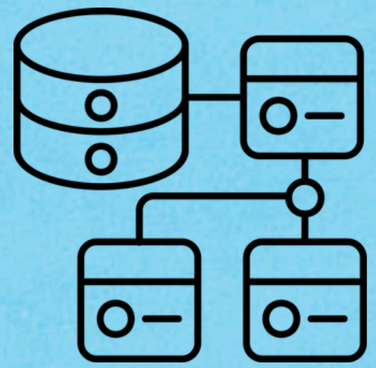
IIC2413

AYUDANTÍA 6

OBJETIVOS

- Identificar tipos de errores en datos
- Definir restricciones de Integridad (IC): qué son y cuándo se aplican en SQL
- Uso de PHP: herramientas para limpiar datos antes de cargar





LIMPIEZA DE DATOS

RESTRICCIONES DE INTEGRIDAD (IC)

¿QUÉ ES UNA IC?

Son reglas que garantizan la **coherencia, calidad** y **validez** de los datos en una BDD.



Se definen en dos tipos:

Estructurales:
PK, FK, UNIQUE, NOT NULL, CHECK – se definen en el esquema SQL

De negocio:
formato de correos, RUN válido, teléfono de 9 dígitos, fechas – se verifican en PHP

ERRORES TÍPICOS

Datos no estandarizados:
El mismo valor escrito de formas distintas. Ej: "chile", "Chile", "CHILE" en columna País

Tipo de dato incorrecto:
Tipo de dato incorrecto Se espera un tipo y llega otro. Ej: columna Edad recibe "veinte" en vez de 20

Datos nulos indebidos:

Falta algún campo obligatorio. Por ejemplo: nombre = “” o RUN = NULL.

Datos duplicados:
Misma entidad registrada más de una vez. Ej: dos filas con el mismo correo en tabla Usuarios

Incoherencia entre datos:
Dos campos que se contradicen entre sí. Ej: fecha de nacimiento 2010 pero edad registrada 45

Fuera de formato:
El valor existe pero no respeta el formato. Ej: correo "juan@@gmail.com", teléfono "1234"

¿QUÉ HACER?

Siempre se debe buscar la pérdida mínima de información.

1

Corregir

Si se tiene información suficiente para inferir el valor correcto

Ej: juan@gmail..com → juan@gmail.com

2

Asignar “nulo”/NULL

Cuando no se puede arreglar, pero se puede dejar un dato genérico para respetar restricciones del esquema.

Ej: Teléfono 1234 → 9 0000 0000

3

Eliminar

ÚLTIMO RECURSO. Solo se hace si el dato es irreparable y sin él la tupla pierde todo sentido.

Ej: Atención médica donde el paciente no existe en ninguna parte de la BD.

Pero... ¿cómo podemos corregir los datos?

LIMPIEZA DE DATOS

USANDO PHP

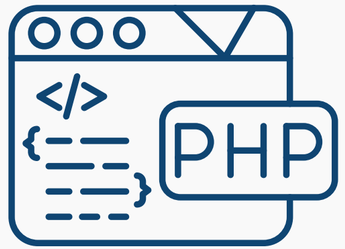
CONOZCAMOS

php

¿Qué es? Es un lenguaje de programación orientado al desarrollo web del lado del servidor. Permite crear sitios dinámicos, procesar formularios, conectarse con bases de datos y automatizar tareas con datos.

¿PARA QUÉ NOS SIRVE?

- Validar datos antes de su carga en SQL.
- Limpiar errores de formato en registros provenientes de archivos.
- Corregir tuplas con espacios extra y campos inconsistentes.
- Estandarizar textos, fechas y valores repetidos.
- Detectar registros incompletos antes de insertarlos.
- Generar archivos limpios.
- Complementar las restricciones de integridad definidas en SQL.



COMANDOS BÁSICOS

- Para ejecutar el archivo desde la terminal se usa > **php nombrearchivo.php**
- Todo archivo php empieza con **<?php**
- Las variables se definen con **$** ($nombre, $edad, etc.)
- Cada línea termina con **;**
- Para imprimir se usa **echo** (equivalente de print)

```php
<?php
$nombre = "Juan";
$edad = 25;
$frutas = ["manzana", "uva"];

echo $nombre; //Imprime Juan
echo "Tiene ", $edad, " años"; //Imprime Tiene 25 años
echo "Le gusta la ", $frutas[0], " y la ", $frutas[1] //Imprime Le gusta la manzana y la uva
?>
```

MANEJO DE STRINGS

Algunos comandos que les pueden servir son:

- **trim($s):** Borra espacios en blanco al inicio y al final
- **strtolower/strtoupper:** Hace que el texto esté en minúsculas o mayúsculas.
- **str_replace($a, $b, $s):** Reemplaza a por b en las partes que aparece en s.
- **strpos($b, $a):** Entrega la posición de la primera ocurrencia de a en b (false si no)
- **strlen($s):** Largo del string
- **explode($sep, $s):** Separa s de acuerdo al separador (sep).
- **implode($sep, $arr):** Junta los elementos de arr en un string con sep como separador.

```php
<?php
$texto = "  Juan PEREZ  ";

// Limpiar espacios al inicio y al final
$limpio = trim($texto); //"Juan PEREZ"

// Mayúsculas/minúsculas
$mayus = strtoupper($limpio); //"JUAN PEREZ"
$minus = strtolower($limpio); //"juan perez"

// Largo del string
$largo = strlen($limpio); //10

// Reemplazar texto
$nuevo = str_replace("PEREZ", "Perez", $limpio); //"Juan Perez"

// Buscar un caracter o substring (posición)
$pos = strpos("juan@mail.com", "@"); //4

// Dividir un string en una lista
$partes = explode(",", "a,b,c"); //["a", "b", "c"]

// Juntar el array en un string
$juntos = implode(", ", $partes); //"a, b, c"
?>
```

CONDICIONALES Y LOOPS

Operadores: ==, !=, <, >, <=, >=, && (and), || (or), ! (not)

```php
<?php
// If / else
if ($nota >= 3.95) {
    echo "Azul";
} elseif ($nota >= 3.5 && $nota <= 3.94) {
    echo "Morado";
}
else {
    echo "Rojo";
}
```

```php
// While
$i = 0;
while ($i < 5) {
    echo $i;
    $i++;
}

// For
for ($i = 0; $i < 5; $i++) {
    echo $i;
}
?>
```

ARRAYS

¡Muy importante!
Cada fila de los archivos .csv que reciban van a ser un array.

*Nota: Hay **muchos** otros comandos, estos son los más básicos. Se recomienda fuertemente leer la **documentación de php,** donde se incluyen ejemplos (https://www.php.net/).

```php
<?php
// Crear una lista
$frutas = ["manzana", "pera", "frutilla"];

// Acceder a un elemento (empieza en 0)
echo $frutas[0]; //"manzana"

// Modificar
$frutas[1] = "naranja";

// Agregar al final
$frutas[] = "uva";
array_push($frutas, "kiwi");

// Largo de la lista
$n = count($frutas); //5

// Recorrer con foreach
foreach ($frutas as $fruta) {
    echo $fruta . "\n";
};

// Saber si hay un elemento en el array
echo in_array("frutilla", $frutas); //Imprime 1
?>
```

MANEJO DE ARCHIVOS

- Para abrir: fopen(“ruta_archivo”, “r”)
- Para escribir: fopen(“ruta_archivo”, “w”)
- Para recibir la fila como array: fgetcsv(archivo)
- Para cerrar: fclose(archivo)

```php
<?php
// Abrir archivo en modo lectura ("r")
$archivo = fopen("Persona.csv", "r");

// Leer linea por linea
while (($fila = fgetcsv($archivo, 0, ';', '"', '\\')) !== false) {
    // $fila de un csv: ["1";"12345678-9";"Juan";"Perez";...]
    $rut    = $fila[1];
    $nombre = $fila[2];
    echo $nombre . " tiene rut " . $rut . "\n";
}

// Siempre cerrar el archivo al final
fclose($archivo);
?>
```

```php
<?php
// Abrir en modo escritura (sobrescribe lo que haya)
$salida = fopen("PersonaOK.csv", "w");
$columnas = ["ID", "RUN", "Nombre", "Apellido"];
fputcsv($salida, $columnas);

// Escribir un array como linea CSV (agrega la coma y el salto de linea)
$fila = ["1", "12345678-9", "Juan", "Perez"];
fputcsv($salida, $fila);

// Tambien podemos usar fwrite para escribir texto directo
fwrite($salida, "Esto es una linea de texto\n");

fclose($salida);

// Modo "a" => agregar sin borrar el contenido previo
$log = fopen("PersonaLOG.txt", "a");
fwrite($log, "Se corrigio el correo de Juan\n");
fclose($log);
?>
```

FUNCIONES

Las funciones en php se definen como **function** nombre_funcion($variables) { }

Como van a estar limpiando un gran volumen de datos, las funciones se les pueden hacer muy útiles para hacer la limpieza más rápida y reutilizar las funciones en distintos archivos (por ejemplo, todos los RUN deben tener cierto formato, así que pueden hacer una función que limpie eso y ocuparla siempre).

```php
<?php
1 reference
function limpiar_nombre($nombre) {
    // Quitamos espacios y dejamos solo la primera en mayúscula
    $nombre = trim($nombre);
    $nombre = strtolower($nombre);
    // Separador opcional, por ejemplo si trabajamos con apellido
    $nombre = ucwords($nombre, " ");
    return $nombre;
}

// Usar la funcion
$resultado = limpiar_nombre("  juan perez "); //"Juan Perez"

// Funcion con valor por defecto
2 references
function formatear($texto, $mayuscula = true) {
    if ($mayuscula) {
        return strtoupper(trim($texto));
    }
    return trim($texto);
}

echo formatear("  hola  ");//"HOLA"
echo formatear("  hola  ", false);//"hola"
?>
```

REPASO COMANDOS DE PHP

Función	Qué hace	Ejemplo
trim($s)	Quita espacios al inicio y final	" Ana " → "Ana"
strtolower($s)	Todo minúscula (solo ASCII)	"ANA" → "ana"
strtoupper($s)	Todo mayúscula (solo ASCII)	"ana" → "ANA"
str_replace($a,$b,$s)	Reemplaza $a por $b	"..com" → ".com"
strtr($s, $mapa)	Reemplaza múltiples a la vez	quitar tildes
explode($sep,$s)	Separa en array	"a@b.com" → ["a","b.com"]
implode($sep,$arr)	Une array en string	["9","12345678"] → "912345678"
strlen($s)	Largo en bytes	strlen("café") → 5
iconv_strlen($s,'UTF-8')	Largo real en caracteres	iconv_strlen("café",'UTF-8') → 4
iconv_substr($s,0,$n,'UTF-8')	Subcadena en UTF-8	no corta caracteres acentuados
preg_match($r,$s)	Valida con regex	formato RUN, teléfono
preg_replace($r,$rep,$s)	Limpia con regex	eliminar caracteres invisibles
filter_var($s, $flag)	Valida tipos conocidos	FILTER_VALIDATE_EMAIL

PRO TIP: ¡Investiguen sobre regex!

EJEMPLOS TÍPICOS:
CORREO ELECTRÓNICO

¿Qué pasa si tenemos un mail con mayúsculas?
¿Qué pasa si tenemos un mail con dos @?
Estos (y más) tipos de errores de formato del correo electrónico se pueden solucionar con php.

A modo de ejemplo se deja una función que puede limpiar esos errores (se puede hacer de muchas otras formas).

```php
function validar_correo($str) {
    // No puede estar vacio ni faltar el @
    if ($str == "" || strpos($str, "@") === false) {
        return false;
    }

    // Dividimos en parte local y dominio
    $partes = explode("@", $str);
    // Ejemplo: Hay mas de un @ (ej: "a@@b.com")
    if (count($partes) != 2) {
        if (in_array("", $partes)) {
            $partes = array_diff($partes, [""]);
            $partes = implode("@", $partes);
            return $partes;
        }
    }

    $local = trim($partes[0]);
    $dominio = trim($partes[1]);

    return strtolower($local . "@" . $dominio);
}

validar_correo("Juan@gMail.com"); //"juan@mail.com"
validar_correo("perez@@gmail.com"); //"perez@mail.com"
?>
```

FLUJO EN PHP

1. Se **lee el CSV,** se aplican las **funciones de limpieza,** y según el resultado **se escribe en el archivo OK o en el ERR.**
2. El **archivo LOG registra qué error tuvo cada tupla rechazada,** lo que permite revisar y corregir en la siguiente iteración.
3. Se vuelve a iterar hasta **minimimizar los errores,** siempre **revisando datos OK** que **podrían seguir con errores agregando nuevas funciones de limpieza.**

```php
<?php
$entrada = fopen("Persona.csv",    "r");
$ok = fopen("PersonaOK.csv",  "w");
$err = fopen("PersonaERR.csv", "w");

// Saltar encabezado
$header = fgetcsv($entrada, 0, ";");
fputcsv($ok,  $header, ";");
fputcsv($err, $header, ";");

while (($fila = fgetcsv($entrada, 0, ";")) !== false) {
    [$id, $nombre, $correo, $telefono] = $fila;

    $errores = [];

    $nombre  = limpiar_nombre($nombre);       // trim + ucwords
    $correo  = validar_correo($correo);       // función del ejemplo anterior
    $digitos = preg_replace('/[^0-9]/', '', $telefono);

    if (empty($nombre))
        $errores[] = "nombre vacío";
    if ($correo === false)
        $errores[] = "correo inválido";
    if (strlen($digitos) !== 9)  $errores[] = "teléfono inválido";

    if (count($errores) === 0) {
        fputcsv($ok, [$id, $nombre, $correo, $digitos], ";");
    } else {
        fputcsv($err, $fila, ";");
        fwrite($log, "Fila $id: " . implode(", ", $errores) . "\n");
    }
}
fclose($entrada); fclose($ok); fclose($err); fclose($log);
?>
```

LIMPIEZA DE DATOS

USANDO SQL

EJEMPLOS TÍPICOS:
PRIMARY KEY & NOT NULL

- **PRIMARY KEY** identifica de forma única cada tupla de una tabla. No puede repetirse ni ser NULL.
- **NOT NULL** garantiza que un campo siempre tenga un valor no vacío. Se usa en atributos obligatorios.
- Si PHP no filtra una tupla con un valor vacío o con un ID duplicado, **SQL la rechaza.**

```sql
CREATE TABLE Persona (
    id INTEGER PRIMARY KEY,
    -- único por tabla, no acepta NULL

    run VARCHAR(10) NOT NULL,
    -- campo obligatorio, no puede quedar vacío

    nombre VARCHAR(30) NOT NULL,
    apellido VARCHAR(30) NOT NULL,

    telefono VARCHAR(9) NOT NULL
);
```

EJEMPLOS TÍPICOS: UNIQUE & CHECK

- **UNIQUE** garantiza que un **valor no se repita en la columna.** Es útil para atributos que deben ser únicos pero no son la llave primaria, como el correo electrónico.
- **CHECK** valida que el valor cumpla una **condición lógica.** Permite restringir qué valores son válidos en un campo, como el tipo de beneficiario o el formato del teléfono.

```sql
CREATE TABLE Persona (
    id INTEGER PRIMARY KEY,

    correo TEXT UNIQUE NOT NULL,
    -- no se repite

    tipo VARCHAR(15)
        CHECK (tipo IN ('titular', 'beneficiario')),
    -- solo acepta esos dos valores

    telefono CHAR(9)
        CHECK (telefono ~ '^[0-9]{9}$')
        -- regex en sql
);
```

EJEMPLOS TÍPICOS: FOREIGN KEY

- **FOREIGN KEY** relaciona llaves entre tablas siempre que existan en ambas. Si se intenta insertar una atención cuyo paciente no está en Persona, SQL rechaza la operación.
- Conecta tuplas que **PHP no puede manejar***, ya que no sabe directamente si un atributo existe en otra tabla.
- **Siempre hay que cargar primero la tabla referenciada.**

```sql
CREATE TABLE Atencion (
    id INTEGER PRIMARY KEY,
    run_paciente VARCHAR(10) NOT NULL,
    run_medico VARCHAR(10) NOT NULL,
    diagnostico TEXT,

    -- El paciente debe existir en Persona
    FOREIGN KEY (run_paciente) REFERENCES Persona(run),

    -- El médico también debe existir en Persona
    FOREIGN KEY (run_medico) REFERENCES Persona(run)
);
```

* En realidad sí puede, pero mantener la integridad haciendo búsquedas secuenciales dentro de ciclos (for/foreach), puede demorar en el orden de $O(n^2)$.

CASOS DE USO: ¿PHP O SQL?

Error	Dónde corregir	Ejemplo
Texto con tildes o mayúsculas	PHP	"REGIÓN" → "Region"
Correo mal formado	PHP	"a@@mail.com" → descartar
Teléfono con guiones o espacios	PHP	"9-123-4567" → "912345678"
Espacios sobrantes en un campo	PHP	" Juan " → "Juan"
Typos	PHP	"Salú Ltda" → "Salud Ltda"
Dos registros con el mismo correo	SQL	UNIQUE
Campo obligatorio sin valor	SQL	NOT NULL
Valor fuera del dominio permitido	SQL	CHECK (tipo IN (...))
Referencia a una fila que no existe	SQL	FOREIGN KEY

EJEMPLO DE CARGA

- psql -f DUMP2.sql > Crea el esquema de la Base de Datos

```
-- Carga masiva desde el CSV limpio a una tabla temporal
\copy temp_persona FROM 'datos_ok/PersonaOK.csv' WITH (FORMAT CSV, DELIMITER ';', HEADER);

-- Inserción
INSERT INTO "Persona" ("ID", "RUN", "Nombres", "Apellidos", "InstSalud")
SELECT DISTINCT ON (RUN) -- SQL elimina duplicados que PHP no detectó y se deben revisar
    ID, RUN, Nombres, Apellidos, codigo_instsalud
FROM temp_persona
ORDER BY RUN, ID;
```

Ajustar PHP:

Cuando el dato está "sucio" (formato).

TablaLOG.txt



Ajustar SQL:

Identificar qué tuplas no se insertan en el esquema y por qué.

CargaLOG.txt



EJEMPLOS DE LOG

```
ITERACIÓN 4:

Tupla con error:
2886;185134-9;Javier;Díaz Contreras;Av. Manuel Montt
7747;javier.díaz@dominiofake.net;984774286;titular;185134-9;administrativo;;;;Salú  Ltda.;

Problema identificado:
    El nombre de institución "Salú  Ltda." (sin 'd' al final de Salú) no coincide con "Salud Ltda."
    en el catálogo de instituciones. La estandarización de espacios y limpiar_texto() no corrige
    este error específico donde falta una letra.

Cambios:
    En persona.php: agregar array de estandarización $estandarizaciones_inst con mapeo
    'salú ltda.' => 'salud ltda.' para corregir este error tipográfico conocido.
    Aplicar estandarización después de normalizar espacios y antes de buscar en el índice.

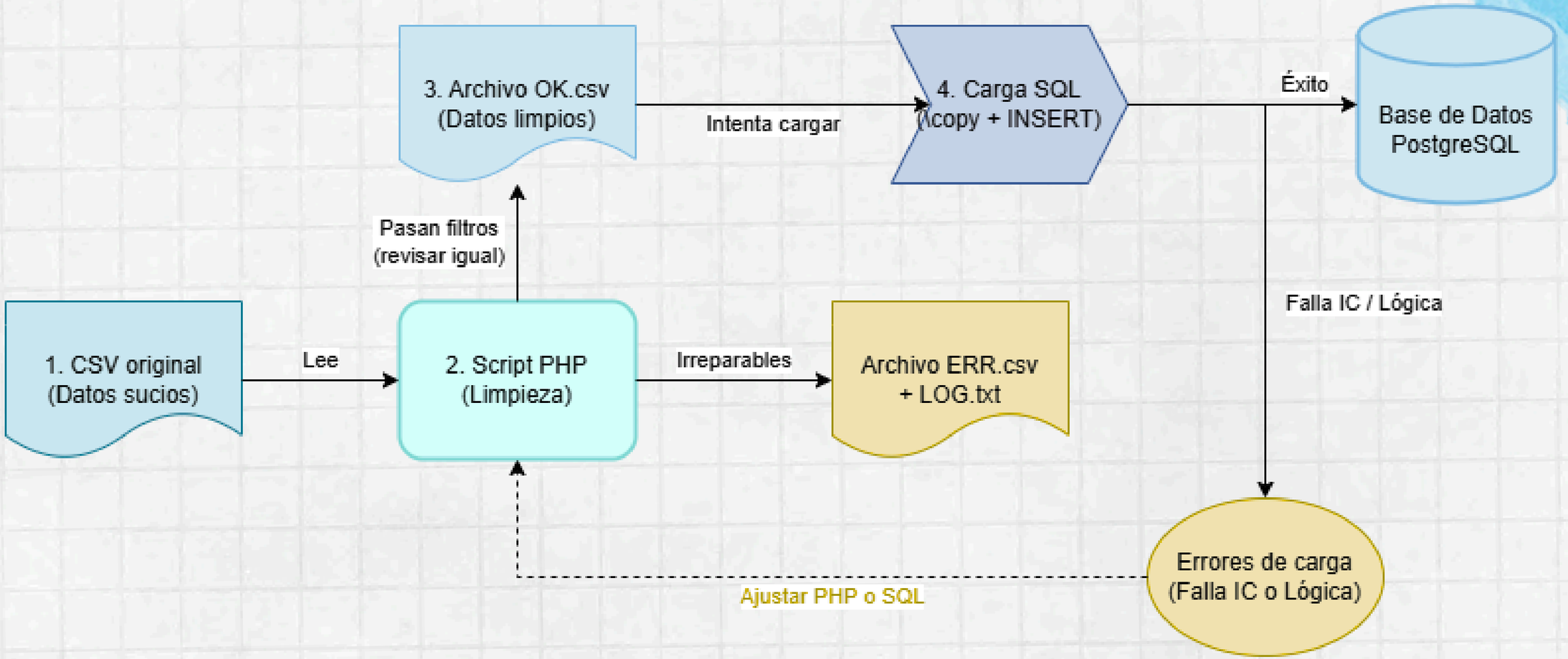
ITERACIÓN 5:
Ninguna tupla eliminada
Todas OK
```

PersonaLOG.txt

```
INSERT 0 5256
DROP TABLE
CREATE TABLE
COPY 3001
INSERT 0 2997
INSERT 0 1446
INSERT 0 454
INSERT 0 2727
DROP TABLE
CREATE TABLE
COPY 4878
INSERT 0 4878
DROP TABLE
CREATE TABLE
COPY 3937
INSERT 0 3937
DROP TABLE
CREATE TABLE
COPY 19487
INSERT 0 19487
COPY 0
DROP TABLE
CREATE TABLE
COPY 25
UPDATE 25
COPY 25
UPDATE 25
COPY 25
UPDATE 25
COPY 25
UPDATE 25
COPY 25
UPDATE 25
COPY 25
UPDATE 25
COPY 20
UPDATE 20
COPY 25
UPDATE 25
INSERT 0 158
DROP TABLE
       tabla         | tuplas
---------------------+--------
 InstituciondeSalud  |     11
 Persona             |   2997
 TipoRol             |      3
 Rol                 |   1446
 profesion           |    454
 beneficiario        |   2727
 Grupo               |     25
 Arancel             |   2662
 Farmacia            |   5256
 Atencion            |   4878
 medicamentos        |   3937
 Orden               |  19487
 Planes              |    158
(13 filas)
```

CargaLOG.txt

FLUJO DE LIMPIEZA



¡GRACIAS!