

Ejercicios SQL - Definición de Datos - Solución

Ejercicio 1:

Disponemos de la siguiente Base de Datos para gestionar la información de los pubs de una determinada provincia.

PUB	TITULAR	EMPLEADO
#COD_PUB	#DNI_TITULAR	#DNI_EMPLEADO
NOMBRE	NOMBRE	NOMBRE
LICENCIA_FISCAL	DOMICILIO	DOMICILIO
DOMICILIO	COD_PUB	
FECHA_APERTURA		
HORARIO		
COD_LOCALIDAD		

EXISTENCIAS	LOCALIDAD	PUB_EMPLEADO
#COD_ARTICULO	#COD_LOCALIDAD	#COD_PUB
NOMBRE	NOMBRE	#DNI_EMPLEADO
CANTIDAD		#FUNCION
PRECIO		
COD_PUB		

Se pide escribir los comandos SQL que permitan la creación de las tablas anteriores teniendo en cuenta las siguientes restricciones:

- Todos los valores son de tipo carácter excepto los campos FECHA_APERTURA (fecha) , CANTIDAD, PRECIO y COD_LOCALIDAD (numéricos).
- Los únicos campos que no son obligatorios son los campos DOMICILIO.
- Los valores del campo horario sólo pueden ser HOR1, HOR2 y HOR3.
- No es posible dar de alta EXISTENCIAS a precio 0.
- El campo función de la tabla PUB_EMPLEADO sólo puede tener los valores CAMARERO, SEGURIDAD, LIMPIEZA.
- Se ha de mantener la integridad referencial entre las tablas.
- Las claves primarias vienen marcadas con el símbolo #.

Solución:

Sentencias SQL de creación de tablas:

```
CREATE TABLE pub (  
    cod_pub          VARCHAR(5)    NOT NULL,  
    nombre           VARCHAR(60)   NOT NULL,  
    licencia_fiscal  VARCHAR(60)   NOT NULL,  
    domicilio        VARCHAR(60)   ,  
    fecha_apertura  DATE           NOT NULL,  
    horario           VARCHAR(60)   NOT NULL,  
    cod_localidad    INTEGER        NOT NULL ) ;
```

```
CREATE TABLE titular (  
    dni_titular      VARCHAR(8)    NOT NULL,  
    nombre           VARCHAR(60)   NOT NULL,  
    domicilio        VARCHAR(60)   ,  
    cod_pub          VARCHAR(5)    NOT NULL ) ;
```

```
CREATE TABLE empleado (  
    dni_empleado     VARCHAR(8)    NOT NULL,  
    nombre           VARCHAR(60)   NOT NULL,  
    domicilio        VARCHAR(60)   ) ;
```

```
CREATE TABLE existencias (  
    cod_articulo     VARCHAR(10)   NOT NULL,  
    nombre           VARCHAR(60)   NOT NULL,  
    cantidad         INTEGER       NOT NULL,  
    precio            DECIMAL       NOT NULL,  
    cod_pub          VARCHAR(5)    NOT NULL ) ;
```

```
CREATE TABLE localidad (  
    cod_localidad    INTEGER        NOT NULL,  
    nombre           VARCHAR(60)   NOT NULL ) ;
```

```
CREATE TABLE pub_empleado (  
    cod_pub          VARCHAR(5)    NOT NULL,  
    dni_empleado     VARCHAR(8)    NOT NULL,  
    funcion          VARCHAR(9)    NOT NULL ) ;
```

Sentencias SQL de creación restricciones:

```
ALTER TABLE pub ADD CONSTRAINT pk_pub  
PRIMARY KEY (cod_pub) ;
```

```
ALTER TABLE localidad ADD CONSTRAINT pk_localidad  
PRIMARY KEY (cod_localidad) ;
```

```
ALTER TABLE titular ADD CONSTRAINT pk_titular  
PRIMARY KEY (dni_titular) ;
```

```
ALTER TABLE empleado ADD CONSTRAINT pk_empleado  
PRIMARY KEY (dni_empleado) ;
```

```
ALTER TABLE existencias ADD CONSTRAINT pk_existencias  
PRIMARY KEY (cod_articulo) ;
```

```
ALTER TABLE pub_empleado ADD CONSTRAINT pk_pub_empleado  
PRIMARY KEY (cod_pub, dni_empleado, funcion) ;
```

```
ALTER TABLE pub ADD CONSTRAINT fk_pub_localidad  
FOREIGN KEY (cod_localidad)  
REFERENCES localidad (cod_localidad) ;
```

```
ALTER TABLE titular ADD CONSTRAINT fk_titular_pu  
FOREIGN KEY (cod_pub)  
REFERENCES pub (cod_pub) ;
```

```
ALTER TABLE existencias ADD CONSTRAINT fk_existencias_pub  
FOREIGN KEY (cod_pub)  
REFERENCES pub (cod_pub) ;
```

```
ALTER TABLE pub_empleado ADD CONSTRAINT fk_pubemple_pub  
FOREIGN KEY (cod_pub)  
REFERENCES pub(cod_pub) ;
```

```
ALTER TABLE pub_empleado ADD CONSTRAINT fk_pubemple_empleado  
FOREIGN KEY (dni_empleado)  
REFERENCES empleado (dni_empleado) ;
```

```
ALTER TABLE pub ADD CONSTRAINT ck_horario  
CHECK (horario IN ('HOR1', 'HOR2', 'HOR3')) ;
```

```
ALTER TABLE existencias ADD CONSTRAINT ck_precio  
CHECK (precio <> 0) ;
```

```
ALTER TABLE pub_empleado ADD CONSTRAINT ck_funcion  
CHECK (funcion IN ('CAMARERO', 'SEGURIDAD', 'LIMPIEZA')) ;
```

Ejercicio 2:

La siguiente base de datos está pensada para almacenar la información necesaria para gestionar la venta automática de entradas para diferentes espectáculos desde múltiples puntos de venta, como pueden ser oficinas bancarias, terminales tipo Servicaixa, o las mismas taquillas de teatros u otros recintos.

ESPECTACULOS (COD_ESPECTACULO, NOMBRE, TIPO, FECHA_INICIAL, FECHA_FINAL, INTERPRETE, COD_RECINTO)

PRECIOS_ESPECTACULOS (COD_ESPECTACULO, COD_RECINTO, ZONA, PRECIO)

RECINTOS (COD_RECINTO, NOMBRE, DIRECCION, CIUDAD, TELEFONO, HORARIO)

ZONAS_RECINTOS (COD_RECINTO, ZONA, CAPACIDAD)

ASIENTOS (COD_RECINTO, ZONA, FILA, NUMERO)

REPRESENTACIONES (COD_ESPECTACULO, FECHA, HORA)

ENTRADAS (COD_ESPECTACULO, FECHA, HORA, COD_RECINTO, FILA, NUMERO, ZONA, DNI_CLIENTE)

ESPECTADORES (DNI_CLIENTE, NOMBRE, DIRECCION, TELEFONO, CIUDAD, NTARJETA)

Se pide:

1. Establecer las claves primarias de cada una de las tablas y las restricciones de integridad referencial existentes entre las mismas.
2. Crear las sentencias SQL que nos permiten crear las tablas anteriores y sus restricciones.

Solución:

El problema admite múltiples soluciones en base a los supuestos que se hagan, todas serían válidas siempre que sean coherentes con las claves elegidas. Os propongo la siguiente:

Las claves primarias que se pueden deducir son las siguientes:

ESPECTACULOS (COD_ESPECTACULO, NOMBRE, TIPO, FECHA_INICIAL, FECHA_FINAL, INTERPRETE, COD_RECINTO)

- Primary Key: COD_ESPECTACULO
- Foreign Key: COD_RECINTO

Esta elección implica que un mismo espectáculo tendrá diferentes códigos cuando se representa en las diferentes fechas y recintos.

PRECIOS_ESPECTACULOS (COD_ESPECTACULO, COD_RECINTO, ZONA, PRECIO)

- Primary Key formada por los campos: COD_ESPECTACULO, COD_RECINTO, ZONA puesto que puede haber precios diferentes para las distintas zonas (patio de butacas, palco, etc ...) del recinto dónde se celebra el espectáculo.
- Dos Foreign Key:
 - COD_ESPECTACULO
 - COD_RECINTO, ZONA

RECINTOS (COD_RECINTO, NOMBRE, DIRECCION, CIUDAD, TELEFONO, HORARIO)

- Primary Key formada por los campos: COD_RECINTO

ZONAS_RECINTOS (COD_RECINTO, ZONA, CAPACIDAD)

- Primary Key formada por los campos: COD_RECINTO, ZONA puesto que puede las distintas zonas (patio de butacas, palco, etc ...) del recinto dónde se celebra el espectáculo tienen diferentes capacidades.
- Foreign Key: COD_RECINTO

ASIENTOS (COD_RECINTO, ZONA, FILA, NUMERO)

- Primary Key formada por los campos: COD_RECINTO, ZONA, FILA, NUMERO un asiento de un recinto se identifica por la zona en la que se encuentra, su fila y su número. El asiento 4 correspondiente a la fila 2 del patio de butacas de un determinado recinto.
- Foreign Key: COD_RECINTO, ZONA

REPRESENTACIONES (COD_ESPECTACULO, FECHA, HORA)

- Primary Key formada por los campos: COD_RECINTO, FECHA, HORA puesto que puede celebrarse un mismo espectáculo el mismo día a horas diferentes.
- Foreign Key : COD_ESPECTACULO

ENTRADAS (COD_ESPECTACULO, FECHA, HORA, COD_RECINTO, FILA, NUMERO, ZONA, DNI_CLIENTE)

- Tal cómo esta conformada la tabla la Primary Key formada por los campos: COD_ESPECTÁCULO, FECHA, HORA, FILA, NUMERO, ZONA, DNI_CLIENTE

Un cliente podría comprar más de una entrada para un mismo espectáculo, que se celebra el mismo día y esas entradas podrían corresponder a la misma fila y al mismo número de butaca de diferentes zonas.

- Tres Foreign Key :
 - COD_ESPECTÁCULO
 - COD_RECINTO, ZONA, FILA, NUMERO
 - DNI_CLIENTE

ESPECTADORES (DNI_CLIENTE, NOMBRE, DIRECCIÓN, TELEFONO, CIUDAD, NTARJETA)

- Primary Key formada por el DNI_CLIENTE.

Sentencias SQL de creación de tablas:

```
CREATE TABLE espectaculos (  
    cod_espectaculo VARCHAR(8) NOT NULL,  
    nombre          VARCHAR(80) NOT NULL,  
    tipo            VARCHAR(80) NOT NULL,  
    fecha_inicial  DATE          ,  
    fecha_final    DATE          ,  
    interprete     VARCHAR(80) NOT NULL,  
    cod_recinto    VARCHAR(8)    ) ;
```

```
CREATE TABLE precios_espectaculos (  
    cod_espectaculo VARCHAR(8) NOT NULL,  
    cod_recinto     VARCHAR(8) NOT NULL,  
    zona            VARCHAR(80) NOT NULL,  
    precio          DECIMAL    NOT NULL ) ;
```

```
CREATE TABLE recintos (  
    cod_recinto    VARCHAR(8) NOT NULL,  
    nombre         VARCHAR(80) NOT NULL,  
    direccion      VARCHAR(80) NOT NULL,  
    ciudad         VARCHAR(80) NOT NULL,  
    telefono       VARCHAR(80) ,  
    horario        VARCHAR(80) NOT NULL ) ;
```

```
CREATE TABLE zonas_recintos (  
    cod_recinto    VARCHAR(8) NOT NULL,  
    zona           VARCHAR(80) NOT NULL,  
    capacidad      INTEGER    NOT NULL ) ;
```

```
CREATE TABLE asientos (  
    cod_recinto    VARCHAR(8) NOT NULL,  
    zona           VARCHAR(80) NOT NULL,  
    fila          INTEGER    NOT NULL,  
    numero        INTEGER    NOT NULL ) ;
```

```
CREATE TABLE representaciones (  
    cod_espectaculo VARCHAR(8) NOT NULL,  
    fecha           DATE      NOT NULL,  
    hora           VARCHAR(8) NOT NULL ) ;
```

```
CREATE TABLE entradas (  
    cod_espectaculo VARCHAR(8) NOT NULL,  
    fecha           DATE      NOT NULL,  
    hora           VARCHAR(8) NOT NULL,  
    cod_recinto    VARCHAR(8) NOT NULL,  
    fila           INTEGE     ,  
    numero        INTEGER    ,  
    zona           VARCHAR(80) ,  
    dni_cliente    VARCHAR(9) ) ;
```

```
CREATE TABLE espectadores (  
    dni_cliente    VARCHAR(9) NOT NULL,  
    nombre         VARCHAR(80) NOT NULL,  
    direccion      VARCHAR(80) ,  
    telefono       VARCHAR(80) ,  
    ciudad         VARCHAR(80) ,  
    ntarjeta       VARCHAR(20) NOT NULL ) ;
```

Sentencias SQL de creación de restricciones:

```
ALTER TABLE espectaculos
ADD CONSTRAINT pk_espectaculos
PRIMARY KEY (cod_espectaculo) ;
```

```
ALTER TABLE precios_espectaculos
ADD CONSTRAINT pk_precios_espectaculos
PRIMARY KEY (cod_espectaculo, cod_recinto, zona) ;
```

```
ALTER TABLE recintos
ADD CONSTRAINT pk_recintos
PRIMARY KEY (cod_recinto) ;
```

```
ALTER TABLE zonas_recintos
ADD CONSTRAINT pk_zonas_recintos
PRIMARY KEY (cod_recinto, zona) ;
```

```
ALTER TABLE asientos
ADD CONSTRAINT pk_asientos
PRIMARY KEY (cod_recinto, zona, fila, numero) ;
```

```
ALTER TABLE representaciones
ADD CONSTRAINT pk_representaciones
PRIMARY KEY (cod_espectaculo, fecha, hora) ;
```

```
ALTER TABLE entradas
ADD CONSTRAINT pk_entradas
PRIMARY KEY (cod_espectaculo, fecha, hora, fila, numero, zona, dni_cliente) ;
```

```
ALTER TABLE espectadores
ADD CONSTRAINT pk_espectadores
PRIMARY KEY (dni_cliente) ;
```

```

ALTER TABLE espectaculos
ADD CONSTRAINT fk_espectaculos_recintos
FOREIGN KEY (cod_recinto)
REFERENCES recintos (cod_recinto) ;

ALTER TABLE precios_espectaculos
ADD CONSTRAINT fk_precios_espectaculos
FOREIGN KEY (cod_espectaculo)
REFERENCES espectaculos (cod_espectaculo) ;

ALTER TABLE precios_espectaculos
ADD CONSTRAINT fk_precios_recinto
FOREIGN KEY (cod_recinto, zona)
REFERENCES zonas_recintos (cod_recinto, zona) ;

ALTER TABLE zonas_recintos
ADD CONSTRAINT fk_zonas_recintos
FOREIGN KEY (cod_recinto)
REFERENCES recintos (cod_recinto) ;

ALTER TABLE asientos
ADD CONSTRAINT fk_asientos_recintos
FOREIGN KEY (cod_recinto, zona)
REFERENCES zonas_recintos (cod_recinto, zona) ;

ALTER TABLE representaciones
ADD CONSTRAINT fk_representaciones_espectaculos
FOREIGN KEY (cod_espectaculo)
REFERENCES espectaculos (cod_espectaculo) ;

ALTER TABLE entradas
ADD CONSTRAINT fk_entradas_espectaculo
FOREIGN KEY (cod_espectaculo)
REFERENCES espectaculos (cod_espectaculo) ;

ALTER TABLE entradas
ADD CONSTRAINT fk_entradas_asientos
FOREIGN KEY (cod_recinto, zona, fila, numero)
REFERENCES asientos (cod_recinto, zona, fila, numero) ;

ALTER TABLE entradas
ADD CONSTRAINT fk_entradas_espectadores
FOREIGN KEY (dni_cliente)
REFERENCES espectadores(dni_cliente) ;

```

Ejercicio 3:

Se desea tener una base de datos que almacene la información sobre los empleados de una empresa, los departamentos en los que trabajan y los estudios de que disponen. Guardaremos el historial laboral y salarial de todos los empleados. Para ello contamos con las siguientes tablas:

EMPLEADOS

Column Name	Data Type
-----	-----
DNI	NUMBER(8)
NOMBRE	VARCHAR(10)
APELLIDO1	VARCHAR(15)
APELLIDO2	VARCHAR(15)
DIRECC1	VARCHAR(25)
DIRECC2	VARCHAR(20)
CIUDAD	VARCHAR(20)
PROVINCIA	VARCHAR(20)
COD_POSTAL	VARCHAR(5)
SEXO	VARCHAR(1)
FECHA_NAC	DATE

DEPARTAMENTOS

Column Name	Data Type
-----	-----
DPTO_COD	NUMBER(5)
NOMBRE_DPTO	VARCHAR(30)
DPTO_PADRE	NUMBER(5)
PRESUPUESTO	NUMBER
PRES_ACTUAL	NUMBER

ESTUDIOS

Column Name	Data Type
-----	-----
EMPLEADO_DNI	NUMBER(8)
UNIVERSIDAD	NUMBER(5)
AÑO	NUMBER
GRADO	VARCHAR(3)
ESPECIALIDAD	VARCHAR(20)

HISTORIAL_LABORAL

Column Name	Data Type
-----	-----
EMPLEADO_DNI	NUMBER(8)
TRABAJO_COD	NUMBER(5)
FECHA_INICIO	DATE
FECHA_FIN	DATE
DPTO_COD	NUMBER(5)
SUPERVISOR_DNI	NUMBER(8)

UNIVERSIDADES

Column Name	Data Type
-----	-----
UNIV_COD	NUMBER(5)
NOMBRE_UNIV	VARCHAR(25)
CIUDAD	VARCHAR(20)
MUNICIPIO	VARCHAR(2)
COD_POSTAL	VARCHAR(5)

HISTORIAL_SALARIAL

Column Name	Data Type
-----	-----
EMPLEADO_DNI	NUMBER(8)
SALARIO	NUMBER
FECHA_COMIENZO	DATE
FECHA_FIN	DATE

TRABAJOS

Column Name	Data Type
-----	-----
TRABAJO_COD	NUMBER(5)
NOMBRE_TRAB	VARCHAR(20)
SALARIO_MIN	NUMBER(2)
SALARIO_MAX	NUMBER(2)

Controlar las siguientes restricciones:

1. Los siguientes atributos son obligatorios:
 - NOMBRE (en todas las tablas),
 - APELLIDO1 en EMPLEADOS,
 - PRESUPUESTO en DEPARTAMENTOS,
 - SALARIO en HISTORIAL_SALARIAL y
 - SALARIO_MIN y SALARIO_MAX en TRABAJOS.

Al crear las tablas correspondientes especificar la opción NOT NULL.

```
CREATE TABLE nombre_tabla (nombre_campo TIPO NOT NULL, ... ) ;
```

Si la tabla ya estuviese creada:

```
ALTER TABLE nombre_tabla MODIFY nombre_campo TIPO NOT NULL;
```

2. El atributo SEXO en EMPLEADOS sólo puede tomar los valores H para hombre y M para mujer.

```
ALTER TABLE empleados
ADD CONSTRAINT ck_sexo
CHECK (sexo ='H' OR sexo='M') ;
```

3. Dos DEPARTAMENTOS no se llaman igual. Dos TRABAJOS tampoco.

```
ALTER TABLE departamentos
ADD CONSTRAINT uk_nombre_dpto
UNIQUE (nombre_dpto) ;
```

```
ALTER TABLE trabajos
ADD CONSTRAINT uk_nombre_trab
UNIQUE (nombre_trab) ;
```

4. Cada empleado tiene un solo salario en cada momento. También, cada empleado tendrá asignado un solo trabajo en cada momento.

```
ALTER TABLE historial_salarial
ADD CONSTRAINT pk_historial_salarial
PRIMARY KEY (empleado_dni, salario, fecha_comienzo) ;
```

```
ALTER TABLE historial_laboral
ADD CONSTRAINT pk_historial_laboral
PRIMARY KEY (empleado_dni, trabajo_cod, fecha_inicio) ;
```

5. Se ha de mantener la regla de integridad de referencia y pensar una clave primaria para cada tabla.

```
/* CLAVES PRIMARIAS */

ALTER TABLE empleados
ADD CONSTRAINT pk_empleados
PRIMARY KEY (dni) ;

ALTER TABLE historial_salarial
ADD CONSTRAINT pk_historial_salarial
PRIMARY KEY (empleado_dni, salario, fecha_comienzo) ;
ALTER TABLE historial_laboral
ADD CONSTRAINT pk_historial_laboral
PRIMARY KEY (empleado_dni, trabajo_cod, fecha_inicio) ;

ALTER TABLE departamentos
ADD CONSTRAINT pk_departamentos
PRIMARY KEY (departamento_cod) ;

ALTER TABLE estudios
ADD CONSTRAINT pk_estudios
PRIMARY KEY (empleado_dni, universidad, especialidad) ;

ALTER TABLE universidades
ADD CONSTRAINT pk_universidades
PRIMARY KEY (uni_cod) ;

ALTER TABLE trabajos
ADD CONSTRAINT pk_trabajos
PRIMARY KEY (trabajo_cod) ;

/* CLAVES AJENAS */

ALTER TABLE historial_salarial
ADD CONSTRAINT fk_historial_salarial_empleado
FOREIGN KEY (empleado_dni)
REFERENCES empleados (dni) ;

ALTER TABLE historial_laboral
ADD CONSTRAINT fk_historial_laboral_empleado
FOREIGN KEY (empleado_dni)
REFERENCES empleados (dni) ;

ALTER TABLE historial_laboral
ADD CONSTRAINT fk_historial_laboral_supervisor
FOREIGN KEY (supervisor_dni)
REFERENCES empleados (dni) ;

ALTER TABLE historial_laboral
ADD CONSTRAINT fk_historial_laboral_trabajo
FOREIGN KEY (trabajo_cod)
REFERENCES trabajos (trabajo_cod) ;

ALTER TABLE historial_laboral
ADD CONSTRAINT fk_historial_laboral_dpto
FOREIGN KEY (dpto_cod)
REFERENCES departamentos (dpto_cod) ;
```

```
ALTER TABLE departamentos
ADD CONSTRAINT fk_departamento_padre
FOREIGN KEY (dpto_padre)
REFERENCES departamentos (dpto_cod) ;

ALTER TABLE estudios
ADD CONSTRAINT fk_estudios_empleado
FOREIGN KEY (empleado_dni)
REFERENCES empleados (dni) ;

ALTER TABLE estudios
ADD CONSTRAINT fk_estudios_universidad
FOREIGN KEY (universidad)
REFERENCES universidades (univ_cod) ;
```