artist)
- Album_Single and Album_LP (Depend on the previous ones)
- Song_LP and Song_Single (Depend on Album_LP and Album_Single)
- Radio_Station
- Radio_Hits (Depends on Radio_Station and Album_Single)
- Client
- Purchase (Depends on Client and Album_LP)

After creating all the tables and defining all the constraints, we start to import the information from the all tables.

The simpler tables could easily import data with an INSERT INTO <name of table> SELECT DISTINCT <values> FROM <name of old table>. In some of the other ones we had to add some conditions to meet the requirements of our new design.

To fill the members table we had to do 5 different inserts, one for the field of each of the five members in the old vinyl table. We only included those rows that didn't have a null value in the corresponding field, as well as a null role, as we wanted it to be our primary key with the name. We added the conditions *WHERE member2 IS NOT NULL AND rol2 IS NOT NULL* for member2 for example.

To fill the Album_LP table we selected only the rows from vinyl where the format was LP so we added the condition *WHERE format = 'LP vinyl'*. We did the same for singles with format = 'single'. However, we found that some of the singles in the old table had the field peak null, so to avoid violations of the PK constraint we also added *rel_copies IS NOT NULL AND tot_copies IS NOT NULL.*

Once we had all the information of the albums we could also add the songs. As with the albums, the songs from LP where added with the condition *WHERE format = 'LP vinyl'* and likewise to the single ones.

In clients we had to add a condition to use the DNI as primary key, as some of the clients had that field empty, so we decided to take them out and not import them to our new design with the condition *WHERE cl_dni IS NOT NULL.*

The more complicated inserts were the Radio_Hits and Purchase, as we needed to take information from two tables to complete the design.

The radio_hits table had two different inserts, as in the old table some of them included in the artist field the name of the artist, but in others you had the ISVN there with the format 'ISVN:XXXXXXXX'.

In the first insert we inserted the ones with the artist, taking the isvn, peak and weeks in peak from the vinyl table. We had to add several conditions:

· *hits.title=vinyl.title AND hits.artist=vinyl.artist*, so that we take the equivalent rows from the two tables.

· *vinyl.format='single'*, hits will always be a single, as they are the ones released to be played in the radio.

· *vinyl.rel_copies IS NOT NULL AND vinyl.tot_copies IS NOT NULL*, as they are part of the foreign key and cannot take null value.

The second insert took into account the ones where the isvn was stored in the artist field, so we had to take the name of the artist from the vinyl table. To take the number of the isvn, we used the function SUBSTR, so as to not take the initial 'ISVN:', so we took the substring from the position 6 to the end: *SUBSTR(hits.artist, 6)*.

The conditions were the same ones except that we compared the title and the isvn instead of the title and the artist to get the same rows from the two tables.

With purchases we did a similar thing, as we need to get the isvn of the album to have a reference to the Album table, but in the initial purchases we only had the name of the album and the artist. We considered only LPs for purchases and had to use the condition *WHERE vinyl.album=purchases.album AND vinyl.artist=purchases.artist* to get the equivalent rows in both tables. We only took clients whose DNI wasn't null (as explained before).