Lecturers:	CALLE GOMEZ, FRANCISCO JAVIER		
Group:	16	**Lab User**	DBF16
Student:	Alex Li	**NIA:**	100547697
Student:	Danny Smith	**NIA:**	100547747
Student:	Divinity Gines	**NIA:**	100548084

1 Introduction

The purpose of this lab is to develop a relational database for CaffeineManiacs TM Inc., an e-commerce company specializing in coffee products. The database needs to manage various aspects of the company, including product catalogs, suppliers, clients, and order processing. The current database is highly inadequate, consisting of just three disjointed tables that do not satisfy the necessary constraints or relationships.

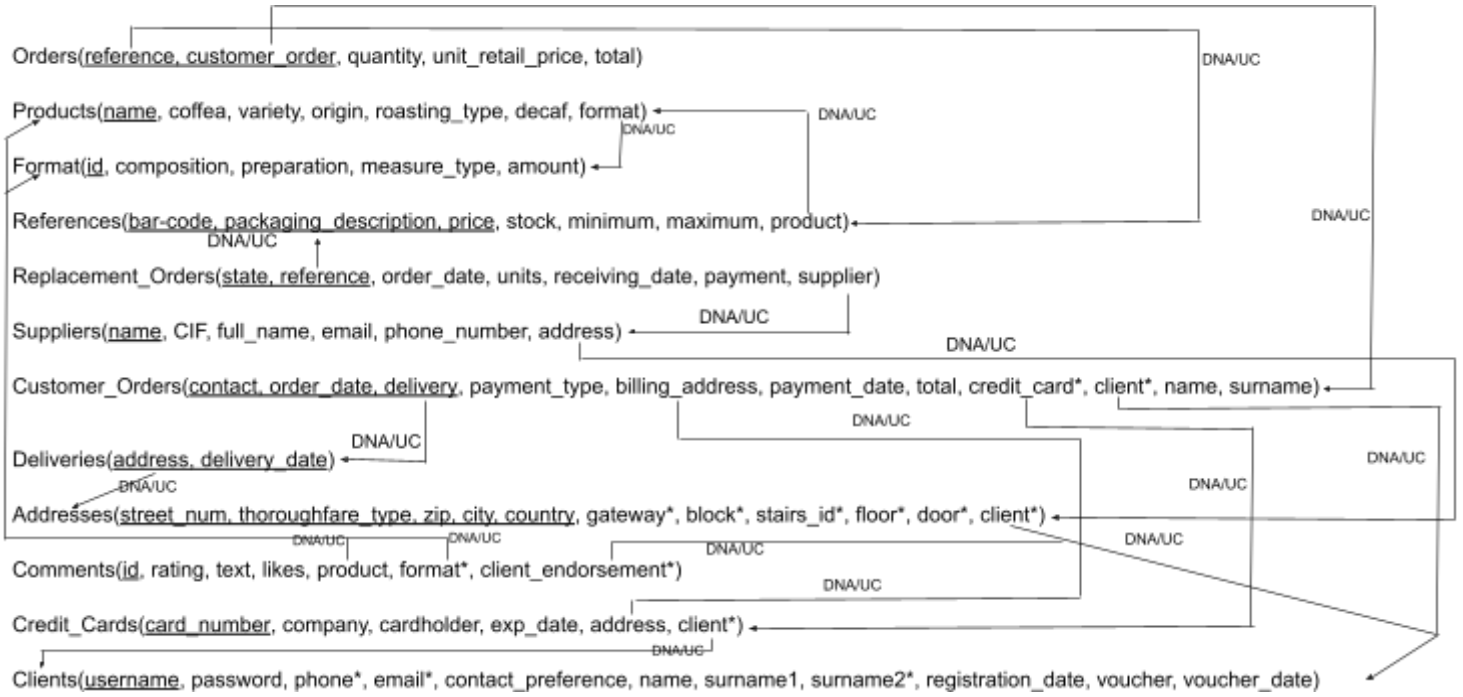
Redesigning a database for an ecommerce coffee company, we began to remap the data by creating a relational design for the entire database; connecting the tables together and understanding the included implicit and non-observed explicit semantics. We added more tables to the database since the initial one only had 3: catalogue, trolley, and posts. Trying to keep the variables similar to the previous database, for an easier transition between databases, we made sure to expand on what there already was. After finishing our relational design, we began to create it with SQL, still being mindful of the semantics of the database. Next step was to upload the workload by moving the data from the old database to the new one we created. This process is documented through the upload.sql file. The steps we took as described above follow the same structure as the current document in order to properly ensure clarity of our thought process from start to finish.

The following sections will cover:

1. The relational schema, detailing the entities and relationships.
2. Implementation of the schema using PL/SQL, with attention to constraints.
3. Data population and addressing issues during the migration process.

2 Relational Design

- Relational Schema



- Implicit semantics: semantic presuppositions that are not found in the explicit description, but which are required to complete the relational design.

Presp id	Stage	Mechanism	Description
I_1	Design	Primary key	Formats are identified by a unique id
I_2	Design	Primary key	Comments are identified by a unique id so they are still uniquely identifiable even if the user is removed from the database

- Non-observed explicit semantics: each of the explicit presuppositions (stated in the problem description) that could not be included in the relational graph.

Presp id	Description
S_1	Phone numbers have 9 digits (at least, at most)
S_2	Emails must include an '@' symbol
S_3	If there is more than one supplier for that reference, one will be chosen in order of cheapest cost, then fastest previous delivery times, then by fewest deliveries in the past year.
S_4	The default min stock for a reference is 5 and the default max stock is 10 for making automatic orders for stock replacement
S_5	When stock falls below the minimum threshold, a replacement order is generated automatically
S_6	Only one replacement order is allowed per reference per day

S_7	New references start with zero stock unless a different value is specified during insertion

3 Relational Statics Implementation in SQL (DDL)

This section must include the creation of each table. In addition to the code (*NEWcreation.sql* script) for creating tables (valid syntax in PL/SQL), you should include the correspondent subsections referring to the excluded semantics that are re-incorporated, the newly incorporated implicit semantics, and the explicit semantics that were observed but are now excluded. All these sections will be accomplished by fulfilling the correspondent table (see tables 3, 4 and 5). Any of these tables is empty (in case), the table should be omitted and replaced by a phrase such as "Has not been reported."

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

Presp_id	Solution Description
S_1	field size is 9; a constraint (*constraint_name*) CHECK (phone≥100000000) is added to the table <*table_name*>
S_2	needs to include '@', CONSTRAINT email_check CHECK (email LIKE '%@%'); is added to the table *Clients*

Table 3: re-incorporated explicit semantics

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

Presp_id	Stage	Mechanism	Description
I_3	Implem	Default	New References have a default stock value of 0, triggering a replacement order if the current stock drops below 5
I_4	Implem	Default	If no voucher value is provided for new clients, it defaults to 0 in the Clients table, ensuring no null values are inserted
I_5	Implem	Default	Contact preference is automatically set to SMS if a phone number exists for the client

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	PL/SQL does not observe this integrity option	Implicit

	update (integrity option UC on the FK referencing *Companies*).		
E_2	With new references having default stock value of 0, a replacement order would have to occur because 0 < min value (default = 5)	PL/SQL does not trigger this behavior automatically	Explicit
E_3	Credit card expiration date cannot be null or invalid; this validation is required	PL/SQL needs additional checks not enforced by table design	Explicit

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

4 Workload (DML)

This section will describe the uploading of the workload (*NEWload.sql* script) from the tables provided (and described in the statement). To this end, we will analyze the problem of populating the tables with the workload. The solution will be described, with emphasis on:

- The specific order of the creation of tables to dump data into them (reasoned).
- The problems that arise (obligatory field value, inconsistencies in the original data, etc...) and the solutions adopted to overcome them.

We created the tables in order of dependency to avoid referential integrity violations. So tables like Clients and Formats were created first because they have no foreign keys to other tables.

For most of the tables, such as Products, References, Suppliers, Comments, and Addresses; it was a smooth transition of variables between the old database and the new one we created. As for the Customer_Orders, Credit_Cards, and Formats; there are small changes in order to make sure the variables line up. In the transition to the new database, the Customer_Orders table consolidates customer details, order information, and delivery data, ensuring that contacts (email/phone) and billing addresses are correctly captured, while adding support for card payments. The Credit_Cards table now uniquely identifies each card, linking it to clients and their associated billing addresses. The Formats table has been enhanced to distinguish between different product packaging and preparation methods, including whether items are sold by volume or unit.

The Deliveries, Replacement_Orders, and Clients tables didn't receive an insert script because their data structures differ significantly from the old database schema, lacking direct counterparts in the old data. The Deliveries table introduces specific links to addresses and delivery dates, which weren't explicitly separated before. Similarly, Replacement_Orders involves supplier and order fulfillment processes, not present in the old schema. For Clients, new attributes like contact preferences and voucher systems were introduced, requiring more nuanced data mapping.