Samantha Tsoi, 19404107
IIC2413 Instructor: Adrian Soto 26/4/2017

Entrega 2 (Par)

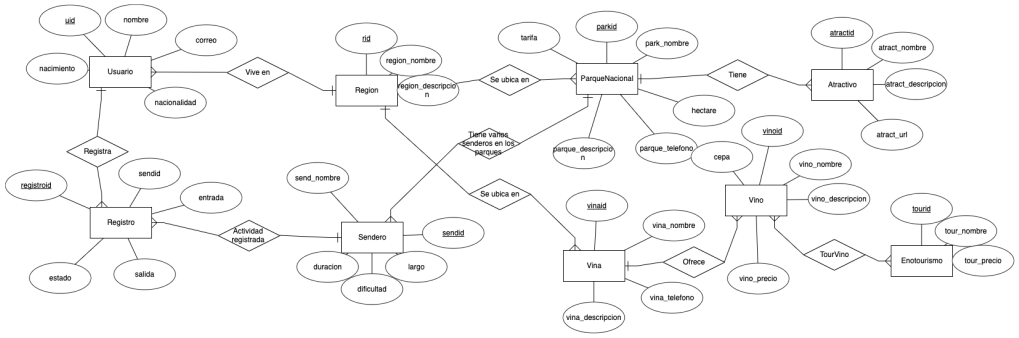
∗ **El esquema de su base de datos, incluyendo nombre de los atributos, su tipo y sus llaves.**

```
CREATE TABLE Usuario(
uid INT PRIMARY KEY not null,
user_nombre VARCHAR(50) not null,
dob DATE not null,
correo VARCHAR(100) not null,
nacionalidad VARCHAR(100) not null
);


CREATE TABLE Region(
rid INT PRIMARY KEY not null,
region_nombre VARCHAR(100) not null,
region_descripcion VARCHAR(200) not null
);

CREATE TABLE ParqueNacional(
rid INT,
parkid INT PRIMARY KEY not null,
park_nombre VARCHAR(100) not null,
hectare FLOAT not null,
parque_descripcion VARCHAR(300) not null,
tarifa FLOAT not null,
FOREIGN KEY (rid) REFERENCES Region(rid) ON DELETE SET NULL
);

CREATE TABLE Sendero(
sendid INT PRIMARY KEY not null,
parkid INT not null,
send_nombre VARCHAR(100) not null,
```

```
largo FLOAT not null,
dificultad VARCHAR(20) not null,
duracion INT not null,
FOREIGN KEY (parkid) REFERENCES ParqueNacional(parkid) ON DELETE CASCADE
);


CREATE TABLE Registro(
registroid INT PRIMARY KEY not null,
uid INT,
sendid INT,
entrada TIMESTAMP not null,
salida TIMESTAMP not null,
estado VARCHAR(10) not null,
FOREIGN KEY (uid) REFERENCES Usuario(uid) ON DELETE SET NULL,
FOREIGN KEY (sendid) REFERENCES Sendero(sendid) ON DELETE SET NULL
);

CREATE TABLE Atractivo(
atractid INT PRIMARY KEY not null,
parkid INT not null,
atract_nombre VARCHAR(100) not null,
atract_descripcion VARCHAR(300) not null,
atract_url VARCHAR(200) not null,
FOREIGN KEY (parkid) REFERENCES ParqueNacional(parkid) ON DELETE CASCADE
);

CREATE TABLE Vina(
rid INT,
vinaid INT PRIMARY KEY not null,
vina_nombre VARCHAR(100) not null,
vina_telefono BIGINT not null,
vina_descripcion VARCHAR(300) not null,
FOREIGN KEY (rid) REFERENCES Region(rid) ON DELETE SET NULL
);

CREATE TABLE Vino(
vinoid INT PRIMARY KEY not null,
vinaid INT not null,
vino_nombre VARCHAR(100) not null,
vino_descripcion VARCHAR(300) not null,
cepa VARCHAR(50) not null,
vino_precio FLOAT not null,
FOREIGN KEY (vinaid) REFERENCES Vina(vinaid) ON DELETE CASCADE
```

```
);

CREATE TABLE Enoturismo(
tourid INT PRIMARY KEY not null,
tour_nombre VARCHAR(100) not null,
tour_precio FLOAT not null
);

CREATE TABLE TourVino(
tourid INT not null,
vinoid INT not null,
PRIMARY KEY (tourid, vinoid),
FOREIGN KEY (vinoid) REFERENCES Vino(vinoid) ON DELETE CASCADE,
FOREIGN KEY (tourid) REFERENCES Enoturismo(tourid) ON DELETE CASCADE
);
```

∗ **Un diagrama E/R de su dominio explicando el signicado tras sus entidades y relaciones.**



∗ **Todas las restricciones de llaves primarias y foraneas de las tablas.**

La justicacion de por que su modelo se encuentra en BCNF utilizando dependencias funcionales.

uid PRIMARY KEY → user_nombre, nacimiento, correo, nacionalidad

rid PRIMARY KEY→ region_nombre, region_descripcion

parkid PRIMARY KEY → park_nombre, hectare, parque_telefono, parque_descripcion, tarifa, rid FOREIGN KEY REFERENCES Region(rid) ON DELETE SET NULL

sendid PRIMARY KEY → parkid FOREIGN KEY REFERENCES ParqueNacional(parkid) ON DELETE CASCADE, send_nombre, largo, dificultad, duracion

registroid PRIMARY KEY → uid FOREIGN KEY REFERENCES Usuario(uid) ON DELETE SET NULL, sendid FOREIGN KEY REFERENCES Sendero(sendid) ON DELETE SET NULL, entrada, salida, estado

atractid PRIMARY KEY → parkid FOREIGN KEY REFERENCES ParqueNacional(parkid) ON DELETE CASCADE, atract_nombre, atract_descripcion, atract_url

vinaid PRIMARY KEY → rid FOREIGN KEY REFERENCES Region(rid) ON DELETE SET NULL, vina_nombre, vina_telefono, vina_descripcion

vinoid PRIMARY KEY → vinaid FOREIGN KEY REFERENCES Vina(vinaid) ON DELETE CASCADE, vino_nombre, vino_descripcion, cepa, vino_precio

tourid PRIMARY KEY → tour_nombre, tour_precio

tourid PRIMARY KEY + FOREIGN KEY REFERENCES Enoturismo(tourid) ON DELETE CASCADE, vinoid PRIMARY KEY + FOREIGN KEY REFERENCES Vino(vinoid) ON DELETE CASCADE

In order to show that our model is in BCNF, the model also has to be in 1NF, 2NF, and 3NF. To show that our model satisfies the 1st normal form, we can see that for every table, all our attributes are single valued, entries in a column have to be of the same type, and rows are uniquely identified. To show that our model satisfies the 2nd normal form, all attributes/columns must be dependent on the primary key and only the primary key, not any non-prime key. Hence, there is no partial dependency. To show that our model satisfies the 3rd normal form, the model must have fields that can only be determined by the key in the table and not other columns. This means that there should not be a 1-to-1 relationship in other columns other than with the column with the primary key. We can see that in our tables as all the attributes of our tables are only relevant to that table. In order to prevent partial dependency between the tours, the wines, and the vinas, we had to set up a junction table of tour and vino. When given one vino, I will know what vina it connects to, as tour and vina has a many-to-many relationship, and vina and vino has a one-to-many relationship, because one vino can only be distinct to one vina but a vina can have many vinos. The functional dependencies are written above to demonstrate this.

∗ **Todas las consultas en SQL que corre su aplicacion.**

```
1. SELECT DISTINCT Usuario.uid, Usuario.user_nombre
FROM Usuario, Sendero, Registro
where Usuario.uid = Registro.uid and
```

```
Sendero.sendid = Registro.sendid and
Usuario.uid = $input;

2. SELECT DISTINCT region_nombre, vina_nombre
FROM Vina, Region
WHERE Vina.rid = Region.rid and Region.rid = 6
UNION
SELECT region_nombre, park_nombre
FROM ParqueNacional, Region
WHERE ParqueNacional.rid = Region.rid AND Region.rid = 6;

3. SELECT DISTINCT Vino.vino_nombre
FROM Enoturismo, TourVino, Vino
WHERE Enoturismo.tourid = TourVino.tourid
AND TourVino.vinoid = Vino.vinoid
AND tour_nombre ilike '$tour'
AND cepa ilike '$cepa';

4. SELECT Vina.vina_nombre,
       t1.vino_nombre,
       t1.vino_descripcion,
       t1.cepa,
       t1.vino_precio
FROM   Vino t1,
       Vina
WHERE  t1.vinaid = Vina.vinaid
       AND t1.vino_precio = (SELECT Max(t2.vino_precio)
                             FROM   Vino t2);

5. SELECT Usuario.uid, Usuario.user_nombre
    FROM Sendero, Registro, Usuario
    WHERE Registro.sendid = Sendero.sendid
    AND Usuario.uid = Registro.uid
    AND Registro.estado = 'en ruta'
    AND Sendero.largo = (SELECT Max(Sendero.largo)
        FROM Sendero);

6.
DROP VIEW IF EXISTS temp;
CREATE VIEW temp AS
SELECT Registro.sendid,
COUNT(Registro.registroid) AS ct
FROM Registro
WHERE Registro.estado = 'perdido'
```

```
GROUP BY Registro.sendid;
SELECT DISTINCT Sendero.send_nombre, Sendero.largo,
Sendero.dificultad, Sendero.duracion, ParqueNacional.park_nombre, temp.ct
FROM temp, Sendero, ParqueNacional
WHERE temp.sendid = Sendero.sendid
AND Sendero.parkid = ParqueNacional.parkid
AND ct = (SELECT MAX(ct) FROM temp);

7. SELECT ParqueNacional.park_nombre, t2.ct, t1.kilo
FROM ParqueNacional,
(SELECT COALESCE(SUM(Sendero.largo),0) AS kilo
    FROM Sendero
    WHERE Sendero.parkid = $parkid) t1,
(SELECT COUNT(Sendero.sendid) AS ct
    FROM Sendero
    WHERE Sendero.parkid = $parkid) t2
WHERE ParqueNacional.parkid = $parkid;

8. SELECT *
FROM Sendero s1
WHERE (SELECT COUNT(*)
    FROM Sendero s2
    WHERE s2.largo > s1.largo) = '$iint';
```

∗ **Todos los comandos SQL utilizados para crear sus tablas.**

```
DROP TABLE IF EXISTS Usuario CASCADE;
CREATE TABLE Usuario(
uid INT PRIMARY KEY not null,
user_nombre VARCHAR(50) not null,
dob DATE not null,
correo VARCHAR(100) not null,
nacionalidad VARCHAR(100) not null
);


DROP TABLE IF EXISTS Region CASCADE;
CREATE TABLE Region(
rid INT PRIMARY KEY not null,
region_nombre VARCHAR(100) not null,
region_descripcion VARCHAR(200) not null
);
```

```
DROP TABLE IF EXISTS ParqueNacional CASCADE;
CREATE TABLE ParqueNacional(
rid INT,
parkid INT PRIMARY KEY not null,
park_nombre VARCHAR(100) not null,
hectare FLOAT not null,
parque_descripcion VARCHAR(300) not null,
tarifa FLOAT not null,
FOREIGN KEY (rid) REFERENCES Region(rid) ON DELETE SET NULL
);

DROP TABLE IF EXISTS Sendero CASCADE;
CREATE TABLE Sendero(
sendid INT PRIMARY KEY not null,
parkid INT not null,
send_nombre VARCHAR(100) not null,
largo FLOAT not null,
dificultad VARCHAR(20) not null,
duracion INT not null,
FOREIGN KEY (parkid) REFERENCES ParqueNacional(parkid) ON DELETE CASCADE
);


DROP TABLE IF EXISTS Registro CASCADE;
CREATE TABLE Registro(
registroid INT PRIMARY KEY not null,
uid INT,
sendid INT,
entrada TIMESTAMP not null,
salida TIMESTAMP not null,
estado VARCHAR(10) not null,
FOREIGN KEY (uid) REFERENCES Usuario(uid) ON DELETE SET NULL,
FOREIGN KEY (sendid) REFERENCES Sendero(sendid) ON DELETE SET NULL
);

DROP TABLE IF EXISTS Atractivo CASCADE;
CREATE TABLE Atractivo(
atractid INT PRIMARY KEY not null,
parkid INT not null,
atract_nombre VARCHAR(100) not null,
atract_descripcion VARCHAR(300) not null,
atract_url VARCHAR(200) not null,
FOREIGN KEY (parkid) REFERENCES ParqueNacional(parkid) ON DELETE CASCADE
);
```

```
DROP TABLE IF EXISTS Vina CASCADE;
CREATE TABLE Vina(
rid INT,
vinaid INT PRIMARY KEY not null,
vina_nombre VARCHAR(100) not null,
vina_telefono BIGINT not null,
vina_descripcion VARCHAR(300) not null,
FOREIGN KEY (rid) REFERENCES Region(rid) ON DELETE SET NULL
);

DROP TABLE IF EXISTS Vino CASCADE;
CREATE TABLE Vino(
vinoid INT PRIMARY KEY not null,
vinaid INT not null,
vino_nombre VARCHAR(100) not null,
vino_descripcion VARCHAR(300) not null,
cepa VARCHAR(50) not null,
vino_precio FLOAT not null,
FOREIGN KEY (vinaid) REFERENCES Vina(vinaid) ON DELETE CASCADE
);

DROP TABLE IF EXISTS Enoturismo CASCADE;
CREATE TABLE Enoturismo(
tourid INT PRIMARY KEY not null,
tour_nombre VARCHAR(100) not null,
tour_precio FLOAT not null
);

DROP TABLE IF EXISTS TourVino CASCADE;
CREATE TABLE TourVino(
tourid INT not null,
vinoid INT not null,
PRIMARY KEY (tourid, vinoid),
FOREIGN KEY (vinoid) REFERENCES Vino(vinoid) ON DELETE CASCADE,
FOREIGN KEY (tourid) REFERENCES Enoturismo(tourid) ON DELETE CASCADE
);
```

∗ **Incluir cualquier supuesto que hayan realizado sobre la entrega, mientras sea razonable.**
Para copiar las tablas de CSV a las tablas:

```
\copy Usuario from '../Entrega2/Datos/Usuario.csv' delimiter ',' CSV HEADER ;
\copy Region from '../Entrega2/Datos/Region.csv' delimiter ',' CSV HEADER ;
```

```
\copy ParqueNacional from '../Entrega2/Datos/ParqueNacional.csv' delimiter ',' CSV HEADER
\copy Sendero from '../Entrega2/Datos/Sendero.csv' delimiter ',' CSV HEADER ;

CREATE TABLE TourTemp(
tourid INT not null,
tour_nombre VARCHAR(100) not null,
tour_precio FLOAT not null
);

\copy TourTemp from '../Entrega2/Datos/Enoturismo.csv' delimiter ',' CSV HEADER ;
INSERT INTO Enoturismo(tourid, tour_nombre, tour_precio) SELECT DISTINCT * FROM TourTemp;
DROP TABLE TourTemp;


# \copy Enoturismo from '../Entrega2/Datos/Enoturismo.csv' delimiter ',' CSV HEADER ;
\copy Atractivo from '../Entrega2/Datos/Atractivo.csv' delimiter ',' CSV HEADER ;

CREATE TABLE VinaTemp(
rid INT,
vinaid INT not null,
vina_nombre VARCHAR(100) not null,
vina_telefono BIGINT not null,
vina_descripcion VARCHAR(300) not null
);

\copy VinaTemp from '../Entrega2/Datos/Vina.csv' delimiter ',' CSV HEADER ;
INSERT INTO Vina(rid, vinaid,vina_nombre,vina_telefono, vina_descripcion) SELECT DISTINCT
DROP TABLE VinaTemp;

\copy Vino from '../Entrega2/Datos/Vino.csv' delimiter ',' CSV HEADER ;
\copy TourVino from '../Entrega2/Datos/TourVino.csv' delimiter ',' CSV HEADER ;


iconv -t utf-8 Registro.csv > Registro-utf8.csv

iconv -f ISO-8859-1 -t utf-8 Region.csv > Region-utf8.csv

iconv -f ISO-8859-1 -t utf-8 Usuario.csv > Usuario-utf8.csv

iconv -f ISO-8859-1 -t utf-8 Enoturismo.csv > Enoturismo-utf8.csv

iconv -f ISO-8859-1 -t utf-8 Sendero.csv > Sendero-utf8.csv

iconv -f ISO-8859-1 -t utf-8 ParqueNacional.csv > ParqueNacional-utf8.csv
```

```
iconv -f ISO-8859-1 -t utf-8 Vino.csv > Vino-utf8.csv

iconv -f ISO-8859-1 -t utf-8 Vina.csv > Vina-utf8.csv

iconv -f ISO-8859-1 -t utf-8 TourVino.csv > TourVino-utf8.csv

iconv -f ISO-8859-1 -t utf-8 Atractivo.csv > Atractivo-utf8.csv
```