VIRTUAL MODEL DATASET SGIJ GENERATOR

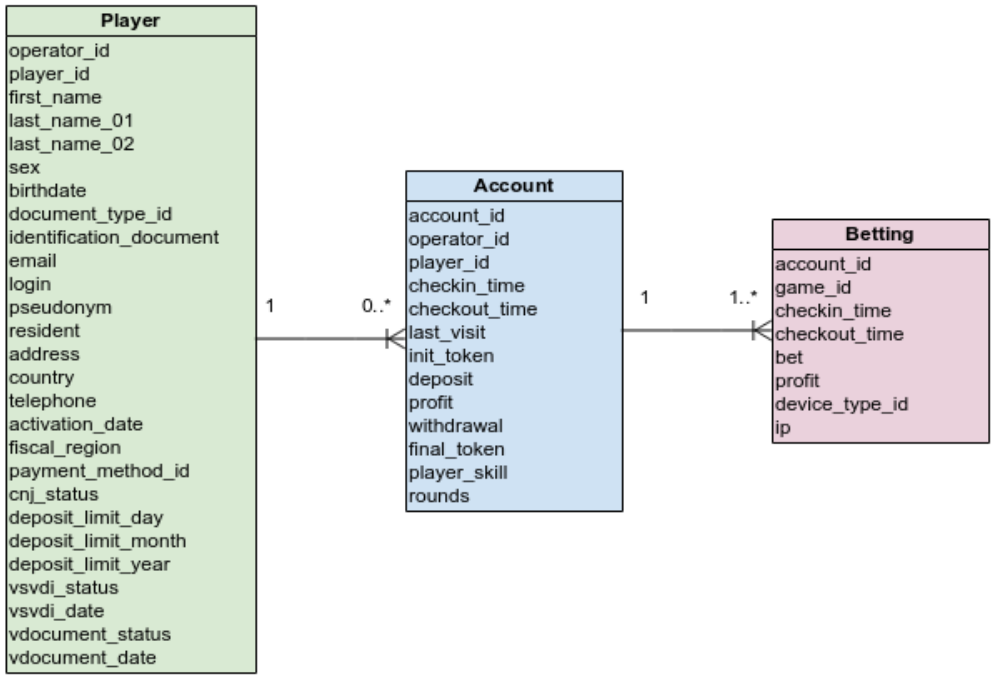
Presentación

Este documento presenta como se ha generado el Dataset para SGIJ teniendo en cuenta todas las asunciones necesarias. Se explicará el script creado a tal efecto así como las variables de entrada que se le pueden aportar para generar diferentes datasets. Igualmente se explicará los pasos a seguir para importar los mismos en una base de datos.

Modelo del Dataset

El Dataset generado por el script **'sgij_dataset.py'** esta compuesto de tres ficheros en formato CSV, fácilmente importable en cualquier base de datos. Estos ficheros reflejan el modelo por defecto creado en la base de datos que contendrá el contenido generado por el algoritmo y que representa un Dataset Virtual o ficticio capaz de ser utilizado a la hora de desarrollar un estudio de DataScience.

En la siguiente figura se puede ver este modelo así como las entidades que lo conforman y sus relaciones.



En este modelo relacional podemos distinguir tres entidades:

- **Player**: esta entidad representa el **Registro de usuario**, es decir los datos de indentificación de los participantes, los relativos a los límites de depósitos y otros datos de configuración tales como el estado del participante en la plataforma de juego.
- **Account**: esta entidad representa la **Cuenta de juego,** es decir, incluye para cada periodo y participante, los totales correspondientes al saldo inicial, las transacciones realizadas por depósitos y retiradas, saldo inicial y final en el periodo y total de jugadas en el mismo
- **Betting**: sta entidad representa las **Partidas de juego**, es decir, todos los registros o eventos durante un periodo de cuenta en donde se ve detalladamente en que juegos a participado el jugador así como las partidas ganadas y perdidas durante este periodo de cuenta de juego

Voy a explicar brevemente cada atributo de cada entidad involucrada en el modelo:

Entidad	**Atributo**	**Tipo**	**Descripción**
Player	operator_id	Integer	Identificador del operador
Player	player_id	Integer	Identificador del jugador. La clave operator_id junto a player_id es clave primaria de la tabla Player
Player	first_name	Varchar	Nombre de pila del jugador
Player	last_name_01	Varchar	Primer apellido del jugador
Player	last_name_02	Varchar	Segundo apellido del jugador
Player	sex	Char	Sexo del jugador
Player	birthdate	Date (YYYY-MM-DD)	Fecha nacimiento del jugador
Player	document_type_id	Varchar	Tipo documento identifación jugador. Validado por documento DGOJ_Monitorizacion_2.13.xsd
Player	identification_document	Varchar	Documento identificador jugador
Player	email	Varchar	Email del jugador
Player	login	Varchar	Login del jugador
Player	pseudonym	Varchar	Pseudónimo del jugador

Player	resident	Char. Posibles valores: S, N	Si es residente o no el jugador en España
Player	address	Varchar	Dirección del jugador
Player	country	Varchar	País donde vive el jugador. Coincide siempre con su residencia.
Player	telephone	Varchar	Teéfono del jugador
Player	activation_date	Date (YYYY-MM-DD)	Fecha de activación del jugador por parte del operador.
Player	fiscal_region	Varchar	Region fiscal del jugador.
Player	payment_method_id	Varchar	Método de pago registrado por el juagador.
Player	cnj_stat	Varchar	Estdo según codificación CNJ
Player	deposit_limit_day	Integer	Límite de depósito diario del jugador
Player	deposit_limit_month	Integer	Límite de depósito mensual del jugador
Player	deposit_limit_year	Integer	Límite de depósito anual del jugador
Player	vsvdi_status	Varchar	Estado según codificación VSVDI
Player	vsvdi_date	Date (YYYY-MM-DD)	Ultima fecha actualización estado VSDI
Player	vdocument_status	Varchar	Estado del documento jugador
Player	vdocument_date	Date (YYYY-MM-DD)	Ultima fecha actualización estado documento jugador
Account	account_id	Integer	Identificador único evento cuenta de jugador agrupado por periodo
Account	operator_id	Integer	Identificador único operador
Account	player_id	Integer	Identificador único jugador dentro de la plataforma del operador
Account	checkin_time	Datetime	Fecha inicio del periodo de

		(YYYY-mm-DD HH:MM:SS	movimiento de cuenta de jugador
Account	checkout_time	Datetime (YYYY-mm-DD HH:MM:SS)	Fecha fin del periodo de movimiento de cuenta de jugador
Account	last_visit	Datetime (YYYY-mm-DD HH:MM:SS)	Ultima visita jugador
Account	init_toke	Integer	Token en cuenta al inicio del periodo
Account	profit	Integer	Beneficio durante el periodo
Account	deposit	Integer	Depósitos realizados por el jugador en el periodo
Account	withdrawal	Integer	Retiradas realizadas por el jugador en el periodo
Account	final_token	Integer	Tokens en cuenta al final del periodo
Account	player_skill	Float	Certidumbre del jugador durante el periodo
Account	Rounds	Integer	Numero de jugadas realizadas por el jugador durante el periodo
Betting	account_id	Integer	Identificador único evento cuenta de jugador agrupado por periodo
Betting	game_id	Integer	Tipo de juego en donde el jugador ha realizado su aportación
Betting	checkin_time	Datetime (YYYY-mm-DD HH:MM:SS)	Fecha inicio del evento de juagada en el tipo de juego seleccionado
Betting	checkout_time	Datetime (YYYY-mm-DD HH:MM:SS)	Fecha fin del evento de jugada en el tipo de juego seleccionado
Betting	bet	Integer	Apuesta realizada por el jugador en el tipo de juego seleccionado
Betting	Profit	Integer	Beneficio realizado por el jugador durante la apuesta
Betting	device_type_id	Varchar	Tipo de dispositivo utilizado para realizar la apuesta
Betting	ip	Varchar	IP desde donde se ha realizado la apuesta

Asunciones del dataset

- **Birhdate:** desde 1 Enero 1970 hasta 1 Enero 2000
- **document_type_id:** Validado por documento DGOJ_Monitorizacion_2.13.xsd
- **country:** Validado por documento DGOJ_Monitorizacion_2.13.xsd
- **device_type_id:** Validado por documento DGOJ_Monitorizacion_2.13.xsd
- **activation_date**: desde 1 Enero 2010 hasta actualidad
- **cnj_status**: Validado por documento DGOJ_Monitorizacion_2.13.xsd
- **Fórmula tokens** → final_token = initial_token + deposit + profit + withdrawal
- **Fórmula bets** → profit en account = SUM(profits en betting) para cada jugador operario
- **Periodo de juegos**: Periodo de cada registro en la tabla de **‘account’** representa el máximo intervalo en donde todos los subperiodos de la tabla Betting se encuentran para todos las juagas representadas en la tabla de **‘betting’**

Modelo datos en MySQL

Antes de poder importar nuestro dataset generado debemos de crear el modelo de datos o tablas en un schema creado al efecto. El schema creado en MySQL se ha llamado ‘gaming’ y el usuario de MySQL utilizado para crear el modelo debe de tener los privilegios necesarios para ello.

A la hora de crear el schema escoger como Default Collation: **‘utf8-utf8_spanish_ci’**, pues algunos jugadores genetados pueden ser extranjeros y el juego de caracteres de la base de datos debe de soportarlos.

Una vez conectado a la base de datos y creado el schema antes mencionado podemos ejecutar los scrips de creación como sigue:

Creación de la tabla **player**

```
CREATE TABLE IF NOT EXISTS player (
    operator_id INT(11),
    player_id INT(11),
    first_name VARCHAR(65),
    last_name_01 VARCHAR(65),
```

```
    last_name_02 VARCHAR(65),
    sex CHAR(1),
    birthdate DATE,
    document_type_id VARCHAR(10),
        identification_document VARCHAR(65),
    email VARCHAR(65),
    login VARCHAR(65),
    pseudonym VARCHAR(65),
    resident CHAR(1),
    address VARCHAR(255),
    country VARCHAR(65),
    telephone VARCHAR(65),
    activation_date DATE,
    fiscal_region VARCHAR(65),
    payment_method_id VARCHAR(10),
    cnj_status VARCHAR(10),
    deposit_limit_day INT(11),
    deposit_limit_month INT(11),
    deposit_limit_year INT(11),
    vsvdi_status VARCHAR(10),
    vsvdi_date DATE,
    vdocumental_status VARCHAR(10),
    vdocumental_date DATE,
    PRIMARY KEY (operator_id, player_id)
)  ENGINE=INNODB;
```

Creación de la tabla **account**

```
CREATE TABLE IF NOT EXISTS account (
    account_id INT(11),
    operator_id INT(11),
    player_id INT(11),
    checkin_time DATETIME,
```

```
    checkout_time DATETIME,
    last_visit DATETIME,
    init_token INT(11),
    deposit INT(11),
    profit INT(11),
    withdrawal INT(11),
    final_token INT(11),
    player_skill FLOAT,
    rounds INT(11),
    PRIMARY KEY (account_id)
)  ENGINE=INNODB;
```

Creación de la tabla **betting**

```
CREATE TABLE IF NOT EXISTS betting (
    account_id INT(11),
    game_id VARCHAR(10),
    checkin_time DATETIME,
    checkout_time DATETIME,
    bet INT(11),
    profit INT(11),
        device_type_id VARCHAR(10),
    ip VARCHAR(125)
)  ENGINE=INNODB;
```

En cualquier caso el script **import_dataset_mysql.sql** contiene todos los comandos comentados

Importar Dataset en MySQL

Por defecto MySQL importa los ficheros CSV desde la carpeta '/var/lib/mysql-files' de Linux. Para acceder a la misma y copiar los ficheros CSV que representan nuestro Dataset Virtual deberemos de hacerlo como superusuario.

```
miguel@host: sudo su
miguel@host: cd /var/lib/mysql-files
```

miguel@host: cp /home/miguel/git/training-vertica-python/csv/*.* ./

En la carpeta de importación deberemos de tener tres ficheros:

- account_register.csv
- betting_register.csv
- player_register.csv

Una vez copiados los ficheros solamente deberemos de ejecutar los comandos de importación SQL de MySQL desde cualquier cliente SQL conectado a nuestra base de datos y ejecutar estos comandos:

```
# import player data
LOAD DATA INFILE '/var/lib/mysql-files/player_register.csv'
INTO TABLE player
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

# import account data
LOAD DATA INFILE '/var/lib/mysql-files/account_register.csv'
INTO TABLE account
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

# import betting data
LOAD DATA INFILE '/var/lib/mysql-files/betting_register.csv'
INTO TABLE betting
FIELDS TERMINATED BY ','
ENCLOSED BY ""
```

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;

Ejecutar script

Para poder generar los Virtual Dataset debemos primero tener instalado y configurado correctamente en nuestro equipo python3 así como los siguientes módulos de python3:

- **Numpy**: pip3 install numpy --user
- **Faker**: pip3 install Faker --user
- **PyCountry**: pip3 install pycountry –user

Para ver la ayuda del script

python3 sgij_dataset.py –help

Para ejecutar con valores por defecto, es decir, un solo operador con 2 jugadores y 2 movimientos de cuenta para cada uno de ellos, ejecutamos lo siguiente:

python3 sgij_dataset.py

Para pasar por ejemplo 200, 500 y 300 jugadores repartidos en 3 operadores con 3 movimientos de cuenta para cada uno de ellos, ejecutamos lo siguiente:

python3 sgij_dataset.py -p 200 500 300 -r 3