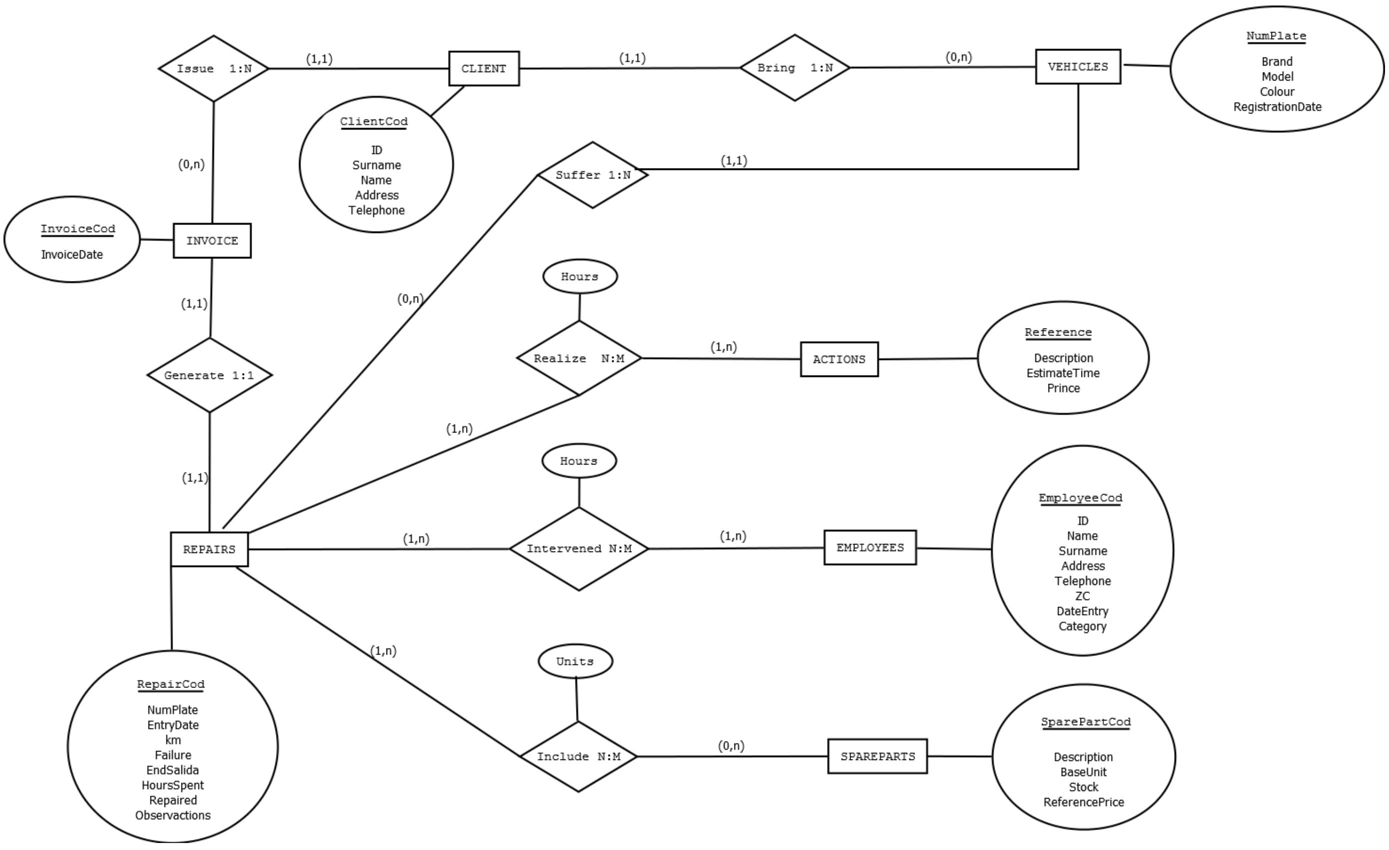
GBD02
Por Pedro Antonio Ruiz Martínez

TASK 1



7 initially define entities (**CLIENT**, **VEHICLES**, **INVOICE**, **REPAIRS**, **EMPLOYEES**, **SPAREPARTS** and **ACTIONS**).
The degree of relations between entities is of second degree because relationships not involving in more than two entities.
The possibility about binary relations correspondence are three: 1: 1, 1: N, N: M.
There are 3 relationships N: M will result tables in the relational model whose
main keys are the keys of the entities that make up the relationship. Furthermore 3
N: M have their own attributes, which in any case also generate a new
table in the relational model but the correspondence was 1: 1 or 1: N.
I put some examples:

Relation between CLIENT and VEHICLES:

A CLIENTE may have one or various VEHICLES, and a VEHICLES belongs to a CLIENT.
Therefore, placing the vehicle view of a VEHICLES can obtain belong
at least one client and at most to a CLIENT, accounting for a cardinality (1,1).
Placing the CLIENTE obtain views on this may have at least one VEHICLES and as Maximum number of vehicles being the cardinality at this end (0,n).
The maximum cardinality, 1 and N defines the degree of the relation in this case is (1: N).

Relation RAPAIRS and EMPLOYEES:

In a REPAIRS they may have involved one or more EMPLOYEES, and an EMPLOYEE may be involved in one or several REPAIRS. "

Therefore, placing the view on REPAIRS obtain a repair may have been make for at least one EMPLOYEE and a maximum of several EMPLOYEES, establishing cardinality (1,n). Placing the view over EMPLOYEES get this may have been involved in at least one REPAIR and at most several REPAIRS, establishing new cardinality (1, N).
The maximum cardinality of both ends are N and N, defined in this case the degree of relationship of several to several N: M.

OBTAINING THE RELATIONAL MODEL:

Relation between **CLIENT** and **INVOICE** is of one to various 1:N.
The key field in the **CLIENT** table (**ClientCode**) must be on the table
INVOICE as foreign key.

The relationship between the **CLIENT** entities and **VEHICLES** is one to multiple 1: N, so the key field in the **CLIENT** table (**ClientCode**) must be on the table **VEHICLES** as foreign key.

The relationship between the **REPAIRS** and **INVOICE** entities is one to one 1:1. In this If we propagate in the **RepairCod** **INVOICE** table key, since data for the elaboration of the **INVOICE** will be obtained from the already obtained in the table **REPAIRS**.

The relationship between entities **REPAIRS** and **VEHICLES** is one to several 1:N, so the key field of the table **VEHICLES** (**NumPlate**) must be included in the **REPAIRS** table as foreign key.

The "Intervienen" relationship defined between the **REPAIRS** entities and **EMPLOYEES**, is several to several N: M, so that in the relational model will result in a new table whose keys are the main keys of both entities **EmployeeCod**, **RepairCode**, and also it must include the attribute of the relationship, "**Hours**".

The relationship "**Include**" defined between entities **REPAIRS** and **SPAREPARTS** is too many to many N:M, so that in the relational model will lead to a new table whose keys are the main keys of both entities, **SparePartCod**, **RepairCod**, and also it must include the proper attribute "**Units**".

The "**Realize**" relationship defined between the **REPAIRS** and **ACTIONS** is too many to many N:M, so that in the relational model will lead to a new table whose keys are the main keys of both entities, **SparePartCod**, **Reference**, and also it must include the attribute of the relationship, "**Hours**".
It is important to note that the three relationships "**Intervened**", "**Include**" and "**Realize**" would also place tables in the relational model was even one to many as 3 have attributes, which would generate a new table.

With the foregoing, the relational model that remains is:

Relational Model:

CLIENT (**ClientCod**, ID, Surname, Name, Address, Telephone)
INVOICE (**InvoiceCod**, InvoiceDate, ClienteCod, RepairCod)
VEHICLES (**NumPlate**, Brand, Model, Colour, RegistrationDate, ClienteCod)
REPAIRS (**RepairCod**, NumPlate, EntryDate, Km, failure, EndDate, Repaired, Observations, HoursSpent)
EMPLOYEES (**EmployeeCod**, ID, Name, Surnames, Address, Telephone, ZC, DateEntry, Category)
SPAREPARTS (**SparePartCod**, Description, BaseUnit, Stock, ReferencePrice)
ACTIONS (**Reference**, Description, EstimateTime, Price)
Intervened (**EmployeeCod**, RepairCod, Hours)
Include (**SparePartCod**, RepairCod, Units)
Realize (**RepairCod**, Reference, Hours)

GBD02
Por Pedro Antonio Ruiz Martínez

TASK 2

1FN
For this we will have to eliminate repetitive groups, if they exist. We will study the dependencies between attributes to choose an appropriate key.

Order_Number + Product_ID → Order_Date, Provider_Code, Provider_Name, Provider_Address, Product_Name, Product_Price, Amount.

ORDER (Order_Number, Order_Date, Provider_Code, Provider_Name, Provider_Address, Product_ID, Product_Name, Product_Price, Amount)

It is already in the first normal way to comply:

- It has a definite key
- There are no repetitive groups.
- All attributes depend on the primary key.

2FN
To be in 2FN there can be no functional dependencies of any attribute on a part of the primary key. We must therefore see whether there are partial dependencies. The functional dependencies of non-key attributes, relative to the key are:

Order_Number + Product_ID → Amount
Order_Number → Order_Date, Provider_ID, Provider_Name, Provider_Address
CodProducto → Product_Name, Product_Price

For it is in 2FN talk that removing those dependencies by decomposing the tables:

ORDER-PRODUCT (**Order_Number**, **Product_ID**, Amount)
ORDER (**Order_Number**, Order_Date, Provider_ID, Provider_Name, Provider_Address)
PRODUCT (**Product_ID**, Product_Name, Product_Price)

It is already in the second normal way to comply:

- It is in 1FN .
- There are no partial dependencies, ie no attribute depends on a part of the primary key.

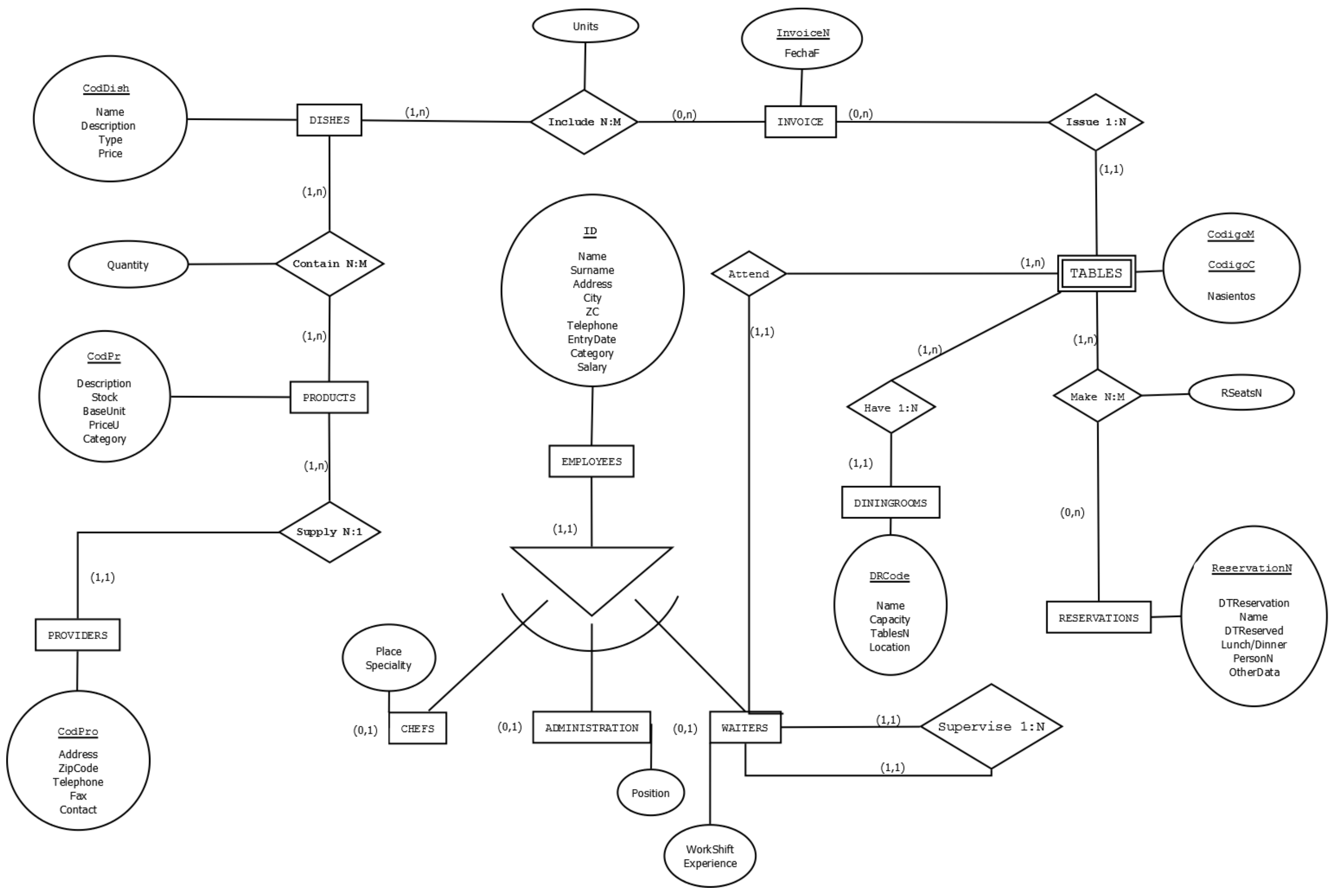
3FN
For the tables to be in 3FN we will have to eliminate, if they exist, the transitive dependencies between non-key attributes. We see that in the REQUEST relationship there are these types of dependencies:
NumPedido → FechaPedido
CodProveedor → NombreProveedor, DirecciónProveedor

ORDER-PRODUCT (**Order_Number**, **Product_ID**, Amount)
ORDER (**Order_Number**, Order_Date, Provider_ID)
PRODUCT (**Product_ID**, Product_Name, Product_Price)
PROVIDER (**Provider_ID**, Provider_Name, Provider_Address)

Already in third normal way to comply:

- It is in 2FN
- There are no transitive functional dependencies.

TASK 3



We define 11 entities that are (DININGROOMS, TABLES, RESERVATIONS, EMPLOYEES, WAITERS,CHEFS, ADMINISTRATION, INVOICE, DISHE, PRODUCTS, PROVIDERS).
The relationships are of second grade. In the model three types of relationships are given: 1:1, 1:N, N:M.

<u>OBTAINING THE RELATIONAL MODEL</u>:

Relationship between **DININGROOMS** and **TABLES** is 1: N so the key field of the **DININGROOMS** (**DRCode**) table must be included in the table **TABLES** as foreign key.

The "**Make**" relationship between the **RESERVATIONS** and **TABLES** entities is several to several N: M so the Relational model will lead to a new table whose keys are the main keys of both entities, such as **DRCode**, **ReservationN**, **TableC**, and which must also include the Relationship, " **RSeatsN** ".

In order to avoid null values in the **EMPLOYEES** entity, generalization hierarchies are used. having the Supertype entity **EMPLOYEES** with the common attributes (ID ... Name, etc ...) and the Subtype entities (**WAITERS**, **CHEFS**, **ADMINISTRATION**) that inherit the attributes of the Supertype. The cardinalities of the **EMPLOYED** supertype are (1,1) and those of the subtypes of (0,1).

Relationship between **TABLES** and **INVOICE** is 1: N so the key field in the TABLES table (**TableC, DRCode**) must be included in the INVOICE table.

The "**Include**" relationship between the **INVOICES** and **PLATES** is from several to several N: M so the Relational model will lead to a new table whose keys are the main keys of both Entities (**InvoiceN**, **DishC**) and that also must include the own attribute of the relation,
"**Units**".

The relationship "**Contain**" between **DISHES** and **PRODUCT**S entities is from several to several N: M so The relational model will give rise to a new table whose keys are the main keys of both Entities (**DishC**, **ProductC**) and that also must include the own attribute of the relation,"**Quantity**".

<u>Relational Model</u>:

EMPLOYEES (**ID**, Name, Surname, Address, City, Telphone, ZC, EntryDate, Category, Salary)

CHEFS (**ID**, Place, Especiality)

ADMINISTRATION (**ID**, Position)

WAITERS (**ID**, WorkShift, Experience, Manager_ID)

DININGROOMS (**DRCode**, Name, Capacity, TablesN, Location)

TABLES (**TableC, DRCode**, SeatsN, WaitersID)

Make (**TableC, DRCode, ReservationN**, RSeatsN)

RESERVATIONS (**ReservationN**, DTReservation, Name, DTReserved, Lunch/Dinner, PersonN, OtherData)

INVOICE (**InvoiceN**, InvoiceD, CustomerID, **TableC, DRCode**)

Include (**InvoiceN, DishC**, Units)

DISHES (**DishC**, Name, Description, TypeD, Price)

Contain (**DishC, CodigoPr**, Amount)

PRODUCTS (**ProductC**, Description, Stock, BaseUnit, UnitPrice, Category, ProviderC)

PROVIDERS (**ProviderC**, Address, ZC, Telphone, Fax, Contact)