Bases de Datos I
Unidad VIII

Lenguaje de Definición de Datos
SQL/DDL (Diseño Físico)

https://www.flaticon.com/free-icons/relations

SQL/DDL

- **Lenguaje de definición de datos (LDD) o Data Definition Languaje (DDL).**
- **Formado por sentencias que permiten la manipulación de los diferentes elementos que componen un base de datos.**
- **Vamos a utilizarlo para realizar el diseño físico (Una parte).**
- **Contiene sentencias para la creación (CREATE), modificación (ALTER) y eliminación (DROP) de los diferentes elementos.**

Sentencias

- **Sentencias que lo componen:**

CREATE: permite la creación o definición de un nuevo elemento, que formará parte de la base de datos.

ALTER: permite modificar elementos existentes que componen la base de datos.

DROP: permite eliminar elementos existentes que componen la base de datos.

Sintaxis básica CREATE

- **CREATE [tipo_elemento] [nombre] [opciones];**

  - Permite crear el elemento [tipo_elemento] con nombre [nombre], según las opciones [opciones] especificadas.

  - La sentencia falla, si existe otro elemento de mismo tipo [tipo_elemento] con el mismo nombre [nombre].

Sintaxis básica ALTER

- **ALTER [tipo_elemento] [nombre] [opciones];**

  - Permite modificar el elemento [tipo_elemento] con nombre [nombre], según las opciones [opciones] especificadas.

  - La sentencia falla, si no existe un elemento con nombre [nombre] del tipo [tipo_elemento], o si la modificación fuera imposible de realizar.

Sintaxis básica DROP

- **DROP [tipo_elemento] [nombre] [RESTRICT | CASCADE];**
  - Permite eliminar el elemento [tipo_elemento] con nombre [nombre].
  - La sentencia falla si el elemento no existe o si existen otros elementos que dependan del que se intenta eliminar.
  - **CASCADE** Permite eliminar no sólo el elemento, sino también, todo elemento dependiente de él.
  - Si se omite **CASCADE** o **RESTRICT**, por defecto **RESTRICT**.

Tipos de elementos

- **Elementos manipulables con SQL/DDL:**

DATABASE: base de datos (no estándar).

SCHEMA: grupo nombrado de tablas.

TABLE: tabla de la base de datos.

DOMAIN: tipo de dato definido por el usuario.

VIEW: tabla virtual o secuencia almacenada.

Creación de Base de Datos

- **CREATE DATABASE [n];**
  - **Crear la base de datos con nombre [n].**

- **Ejemplo:**

  CREATE DATABASE prueba;

Modificación de Base de Datos

- **ALTER DATABASE [n] RENAME TO [p];**
  - **Modifica el nombre de la base de datos de [n] a [p].**

- **Ejemplo:**

  ALTER DATABASE prueba RENAME TO test;

Eliminación de Base de Datos

- **DROP DATABASE** [n];
  - Elimina la base de datos de nombre [n] y los elementos que contiene la misma.
  - Esta operación no puede realizarse cuando existan conexiones a la base (Está en uso).
  - Esta operación no puede deshacerse.

- **Ejemplo:**

  DROP DATABASE test;

Creación de Esquemas

- **CREATE SCHEMA [n];**
  - Crear el esquema con nombre [n].

- **Ejemplo:**

  CREATE SCHEMA stock;

Modificación de Esquemas

- **ALTER SCHEMA [n] RENAME TO [p];**
  - **Modifica el nombre del esquema de [n] a [p].**

- **Ejemplo:**

  ALTER SCHEMA stock RENAME TO insumos;

Eliminación de Esquemas

- **DROP SCHEMA** [n];
  - Elimina el esquema de nombre [n].
  - Falla si el esquema contiene elementos.
- **DROP SCHEMA** [n] **CASCADE**;
  - Elimina el esquema de nombre [n] y los elementos que contiene el mismo.
- **Ejemplos:**

  DROP SCHEMA insumos;

  DROP SCHEMA insumos **CASCADE**;

Creación de Tablas

- **CREATE TABLE [esquema].[nombre] (**

  [columna] [tipo_dato] [restricción_columna],

  [columna] [tipo_dato] [restricción_columna],

  …

  [restricción_fila], [restricción_fila], …

  );

  - **Crea una nueva tabla con nombre [nombre] en la base de datos, con las columnas, tipos de datos y restricciones.**
  - **El esquema donde incluirla es opcional. Si se omite, los SGBD incluyen un esquema por defecto (en PostrgeSQL es PUBLIC) o la incluyen en el esquema actual.**

Ejemplo creación de tablas

```
CREATE TABLE empleado(
  legajo INTEGER PRIMARY KEY,
  documento INTEGER UNIQUE,
  apellido VARCHAR(50) NOT NULL,
  nombre VARCHAR(50) NOT NULL,
  nacimiento DATE NOT NULL
);
```

Tipos de datos

- **Tipos de datos que incluye SQL:**
  - **Números enteros**
  - **Números de punto flotante**
  - **Número de precisión arbitraria**
  - **Cadenas de caracteres**
  - **Fechas y horas**
  - **Lógicos**
  - **Binarios**

Números enteros

- **SMALLINT** (2 bytes)
  - Número enteros de -32768 a +32767.
- **INTEGER** (4 bytes)
  - Números enteros de -2147483648 a +2147483647.
- **BIGINT** (8 bytes)
  - Números enteros de -9223372036854775808 a +9223372036854775807.

Números de punto flotante

- **REAL** (4 bytes)
  - Número reales de precisión variable, no exactos.
- **DOUBLE** (8 bytes)
  - Número reales de precisión variable, no exactos.
- **Incluye valores especiales:**
  - Infinity
  - -Infinity
  - NaN

Número de precisión arbitraria

- **DECIMAL(precisión, escala)**
  - Número reales de precisión definida, exactos.
- **Precisión:**
  - Cantidad de dígitos en total.
- **Escala:**
  - Cantidad de dígitos decimales.
- **Está especialmente recomendado para el almacenamiento de cantidades en las que se requiere exactitud.**

Cadenas de caracteres

- **CHARACTER VARYING(n) o VARCHAR(n)**
  - Cadenas de caracteres de tamaño variable.
- **CHARACTER(n) o CHAR(n)**
  - Cadenas de caracteres de tamaño fijo.

- **Es posible especificar la codificación de los caracteres haciendo uso de CHARACTER SET.**

Fecha y hora

- **DATE [WITH[OUT] TIMEZONE]**
  - Tipo de datos fecha.
- **TIME [WITH[OUT] TIMEZONE]**
  - Tipo de datos hora.
- **TIMESTAMP [WITH[OUT] TIMEZONE]**
  - Tipo de datos fecha y hora.
- **INTERVAL**
  - Tipo de datos intervalo de tiempo.

Lógicos

- **BOOLEAN**
  - Tipo de datos lógico.
  - Existen las constantes **TRUE** y **FALSE**.
- Hay que tener en cuenta que en **SQL** existen tres valores de verdad **TRUE**, **FALSE** y **UNKNOW**.
- **UNKNOW** surge de la operación con **NULL**.
- Muchos **SGBD** no implementan **UNKNOW**.

Binarios

- **BLOB o BINARY LARGE OBJECT**
  - Para almacenar binarios, por ejemplo imágenes, documentos, etc.

- **CLOB o CHARACTER LARGE OBJECT**
  - Para almacenar cadenas de caracteres de gran tamaño (textos).

Restricciones de columnas

- **Restricciones de columnas:**
  - **De una sola (columna) :**

    NULL / NOT NULL

    DEFAULT

    GENERATED
  - **De una sola (columna) o más de una (fila):**

    PRIMARY KEY

    UNIQUE

    CHECK

    FOREIGN KEY

Restricción NULL / NOT NULL

- **Se utiliza para definir columnas obligatorias (por defecto opcionales).**
- **Sintaxis:**

  [columna] [dominio] **NOT NULL**

- **Ejemplos:**

  fecha **DATE NOT NULL**

  cantidad **INTEGER NOT NULL DEFAULT** 0

- **Omitirla, implica NULL**

Restricción DEFAULT

- **Se utiliza para definir valores por defecto para la columna.**
- **Sintaxis:**

  [columna] [tipo_dato] **DEFAULT** [expresión]

- **Ejemplos:**

  cantidad **INTEGER DEFAULT 0**

  fecha **DATE DEFAULT CURRENT_DATE**

- **Omitirla, implica DEFAULT NULL**

Restricción GENERATED

- **Si bien no es una restricción nueva (SQL:2003), no es ampliamente implementada por los SGBD.**

- **Puede utilizarse para definir:**
  - Identificadores sustitutos
  - Columnas calculadas
- **Si el SGBD no implementa GENERATED pueden usarse secuencias (SEQUENCE).**

Identificadores sustitutos

- **Para identificadores sustitutos, podemos indicarlos como política por defecto u obligatoria**

  GENERATED ALWAYS AS IDENTITY

  GENERATED BY DEFAULT AS IDENTITY

- **Ejemplo:**

  Codigo **INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY**

Columnas calculadas

- **Como columnas calculadas podemos indicar por defecto u obligatorio y almacenada o siempre calculada:**

  GENERATED ALWAYS AS (expression) [VIRTUAL | STORED]

- **Ejemplo:**

  edad **SMALLINT GENERATED ALWAYS AS (CURRENT_DATE** – nacimiento**) VIRTUAL**

Restricción PRIMARY KEY

- **Se utiliza para definir la clave primaria de la tabla.**

- **De columna (claves simples)**

  documento **INTEGER PRIMARY KEY**

- **De fila (claves compuestas)**

  PRIMARY KEY(número, fecha**)**

- **Las columnas marcadas con PRIMARY KEY, no aceptan nulos.**

Restricción UNIQUE

- **Se utiliza para definir claves secundarias de la tabla.**

- **De columna (claves simples)**

  expediente **CHAR(10) UNIQUE**

- **De fila (claves compuestas)**

  UNIQUE(número, año**)**

- **Por defecto las columnas UNIQUE aceptan nulos, a menos que especifiquemos NOT NULL.**

Restricción CHECK

- **Se utiliza para definir restricciones sobre los posibles valores de la columna o columnas, a través de una expresión lógica.**

- **De columna**

  documento **INTEGER CHECK**(documento > 0)

- **De fila**

  CHECK(cantidad * precio > 0**)**

Restricción FOREIGN KEY

- **Se utiliza para definir claves foráneas. Estas pueden referenciar claves primarias o secundarias.**
- **De columna (claves simples)**

  columna **REFERENCES** tabla [política]

  columna **REFERENCES** tabla(clave) [política]
- **De fila (claves compuestas)**

  FOREIGN KEY(columna1, columna2**) REFERENCES** tabla [política]

  FOREIGN KEY(columna1, columna2**) REFERENCES** tabla(clave1, clave2) [política]

Políticas de clave foránea

- **¿Que hacer, en caso de la modificación o la eliminación para mantener la integridad referencial?**

- **Políticas**

  ON [DELETE | UPDATE] [acción]

- **Acciones**

  [RESTRICT | CASCADE | SET NULL | SET DEFAULT]

Ejemplo de Creación de Tabla

- **Creación de una tabla persona, con columna documento como clave primaria y nombre obligatorio no vacío.**

```
CREATE TABLE persona(
  documento INTEGER PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL
  CHECK( nombre <> '')
);
```

Ejemplo de Creación de Tabla

- **Creación de una tabla de vehículos con una clave primaria compuesta y una clave foránea simple.**

```
CREATE TABLE vehiculo(
  letra CHAR(3) NOT NULL,
  numero SMALLINT NOT NULL,
  propietario INTEGER REFERENCES persona
  ON UPDATE CASCADE ON DELETE SET NULL,
  PRIMARY KEY(letra, numero)
);
```

Modificación de tablas (ALTER)

- **ALTER TABLE [nombre] [opc];**
  - Permite realizar distintos tipos de modificaciones sobre una tabla.
- **Ejemplo: modificar el nombre de una tabla**

  ALTER TABLE contribuyente **RENAME TO** persona**;**
- **Ejemplo: modificar el esquema de una tabla**

  ALTER TABLE contribuyente **SET SCHEMA** dgr**;**

Modificación de columnas (ALTER)

- **Ejemplo: agregar una columna a una tabla**

  ALTER TABLE persona **ADD COLUMN** genero **CHAR(1);**

- **Ejemplo: eliminar una columna a una tabla**
  - **ALTER TABLE** persona **DROP COLUMN** genero**;**

Modificación de columnas (ALTER)

- **Ejemplo: renombrar una columna de una tabla**

  ALTER TABLE persona **RENAME COLUMN** sexo **TO** genero**;**

- **Ejemplo: cambiar el tipo de datos de una columna**

  ALTER TABLE persona **ALTER COLUMN** genero **TYPE INTEGER;**

Modificación de restricciones ALTER

- **Ejemplo: agregar una restricción a la tabla**

  ALTER TABLE persona **ADD CONSTRAINT** genero_o **CHECK(**genero **IN** ('F','M')**);**

- **Ejemplo: eliminar una restricción a la tabla**

  ALTER TABLE persona **DROP CONSTRAINT** genero_o**;**

Modificación de restricciones ALTER

- **Ejemplo: agregar un valor por defecto a la columna**

  ALTER TABLE persona **ALTER COLUMN** genero **SET DEFAULT** 1**;**

- **Ejemplo: eliminar una restricción de obligatoriedad a la columna**

  ALTER TABLE persona **ALTER COLUMN** genero **DROP NOT NULL;**

Eliminación de tablas (DROP)

- **DROP TABLE** [n];

  Elimina la tabla de nombre [n]. Falla si la tabla está relacionada de alguna forma.

- **DROP TABLE** [n] **CASCADE**;

  Elimina la tabla de nombre [n] y los elementos que contengan referencia a la misma.

- **Ejemplo:**

  DROP TABLE persona;

Creación de dominios (CREATE)

- **CREATE DOMAIN [n] [tipo] [restricciones];**

  Crea un nuevo tipo de dato de nombre [n], con tipo [tipo] y restricciones [restricciones].

- **Ejemplo:**

```
CREATE DOMAIN sexo CHAR(1) NOT NULL
CHECK(VALUE IN ('F','M'));
```

Modificación de dominios (ALTER)

- **ALTER DOMAIN [n] RENAME TO [p];**

  Modifica el nombre del dominio de [n] a [p].

- **Ejemplo:**

  ALTER DOMAIN sexo **RENAME TO** genero;

Eliminación de dominios (DROP)

- **DROP DOMAIN** [n];

  Elimina el dominio de nombre [n]. Falla si el dominio ha sido utilizado.

- **DROP DOMAIN** [n] **CASCADE**;

  Elimina el dominio de nombre [n] y las referencias al mismo.

- **Ejemplo:**

  DROP DOMAIN genero;

Uso de dominios

- **Una vez definido un dominio, puede ser utilizado como cualquier tipo de dato de SQL:**

```
CREATE TABLE persona(
...
genero GENERO NOT NULL DEFAULT 'F'
...
);
```

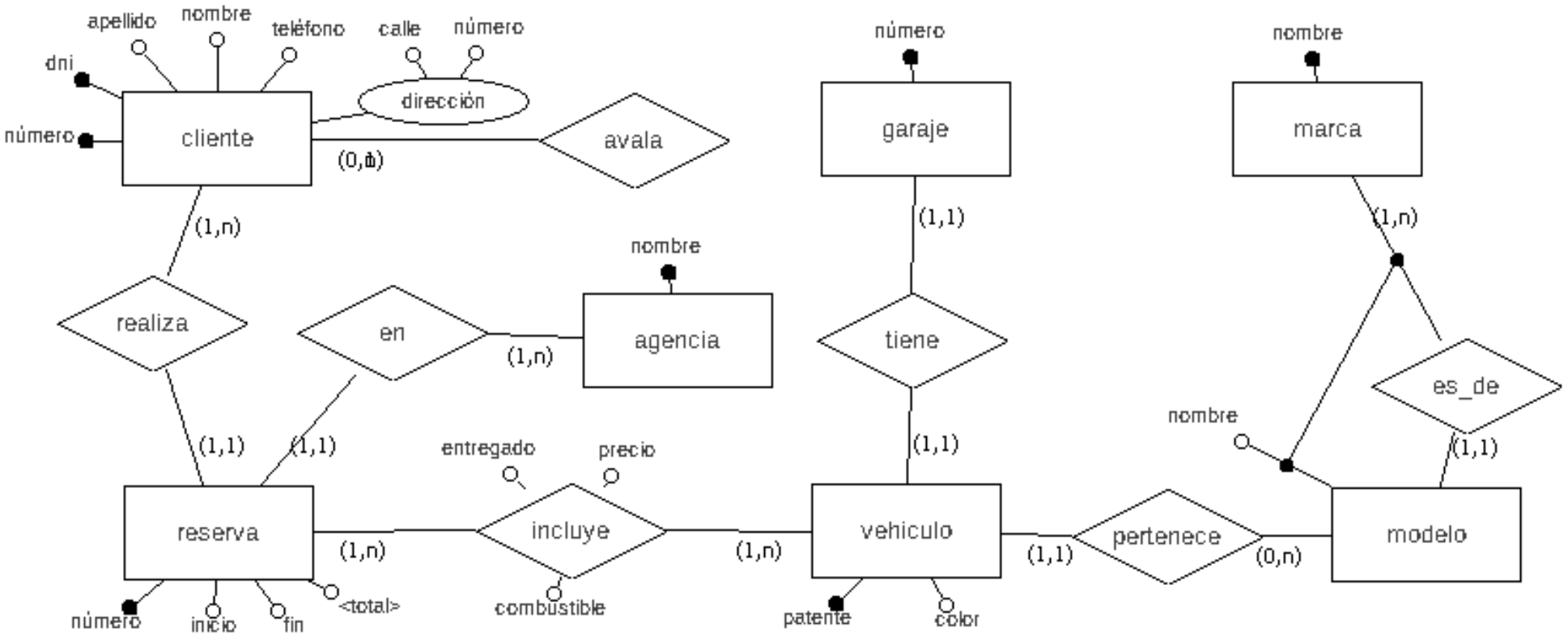
Vistas

- **Las vistas pueden ser vistas como tablas virtuales o como consultas almacenadas.**
- **Pueden utilizarse para proveer vistas de usuarios, columnas calculadas, facilitar cuestiones relacionadas a la seguridad, etc.**
- **Antes necesitamos saber de SQL/DML.**
- **Vamos a retomarlo más adelante.**

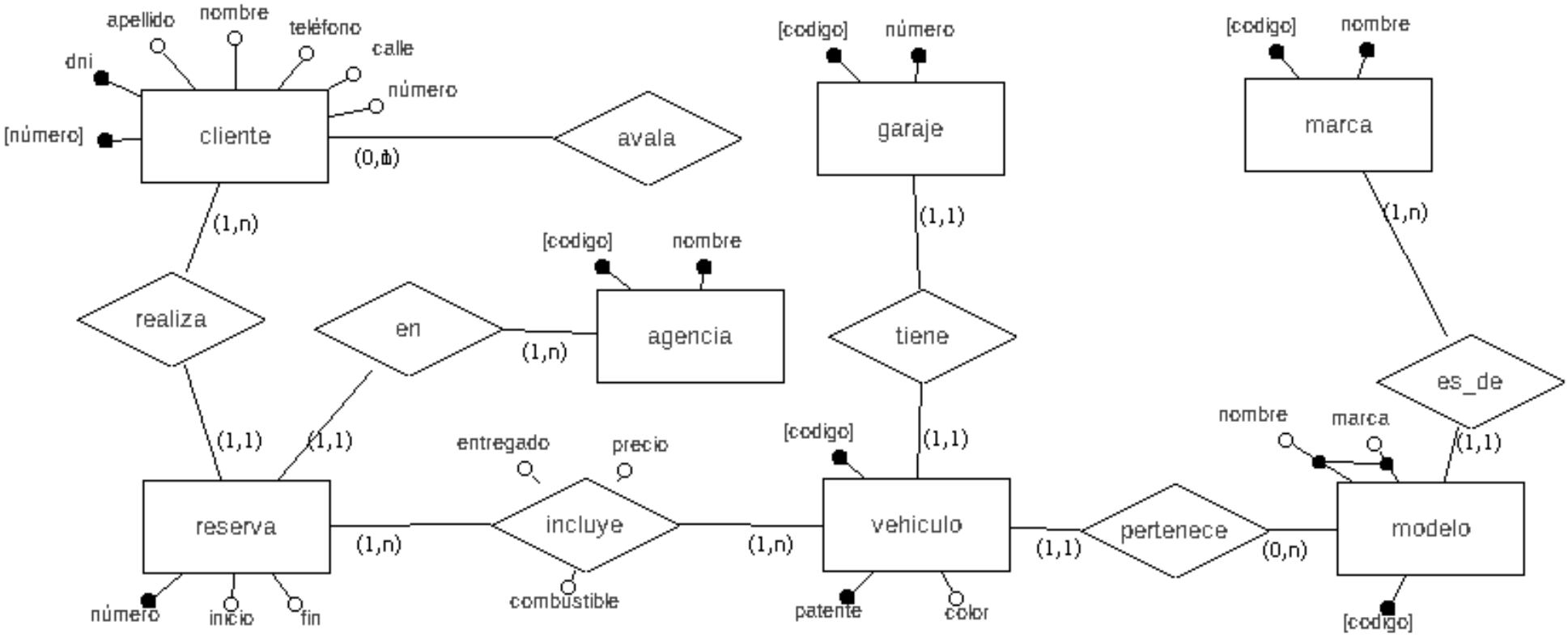
Diseño físico (parcial)

- **Vamos a definir la estructura de nuestra base de datos, para eso vamos a:**
  - Crear tipos de datos personalizados
  - Crear las tablas que componen la misma
    - Definir los tipos de datos de cada columna.
    - Establecer clave subrogadas.
    - Establecer las restricciones de clave primaria en cada tabla.
    - Establecer las restricciones de clave alternativa o secundaria.
    - Establecer otras validaciones de obligatoriedad u opcionalidad.
    - Establecer otras validaciones.
    - Establecer claves foráneas e integridad referencial.
    - Establecer políticas de actualización integridad referencial.
  - Crear otros elementos necesarios (vistas, etc)

Conceptual - Alquiler de vehículos



Lógico - Alquiler de vehículos



Físico - Alquiler de vehículos

```sql
CREATE DOMAIN TNOMBRE VARCHAR(50);

CREATE TABLE cliente(
  numero     INTEGER PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
  dni        INTEGER UNIQUE NOT NULL,
  apellido   TNOMBRE NOT NULL,
  nombre     TNOMBRE NOT NULL,
  telefono   VARCHAR(20) NOT NULL,
  dir_calle  TNOMBRE NOT NULL,
  dir_numero INTEGER NOT NULL,
  avala      INTEGER REFERENCES cliente ON UPDATE SET NULL ON DELETE SET NULL
);
```

Físico - Alquiler de vehículos

```sql
CREATE TABLE garaje(
  codigo INTEGER PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
  numero INTEGER UNIQUE NOT NULL
);

CREATE TABLE marca(
  codigo INTEGER PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
  nombre TNOMBRE UNIQUE NOT NULL
);
```

Físico - Alquiler de vehículos

```
CREATE TABLE modelo(
  codigo INTEGER PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
  nombre TNOMBRE NOT NULL,
  marca  INTEGER NOT NULL REFERENCES marca ON UPDATE CASCADE ON DELETE RESTRICT,
  UNIQUE(marca, nombre)
);

CREATE DOMAIN TCOLOR VARCHAR(6) CHECK(VALUE IN ('Blanco', 'Rojo', 'Negro', 'Gris'));
```

Físico - Alquiler de vehículos

```
CREATE TABLE vehiculo(
  codigo  INTEGER PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
  patente VARCHAR(7) UNIQUE NOT NULL,
  color   TCOLOR NOT NULL,
  garaje  INTEGER UNIQUE NOT NULL REFERENCES garaje ON UPDATE CASCADE ON DELETE RESTRICT,
  modelo  INTEGER NOT NULL REFERENCES modelo ON UPDATE CASCADE ON DELETE RESTRICT
);
```

Físico - Alquiler de vehículos

```
CREATE TABLE vehiculo(
  codigo  INTEGER PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
  patente VARCHAR(7) UNIQUE NOT NULL,
  color   TCOLOR NOT NULL,
  garaje  INTEGER UNIQUE NOT NULL REFERENCES garaje ON UPDATE CASCADE ON DELETE RESTRICT,
  modelo  INTEGER NOT NULL REFERENCES modelo ON UPDATE CASCADE ON DELETE RESTRICT
);
```

Bibliografía

- **SQL-99 Complete, Really. 1999. Peter Gulutzan y Trudy Pelzer.**
- **PostgreSQL Introduction and Concepts. 2001. Momjian, Bruce.**
- **PostgreSQL 9.6 Documentation. 2016. The PostgreSQL Global Development Group.**