DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico Sistema de Parques Naturales

7 de Junio de 2019 Bases de Datos

Integrante	LU	Correo electrónico
Sebastián Fernández Ledesma	392/06	sfernandezledesma@gmail.com
Brian Goldstein	27/14	brai.goldstein@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - Pabellón I
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Argentina
Tel/Fax: (54 11) 4576-3359
http://exactas.uba.ar

Índice

1. Introducción

1.1. Consideraciones Generales

En este trabajo diseñamos e implementamos una base de datos relacional para un sistema de parques naturales (en el Apéndice se encuentra el enunciado completo de problema).
El objetivo del informe será detallar el proceso para esta implementación, comenzando con la interpretación del problema, que desarrollaremos a continuación. Continuaremos en la siguiente sección con la etapa de diseño, la que a su vez se compone en una primera sub-etapa en la que planteamos un Diagrama de Entidad-Relacion (DER) basados en la interpretación del enunciado y una segunda etapa en la que se plantea el esquema de MR (modelo relacional) basado en este diagrama. Por último se explicarán las consideraciones principales a la hora de implementar el modelo en el motor elegido (PostgreSQL) y las herramientas particulares con las que se implementó lo pedido, también se exhibirán las consultas pedidas y los comportamientos esperados que fueron implementados.

1.2. De la interpretación del problema

Se nos pide modelar la base de datos que guardará la información referente a diferentes consideraciones de los parques nacionales. Estas consideraciones si bien están relacionadas entre sí (y por lo tanto no hay una distinción concreta y dura), pueden dividirse en cuatro categorías.

- **Lo referente a lo territorial/provincial:** De esta primer categoría se nos explica las características a respaldar de las provincias (con sus correspondientes organismos) y los parques con sus respectivas áreas. A destacar del enunciado en esta categoría están las consideraciones de que es necesario responder rápido la superficie por provincia (lo que nos da la pauta que seguramente esta característica será una atributo de la entidad) y por otra parte está la consideración de que los parques pueden ser compartidos entre provincias.

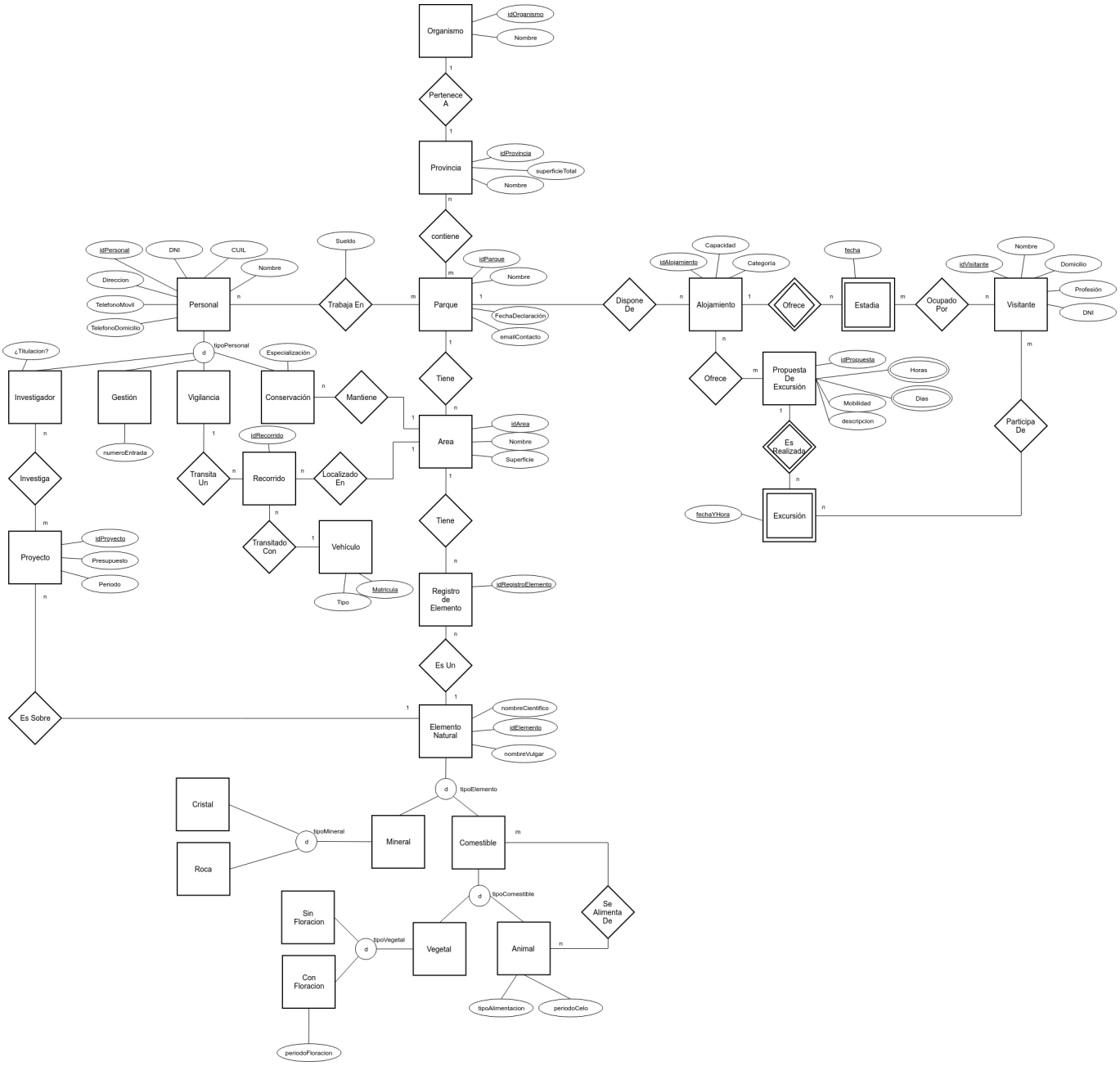
- **Lo referente a los elementos naturales:** Se detalla los diferentes tipos de elementos naturales y diferentes atributos para cada tipo así como se establece la relación de la cadena alimenticia en los elementos comestibles (animales o vegetales) que pueden ser comidos por un animal. También se establece la necesidad de poder identificar en qué áreas se ha encontrado cada elemento (lo que nos sugiere que este registro de un elemento tendrá que ser persistido).

- **Lo referente al personal:** Se explica lo referente a los diferentes puestos de trabajo dentro del parque y para cada categoría se dan los datos importantes a registrar, a cada categoría se la relaciona con aquellos recursos del parque con los que trabaja. De este modo relaciona al personal investigador con los elementos naturales (a través de sus proyectos), al personal de vigilancia con los vehículos que utiliza y las áreas recorridas, al personal de mantenimiento con las áreas que mantiene y al personal de gestión con la entrada a la que fue asignado.

- **Lo referente al alojamiento, visitantes y excursiones:** Se explica lo referente a los visitantes, estos se hospedan en diferentes alojamientos que contratan por un período determinado (seguramente querramos modelar estos contratos), y se explican las excursiones que son otro servicio que se le ofrece a la visita mediante un contrato así que seguramente la estructura servicio-contrato-visitante sea similar en ambos casos.

2. Diagrama Entidad-Relación

2.1. Diagrama



2.2. Restricciones Adicionales

- No hay 2 Recorridos con igual área y distinto vehículo
- El visitante que participa en una excursión tiene que tener una estadía cuya fecha sea la de la excursión

3. Modelo Relacional

3.1. MR

Organismo(IdOrganismo, Nombre)
PK = { IdOrganismo }
FK = { }

Provincia(IdProvincia, Nombre, SuperficieTotal, IdOrganismo)
PK = { IdProvincia }
FK = { IdOrganismo}

Parque(IdParque, Nombre, FechaDeclaracion, EmailContacto)
PK = { IdParque}
FK = { }

Contiene(IdProvincia, IdParque)
PK = { (IdProvincia, IdParque) }
FK = { IdProvincia, IdParque }

Area(IdArea, Nombre, Superficie, IdParque)
PK = { IdArea }
FK = { IdParque }

ElementoNatural(IdElemento, nombreCientifico, nombreVulgar, tipoElemento)
PK = { IdElemento }
FK = { }

Mineral(IdElemento, tipoMineral)
PK = { IdElemento }
FK = { IdElemento }

Comestible(IdElemento, tipoComestible)
PK = { IdElemento }
FK = { IdElemento }

Vegetal(IdElemento, tipoVegetal)
PK = { IdElemento }
FK = { IdElemento }

ConFloracion(IdElemento, periodoFloracion)
PK = { IdElemento }
FK = { IdElemento }

Animal(IdElemento, tipoAlimentacion, periodoCelo)
PK = { IdElemento }
FK = { IdElemento }

SeAlimentaDe(IdDepredador, IdPresa)
PK = { (IdDepredador, IdPresa) }
FK = { IdDepredador, IdPresa }

RegistroElemento(IdRegistroElemento, IdElemento, IdArea)
PK = { IdRegistroElemento }
FK = { IdElemento, IdArea }

Personal(IdPersonal, tipoPersonal, DNI, CUIL, Nombre, Direccion, TelefonoMovil, TelefonoDomicilio)
PK = { IdPersonal }
FK = { }

TrabajaEn(IdPersonal, IdParque, Sueldo)
PK = { (IdPersonal, IdParque) }
FK = { IdPersonal, IdParque }

Investigador(IdPersonal, Titulacion)
PK = { IdPersonal }
FK = { IdPersonal }

Proyecto(IdProyecto, Presupuesto, Periodo, IdElementoNatural)
PK = { IdProyecto }
FK = { IdElementoNatural }

Investiga(IdInvestigador, IdProyecto)
PK = { (IdInvestigador, IdProyecto) }
FK = { IdInvestigador, IdProyecto }

Gestion(IdPersonal, numeroEntrada)
PK = { IdPersonal }
FK = { IdPersonal }

Recorrido(IdRecorrido,IdPersonalVigilancia, IdArea, MatriculaVehiculo)
PK = { IdRecorrido }
FK = { IdPersonalVigilancia, IdArea, MatriculaVehiculo }

Vehiculo(Matricula, Tipo)
PK = { Matricula }
FK = { }

Conservacion(IdPersonal, Especializacion, IdArea)
PK = { IdPersonal }
FK = { IdPersonal }

Alojamiento(IdAlojamiento, Capacidad, Categoria, IdParque)
PK = { IdAlojamiento }
FK = { IdParque }

Estadia(IdAlojamiento, Fecha)
PK = { (IdAlojamiento, Fecha) }
FK = { IdAlojamiento }

Visitante(IdVisitante, Nombre, Domicilio, Profesion, DNI)
PK = { IdVisitante }
FK = { }

OcupadoPor(IdAlojamiento, FechaEstadia, IdVisitante)
PK = { (IdAlojamiento, FechaEstadia, IdVisitante) }
FK = { IdAlojamiento, FechaEstadia, IdVisitante }

PropuestaDeExcursion(IdPropuesta, Mobilidad, Descripcion)
PK = { IdPropuesta }
FK = { }

PropuestaHoras(IdPropuesta, Hora)
PK = { (IdPropuesta, Hora) }
FK = { IdPropuesta }

PropuestaDias(IdPropuesta, Dia)
PK = { (IdPropuesta, Dia) }
FK = { IdPropuesta }

Ofrece(IdAlojamiento, IdPropuesta)
PK = { (IdAlojamiento, IdPropuesta) }
FK = { IdAlojamiento, IdPropuesta }

Excursion(IdPropuesta, FechaYHora)
PK = { (IdPropuesta, fechaYHora) }
FK = { IdPropuesta }

ParticipaDe(IdVisitante, IdPropuesta, FechaPropuesta)
PK = { (IdVisitante, IdPropuesta, FechaPropuesta) }
FK = { IdVisitante, IdPropuesta, FechaPropuesta }

3.2. Restricciones Adicionales

- ElementoNatural.tipoElemento ∈ { "Mineral", "Comestible"}
- Mineral.tipoMineral ∈ {"Cristal", "Roca"}
- Comestible.tipoComestible ∈ {"Vegetal", "Animal"}
- Vegetal.tipoVegetal ∈ {"SinFloracion", "ConFloracion"}
- Recorrrido.IdPersonalVigilancia tiene que corresponder a una entrada en personal con tipoPersonal = "vigilancia"
- SeAlimentaDe.IdDepredador es PK de Animal, SeAlimentaDe.IdPresa es PK de Comestible
- tipoPersonal ∈ {"Investigador", "Gestion", "Vigilancia", "Conservacion"}

4. Implementación

4.1. Diseño físico

Para implementar la base de datos planteada en el modelo relacional, utilizamos el motor PostgreSQL. Por cada tabla y campo planteados en el MR creamos una tabla en esta base de datos y un campo del mismo nombre. La única excepción a esto fue el campo fechaYhora (tabla Excursión) que para poder expresarlo con los tipo provistos por el motor se decidió mapear esa columna de MR como 2 columnas separadas en la base de datos. Adicionalmente se exigió integridad referencial, para que las claves foráneas de una tabla existan efectivamente en la tabla correspondiente, es decir, en la tabla en las cuales dicha clave foránea es clave primaria.

4.2. Consultas pedidas

4.2.1. ¿Cúal es la provincia con más parques naturales?

```
--- CREO UNA VIEW CON CANTIDAD DE PARQUES POR PROVINCIA
CREATE OR REPLACE VIEW provincia_cantparques AS
SELECT p.idProvincia,p.nombre,COUNT(DISTINCT pq.idParque) cantparques FROM Provincia p
INNER JOIN Contiene c ON c.idProvincia = p.idProvincia
INNER JOIN Parque pq ON pq.idParque = c.idParque
GROUP BY p.idProvincia;

--- USO ESA VIEW PARA RESPONDER LA CONSULTA
SELECT nombre FROM provincia_cantparques
WHERE cantparques = (SELECT MAX(cantparques) from provincia_cantparques);
```

4.2.2. ¿Qué especies vegetales se encuentran en al menos la mitad de los parques?

```
SELECT e.idElemento, e.nombreVulgar FROM Parque p
INNER JOIN Area a ON a.idParque = p.idParque
INNER JOIN RegistroElemento r ON a.idArea = r.idArea
INNER JOIN ElementoNatural e ON e.idElemento = r.idElemento
INNER JOIN Vegetal v ON v.idElemento = e.idElemento
GROUP BY e.idElemento
HAVING COUNT(DISTINCT p.idParque)  > ((SELECT COUNT(*) FROM Parque) / 2);
```

4.2.3. ¿Cuántos visitantes estuvieron en los parques cuyos códigos son A y B?

```
SELECT COUNT(DISTINCT v.idVisitante) FROM Alojamiento a
INNER JOIN Estadia e  ON a.idAlojamiento = e.idAlojamiento
INNER JOIN OcupadoPor o ON e.idAlojamiento = o.idAlojamiento AND e.fecha = o.fechaEstadia
INNER JOIN Visitante v ON v.idVisitante = o.idVisitante
WHERE a.idParque = 3 OR a.idParque = 1;
```

4.3. Trigger de disminución de especie

Uno de los requerimientos es que cuando disminuya la cantidad de alguna especie, se envíe un email a la dirección de contacto del parque correspondiente. Para resolver esto, creamos un trigger que se activa antes de borrarse una fila de RegistroElemento, que hace dos tareas:

1. Agrega los datos de la fila de RegistroElemento a ser borrada a una tabla especial llamada ElementosPerdidos, que además tiene un atributo booleano EmailEnviado, que por default es FALSE, y se seteará en TRUE luego de haberse enviado el email (por otra aplicación, no por la base de datos).

2. Envía una notificación con la IdPerdida (PK de ElementosPerdidos) por medio de pg_notify, que será atendida por un daemon encargado de enviar los emails.

A continuación mostramos el código de la tabla ElementosPerdidos y del trigger:

```
CREATE TABLE ElementosPerdidos (
  IdPerdida SERIAL NOT NULL,
  IdRegistroElemento INTEGER NOT NULL,
  IdElemento INTEGER NOT NULL,
  IdArea INTEGER NOT NULL,
  EmailEnviado BOOLEAN NOT NULL DEFAULT FALSE,
  PRIMARY KEY (IdPerdida),
  FOREIGN KEY (IdElemento) REFERENCES ElementoNatural(IdElemento),
  FOREIGN KEY (IdArea) REFERENCES Area(IdArea)
);

CREATE OR REPLACE FUNCTION perdida_elemento() RETURNS TRIGGER AS $$
DECLARE
  IdPerdida INTEGER;
BEGIN
  INSERT INTO ElementosPerdidos(IdRegistroElemento, IdElemento, IdArea)
  VALUES (OLD.IdRegistroElemento, OLD.IdElemento, OLD.IdArea)
  RETURNING ElementosPerdidos.IdPerdida INTO IdPerdida;

  perform pg_notify('perdida_elemento', IdPerdida::TEXT);
  RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_perdida_elemento
BEFORE DELETE
ON RegistroElemento
FOR EACH ROW
EXECUTE PROCEDURE perdida_elemento();
```

El daemon fue programado en Python y simula el envío de emails (en la práctica usamos un server local SMTP para testear). Cuando se ejecuta lo primero que hace es conectarse a la base y chequear la tabla ElementosPerdidos por filas que tengan el atributo EmailEnviado=FALSE, para luego hacer el envío de emails a las direcciones correspondientes. Esto evita que no se envíen emails por una eventual caída del daemon. Además, escucha los pg_notify de la base de datos, y envía cada email en el momento, actualizando la tabla ElementosPerdidos con EmailEnviado=TRUE para cada envío. El código es el siguiente:

```
import os
import select
import psycopg2
import psycopg2.extensions
from dotenv import load_dotenv
import smtplib

load_dotenv()
PORT=1025
DB_NAME=os.getenv("DB_NAME")
DB_HOST=os.getenv("DB_HOST")
DB_USER=os.getenv("DB_USER")
DB_PASSWORD=os.getenv("DB_PASSWORD")
EMAIL_SENDER="admin@gmail.com"

dbc = psycopg2.connect(database=DB_NAME, host=DB_HOST, user=DB_USER, password=DB_PASSWORD)
dbc.set_isolation_level(psycopg2.extensions.ISOLATION_LEVEL_AUTOCOMMIT)
cur = dbc.cursor()

def mostrar_y_enviar_email(row):
  IdRegistroElemento = row[0]
  nombreCientifico = row[1]
  IdPerdida = row[2]
  EmailContacto = row[3]
  msg = """From: Admin <%s>
  To: Contacto Parque <%s>
  Subject: Notificacion Perdida Especimen

  Se perdio un especimen de la siguiente especie: %s (id=%s)""" % (EMAIL_SENDER, EmailContacto,
      nombreCientifico, IdRegistroElemento)
  print(msg)
  server = smtplib.SMTP("localhost", PORT)
  server.sendmail(EMAIL_SENDER, EmailContacto, "\n" + msg)
  server.quit()
  cur.execute("UPDATE ElementosPerdidos SET EmailEnviado=TRUE WHERE IdPerdida=%s;", [IdPerdida])

cur.execute("LISTEN perdida_elemento;")
cur.execute("SELECT IdRegistroElemento, nombreCientifico, IdPerdida, EmailContacto FROM
    ElementosPerdidos NATURAL JOIN ElementoNatural NATURAL JOIN Area JOIN Parque ON Area.IdParque
    =Parque.IdParque WHERE NOT EmailEnviado;")
results = cur.fetchall()
for row in results:
  mostrar_y_enviar_email(row)

while 1:
  if not select.select([dbc], [], [], 5) == ([], [], []):
    dbc.poll()
    while dbc.notifies:
      notify = dbc.notifies.pop()
      cur.execute("SELECT IdRegistroElemento, nombreCientifico, IdPerdida, EmailContacto FROM
          ElementosPerdidos NATURAL JOIN ElementoNatural NATURAL JOIN Area JOIN Parque ON Area.
          IdParque=Parque.IdParque WHERE IdPerdida = %s;", [notify.payload])
      row = cur.fetchone()
      mostrar_y_enviar_email(row)
```

5. Conclusiones

En un análisis integral de la solución podemos destacar aspectos negativos y positivos de la implementación y en general de las herramientas utilizadas: DER, MR, PostgreSQL.
De los aspectos positivos del DER y su correspondiente mecanismo de traducción a MR observamos que es una herramienta versátil en el sentido que si bien la base de datos fue planteada con ciertas consultas en mente, muy probablemente el esquema planteado en la solución funcionaría sin problemas y sin requerir modificaciones estructurales para un amplio panorama de posibles consultas, lo que puede ser muy útil a la hora de desarrollar algún sistema, dado que los requerimientos sobre las consultas pueden cambiar frecuentemente con el tiempo y con el transcurso de desarrollo o incluso con ajustes en el modelo de negocio.
Otro aspecto positivo que podemos destacar del procedimiento es que el método necesariamente lleva a que el MR resultante esté en tercera forma normal y por tanto se evita asi redundancia de datos y demás beneficios propios de esta forma.
Entre los aspectos negativos encontramos que en el procedimiento DER→MR→baseSQL sucede que el diseño condiciona totalmente la implementación, es decir, una vez planteado un diseño teórico de las relaciones entre los datos (DER) queda condicionada la implementación y por tanto es difícil aplicar consideraciones de performance. Seguramente en casos donde es crítica la performance será deseable desnormalizar a costo de tener redundancia de datos pero ganar en velocidad y a costo también de que la base de datos no sea un traslado fiel del modelo planteado en el DER. En la misma línea de pensamiento, si la performance es algo crítico quizás sea conveniente aplicar algun procedimiento para modelar los datos que parta de las consultas a responder y no de la estructura con la que respresentan los datos en la realidad, obviamente para esto sería necesario saber de entrada las consultas y como ya lo mencionamos antes, hay muchas situaciones en las que las consultas no puede saberse de antemano.
Por estas observaciones concluimos en lo referente al procedimiento (DER→MR→baseSQL) que es ideal para aquellos casos en los que se tiene una idea de los datos del negocio pero no se tiene una certeza clara de cómo estos datos serán consultados, o bien se sabe que las consultas podrían cambiar y por otro lado la performance no es algo crítico, observamos que éstas son las condiciones de la mayoria de casos de uso.

6. Apéndice: Enunciado del Problema

Se desea crear un sistema que almacena información sobre los parques naturales gestionados por las provincias.

Se sabe lo siguiente: Una Provincia puede tener varios parques naturales. En toda Provincia existe uno y sólo un organismo responsable de los parques. Un parque puede estar compartido por más de una Provincia.

Un parque natural se identifica por un nombre, fue declarado en una fecha, tiene un email de contacto, se compone de varias áreas identificadas por un nombre y caracterizadas por una determinada extensión.

Por motivos de eficiencia se desea favorecer las consultas referentes al número de parques existentes en cada Provincia y la superficie total declarada parque natural en cada Provincia.

En cada área forzosamente residen elementos naturales que pueden ser de tres tipos: vegetales, animales y minerales. Cada elemento natural tiene una denominación científica, una denominación vulgar y un número inventariado de individuos por área. De los elementos vegetales se desea saber si tienen floración y en qué periodo se produce ésta; de los animales se desea saber su tipo de alimentación (herbívora, carnívora u omnívora) y sus periodos de celo; de los minerales se desea saber si se trata de cristales o de rocas.

Además, interesa registrar qué elementos sirven de alimento a otros elementos, teniendo en cuenta que ningún mineral se considera alimento y que un vegetal no se alimenta de ningún otro elemento.

Del personal del parque se guarda el DNI, número de CUIL, nombre, dirección, teléfonos (domicilio, móvil) y sueldo. Se distinguen los siguientes tipos de personal: Personal de gestión: registra los datos de los visitantes del parque y están destinados en una entrada del parque (las entradas se identifican por un número).

Personal de vigilancia: vigila un área determinada del parque que recorre en un vehículo (tipo y matrícula). Puede ocurrir que use el mismo vehículo para mas de un área, pero siempre es el mismo en cada área que vigila.

Personal investigador: Tiene una titulación que ha de recogerse y pueden realizar (incluso conjuntamente) proyectos de investigación sobre un determinado elemento. Un proyecto de investigación tiene un presupuesto y un periodo de realización.

Personal de conservación: mantiene y conserva un área determinada del parque. Cada uno lo realiza en una especialidad determinada (limpieza, caninos).

Un visitante (DNI, nombre, domicilio y profesión) debe alojarse dentro de los alojamientos de que dispone el parque; éstos tienen una capacidad limitada y tienen una determinada categoría.

Los alojamientos organizan excursiones al parque, en vehículo o a pie, en determinados días de la semana y a una hora determinada. A estas excursiones puede acudir cualquier visitante del parque (independientemente del alojamiento en que este). Un visitante tiene, obligatoriamente, que alojarse en el parque. Una excursión puede ser organizada por más de un alojamiento.