

PEC 3

Fecha solución: 24/04/2007

## Solución oficial

### Presentación y Objetivos

La Prueba de Evaluación Continuada III (PEC3) es la tercera de las cuatro PECs de las que consta el curso. Dicha PEC *pesará* un 25% en la nota final, y los conceptos que en ella se desarrollan corresponden a los tres primeros módulos de la asignatura, incluido el “Diseño de Bases de Datos”.

La PEC consta de tres preguntas: la primera sobre diseño e implementación, la segunda sobre manipulación de datos (incluye consultas SQL), y la tercera sobre la instalación de Apache y PHP. Para la correcta resolución de la última pregunta se deberá seguir y ejecutar el caso práctico adjunto al enunciado de la PEC. Además algunas preguntas disponen de subapartados voluntarios, para aquellos que deseen profundizar los puntos tratados.

La valoración de cada pregunta en el global de la PEC se incluye en la cabecera de cada una. Puesto que la asignatura no tiene examen, es necesario realizar todas las PECs para tener nota final.

## 1. Diseño e implementación (60%)

Siguiendo con los proyectos de los programas disponibles de la PEC1, se determina que 'Seguimiento climático' será el trabajo que aportará más resultados de interés científico. Como parece que tiene posibilidades de recibir una ayuda de la Comunidad Económica Europea, se continuará su desarrollo considerando la siguiente descripción de funcionamiento:

Se define el proyecto como un sistema que permitirá realizar el seguimiento de temperaturas en  $n$  ubicaciones. Una ubicación se designará por sus coordenadas geográficas (longitud y latitud) y tendrá un nombre asociado.

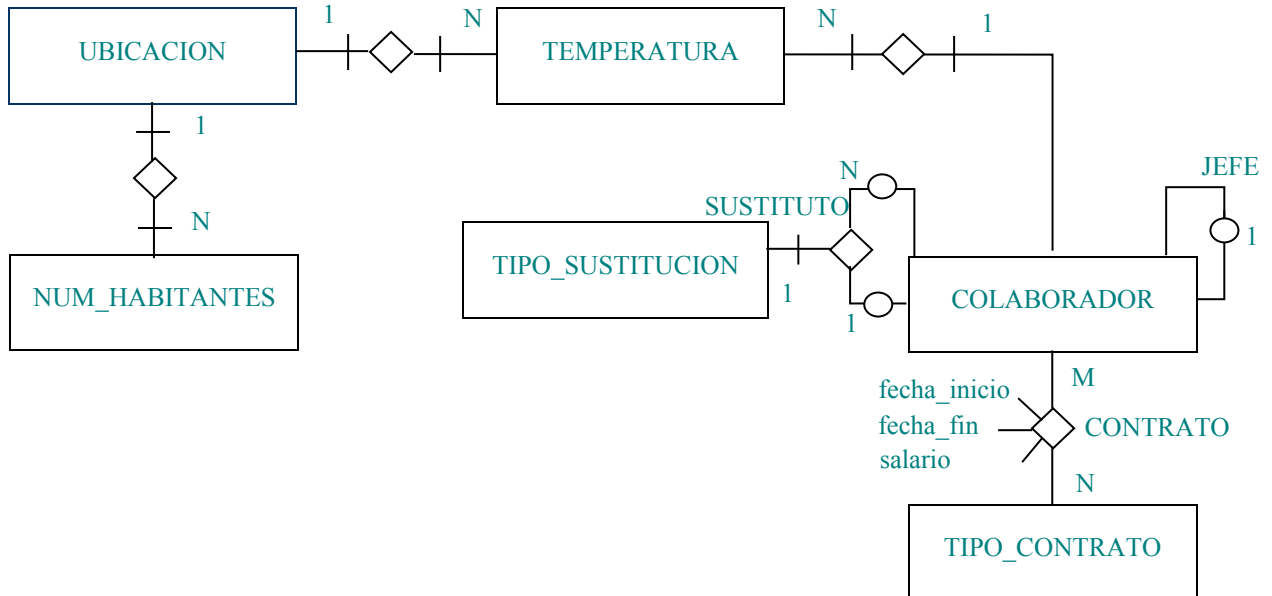
Cada ubicación que exista en el sistema tendrá un identificador único. Para cada una de ellas se podrá: controlar periódicamente el número de habitantes censados en dicha ubicación, anotar su temperatura en grados Celsius (cuatro decimales), así como la fecha y hora en que se realizó la medición.

A parte, se desea almacenar la relación de quienes suministran las temperaturas en el sistema informático, sabiendo que una medición concreta la realizará un único colaborador. Los colaboradores suelen ser estables, por lo que adicionalmente al nombre y apellidos, se desea un identificador único para cada uno de ellos. Estos colaboradores pueden tener como 'jefe responsable' a otro colaborador, pueden no tener jefe, o ellos mismos pueden ser su propio jefe. A cada uno de ellos se le realizará un contrato entre unas fechas determinadas, por lo que será necesario poder almacenar dichas fechas, así como el importe que percibirá. Existen inicialmente tres tipos distintos de contrato: 'Voluntario', 'Aprendiz' y 'Experto', no descartándose la creación de otros tipos en el futuro. A cada colaborador sólo se le puede asignar un tipo de contrato entre un par de fechas.

Finalmente, y a efectos de poder traspasar tareas de un colaborador a otro (para casos de enfermedad u otras vicisitudes), se podrá anotar para cada colaborador, un colaborador o grupo de colaboradores 'sustitutos', pudiéndose indicar para cada posible colaborador 'sustituto', el tipo de sustitución. Así, por ejemplo, el colaborador A, puede tener como sustituto el colaborador X y Z. X será de tipo 'Sustituto Certificado', mientras que Z podría ser de tipo 'Sustituto Ocasional para Emergencias' (un mismo colaborador puede tener asignados distintos tipos de sustitución, dependiendo del colaborador que sustituirá). Los distintos 'tipos de sustitución' se almacenarán en una tabla específica.

Se pide:

a. Realizar el diagrama entidad/relación para la base de datos que dará soporte al sistema informático que almacenará los datos según el modelo descrito.



b. Realizar la transformación del modelo ER al modelo relacional.

**UBICACION** (id\_ubicacion, latitud, longitud, nombre)

**NUM\_HABITANTES** (id\_ubicacion, fecha, num\_habitantes)  
donde {id\_ubicacion} referencia UBICACION

**TEMPERATURA** (id\_ubicacion, fecha, hora, id\_colaborador, graus)  
donde {id\_ubicacion} referencia UBICACION  
donde {id\_colaborador} referencia COLABORADOR

**COLABORADOR** (id\_colaborador, id\_jefe, nombre, apellidos)  
donde {id\_jefe} referencia COLABORADOR

**TIPO\_SUSTITUCION** (id\_tipo\_sustitucion, descripcion)

**SUSTITUTO** (id\_colaborador, id\_sustituto, id\_tipo\_sustitucion)  
donde {id\_colaborador} referencia COLABORADOR  
donde {id\_sustituto} referencia COLABORADOR  
donde {id\_tipo\_sustitucion} referencia TIPO\_SUSTITUCION

**TIPO\_CONTRATO** (id\_tipo\_contrato, descripcion)

**CONTRATO** (id\_colaborador, id\_tipo\_contrato, fecha\_inicio, fecha\_fin, salario)  
donde {id\_colaborador} referencia COLABORADOR  
donde {id\_tipo\_contrato} referencia TIPO\_CONTRATO

c. Crear las tablas según atributos, claves (primarias y foráneas) y restricciones que se entrevean. Adjuntar el código SQL (PostgreSQL) utilizado.

```
CREATE TABLE COLABORADOR (
  ID_COLABORADOR SMALLINT NOT NULL,
  IDENT_USUARIO VARCHAR(20) NOT NULL,
  NOMBRE VARCHAR(20) NOT NULL,
  APELLIDO1 VARCHAR(25) NOT NULL,
  APELLIDO2 VARCHAR(25),
  SEXO CHAR(1) NOT NULL,
  FECHA_NACIMIENTO DATE,
  ID_JEFE SMALLINT DEFAULT NULL,
  CONSTRAINT PK_COLABORADOR PRIMARY KEY (ID_COLABORADOR),
  CONSTRAINT FK_ID_JEFE FOREIGN KEY (ID_JEFE) REFERENCES COLABORADOR
(ID_COLABORADOR),
  CHECK (SEXO = 'H' OR SEXO = 'M')
);
```

```
CREATE TABLE TIPO_CONTRATO (
  ID_TIPO_CONTRATO SMALLINT NOT NULL,
  DESCRIPCION VARCHAR(40) NOT NULL,
  CONSTRAINT PK_TIPO_CONTRATO PRIMARY KEY (ID_TIPO_CONTRATO)
);
```

```
CREATE TABLE CONTRATO (
  ID_CONTRATO SMALLINT NOT NULL,
  ID_COLABORADOR SMALLINT NOT NULL,
  FECHA_INICIO_CONTRATO DATE NOT NULL,
  FECHA_FIN_CONTRATO DATE,
  SALARIO DECIMAL (4,2),
  CONSTRAINT PK_CONTRATO PRIMARY KEY (ID_CONTRATO, ID_COLABORADOR,
FECHA_INICIO_CONTRATO),
  CONSTRAINT FK_ID_CONTRATO FOREIGN KEY (ID_CONTRATO) REFERENCES TIPO_CONTRATO
(ID_TIPO_CONTRATO),
  CONSTRAINT FK_ID_COLABORADOR FOREIGN KEY (ID_COLABORADOR) REFERENCES COLABORADOR
(ID_COLABORADOR),
  CHECK (FECHA_INICIO_CONTRATO < FECHA_FIN_CONTRATO)
);
```

```
CREATE TABLE TIPO_SUSTITUCION (
  ID_TIPO_SUSTITUCION SMALLINT NOT NULL PRIMARY KEY,
  DESCRIPCION VARCHAR(40) NOT NULL
);
```

```
CREATE TABLE SUSTITUTO (
  ID_COLABORADOR SMALLINT NOT NULL,
  ID_SUSTITUTO SMALLINT NOT NULL,
  ID_TIPO_SUSTITUCION SMALLINT NOT NULL,
  CONSTRAINT PK_SUSTITUTO PRIMARY KEY (ID_COLABORADOR, ID_SUSTITUTO,
ID_TIPO_SUSTITUCION),
```

```

        CONSTRAINT FK_SUS_ID_COLABORADOR FOREIGN KEY (ID_COLABORADOR) REFERENCES
COLABORADOR (ID_COLABORADOR),
        CONSTRAINT FK_SUS_SUSTITUTO FOREIGN KEY (ID_SUSTITUTO) REFERENCES COLABORADOR
(ID_COLABORADOR),
        CONSTRAINT FK_TIPO_SUSTITUCION FOREIGN KEY (ID_TIPO_SUSTITUCION) REFERENCES
TIPO_SUSTITUCION (ID_TIPO_SUSTITUCION),
        CHECK (ID_COLABORADOR <> ID_SUSTITUTO)
    );

```

```

CREATE TABLE UBICACION (
    ID_UBICACION INT4 NOT NULL,
    NOMBRE VARCHAR(60),
    UFI CHAR(8) NOT NULL,
    UNI CHAR(8) NOT NULL,
    LAT DECIMAL(9,7) NOT NULL,
    LONG DECIMAL(9,7) NOT NULL,
    CONSTRAINT PK_UBICACION PRIMARY KEY (ID_UBICACION),
    CONSTRAINT AK_UBICACION UNIQUE (UFI, UNI)
);

```

```

CREATE TABLE NUM_HABITANTES (
    ID_UBICACION INT4 NOT NULL,
    FECHA DATE NOT NULL,
    NUM_HABITANTES DECIMAL(8) NOT NULL,
    CONSTRAINT PK_NUM_HABITANTES PRIMARY KEY (ID_UBICACION, FECHA),
    CONSTRAINT FK_NUM_HABITANTES FOREIGN KEY (ID_UBICACION) REFERENCES UBICACION
(ID_UBICACION)
);

```

```

CREATE TABLE TEMPERATURA (
    ID_UBICACION INT4 NOT NULL,
    FECHA DATE NOT NULL,
    HORA TIME NOT NULL,
    ID_COLABORADOR SMALLINT NOT NULL,
    GRADOS DECIMAL(6,4) NOT NULL,
    CONSTRAINT PK_TEMPERATURA PRIMARY KEY (ID_UBICACION, FECHA, HORA),
    CONSTRAINT FK_TEMPERATURA FOREIGN KEY (ID_UBICACION) REFERENCES UBICACION
(ID_UBICACION),
    CONSTRAINT FK_COLABORADOR FOREIGN KEY (ID_COLABORADOR) REFERENCES COLABORADOR
(ID_COLABORADOR)
);

```

### Ejercicio voluntario:

Repetir las sentencias de creación de las tablas con la sintaxis SQL de MySQL con InnoDB.

## 2. Manipulación y consultas (25%)

a. Considerando la estructura de tablas del primer ejercicio, el contenido de los ficheros de datos adjuntos al enunciado de la PEC y el contenido de la tabla `LOAD_CIUDADES_COORDENADAS`, poblad de datos las tablas `UBICACION`, `COLABORADOR` y `TEMPERATURA`.

### Indicaciones para la carga de UBICACION:

Cargad los campos de UBICACION directamente de los datos contenidos en la tabla LOAD\_CIUDADES\_COORDENADAS (es importante almacenar los cuatro campos referentes a coordenadas; UFI, UNI, LAT y LONG).

```
INSERT INTO UBICACION (
  SELECT ID_CIUADAD, FULL_NAME_ND, UFI, UNI, LAT, LONG
  FROM LOAD_CIUDADES_COORDENADAS
);
```

### Indicaciones para la carga de COLABORADOR:

Se considera que el fichero de datos de COLABORADORES ha sido obtenido de 'otra aplicación', por lo que el formato del fichero es 'comma delimited'. Adecuad los parámetros de la sentencia de carga a dicho formato.

```
COPY COLABORADOR FROM 'c:/colaboradores.txt' USING DELIMITERS ',' WITH NULL AS 'null';
```

### Indicaciones para la carga de TEMPERATURA:

Cread una tabla intermedia llamada LOAD\_TEMPERATURAS. Utilizando el comando COPY insertad los datos del fichero 2006\_temperaturas.txt y de 2007\_temperaturas.txt. Realizad la inserción final de datos en TEMPERATURA desde dicha tabla, considerando que UFI y UNI es una clave alternativa (que permite obtener ID\_UBICACION).

#### *Creamos la tabla de carga LOAD\_TEMPERATURAS:*

```
CREATE TABLE LOAD_TEMPERATURAS (
  UFI CHAR(8) NOT NULL,
  UNI CHAR(8) NOT NULL,
  GENERIC VARCHAR(40),
  SORT_NAME VARCHAR(50),
  ID_PERSONA DECIMAL(3) NOT NULL,
  GRADOS DECIMAL(8,6) NOT NULL,
  HORA TIME NOT NULL,
  FECHA DATE NOT NULL,
  CONSTRAINT PK_LOAD_TEMPERATURAS PRIMARY KEY (UFI, UNI, FECHA, HORA)
);
```

#### *Realizamos la carga de datos desde el fichero de texto:*

```
COPY LOAD_TEMPERATURAS FROM 'c:/2006_temperaturas.txt' WITH DELIMITER '\t';
COPY LOAD_TEMPERATURAS FROM 'c:/2007_temperaturas.txt' WITH DELIMITER '\t';
```

#### *Traspasamos los datos a la tabla TEMPERATURAS:*

```
INSERT INTO TEMPERATURA
  SELECT UBI.ID_UBICACION, LT.FECHA, LT.HORA, LT.ID_PERSONA, LT.GRADOS
  FROM LOAD_TEMPERATURAS LT, UBICACION UBI
  WHERE LT.UNI = UBI.UNI AND LT.UFI = UBI.UFI;
```

b. Realizad la siguiente consulta:

- Retornar los grados de temperatura, la latitud (LAT), la longitud (LONG), el nombre de la ubicación, la fecha y la persona que tomó la temperatura del sitio que de entre los que hayan tenido la temperatura más alta, esté más al norte de la península ibérica.

```
SELECT TE.GRADOS, UBI.LAT, UBI.LONG, UBI.NOMBRE, TE.FECHA,  
       COL.APELLIDO1||' '||COL.APELLIDO2||', '||COL.NOMBRE AS COLABORADOR  
FROM  
       UBICACION UBI, TEMPERATURA TE, COLABORADOR COL  
WHERE  
       TE.ID_COLABORADOR = COL.ID_COLABORADOR AND  
       TE.ID_UBICACION = UBI.ID_UBICACION  
ORDER BY TE.GRADOS DESC, UBI.LAT ASC  
LIMIT 1;
```

*El resultado es:*

*34.900000;27.700000;-18.100000;"Playas de los Mozos";"2007-02-05"; "Hernández Alasa, Jaime"*

### 3. Instalación (15%)

Instalad Apache y PHP según las indicaciones de instalación adjuntas al enunciado de la PEC. Anotad las posibles incidencias, describiendo de ser necesario el entorno utilizado (Debian, RedHat, Windows, etc),

Seguir las indicaciones del documento 'Ejemplo PHP+PostgreSQL', hasta ejecutar el programa 'citascitables.php'. Incluid un par de capturas de pantalla para demostrar su funcionamiento e indicad brevemente las incidencias/dificultades de todo el proceso.

### Formato de entrega

El documento a entregar consistirá en un fichero con las siguientes características:

- Formato documento: ODT (preferente), PDF, DOC, RTF o SXW
- Nombre del documento: BD\_PEC3\_Apellido1\_Nombre.extensión  
(p. ejemplo: BD\_PEC3\_Hernández\_Juan.odt)
- Nombre del curso y nombre y apellidos estudiante en la página inicial o portada.

De ser necesario adjuntar ficheros adicionales, se entregará todo en un único fichero comprimido (tipo zip o rar), de nombre BD\_PEC3\_Apellido1\_Nombre.

El documento se entregará a través del apartado de la asignatura: Entrega de Actividades

La fecha de entrega máxima es el **jueves día 19/04/2007**.