IIC2413

AYUDANTÍA

10

Apoyo E3

¿QUÉ VEREMOS?

HOY...

Contexto

Objetivos E3

Reglas de Negocio

Formato Datos

Administrativo

- LEA LAS INSTRUCCIONES DEL ENUNCIADO GENERAL
- **Fecha de inicio:** viernes 17 de octubre
- **Fecha de entrega:** lunes 3 de noviembre de 2025, 23:59.
- **Fecha y hora máxima de entrega atrasada:** jueves 6 de noviembre de 2025, 23:59.
- **Consultas solo por ISSUES** se responden en forma consolidada los jueves y sábado recepción de issues hasta miércoles 29 de octubre
- Revise las DISCUSSIONS de Github antes de preguntar. Solo se responderán las consultas originales.
- **Entregables:** Informe en formato "Markdown" (README.md) según la plantilla entregada. Programa PHP de data cleansing, archivo sql de creación y carga de BDD.
- **Lugar y forma de entrega:** En su cuenta personal <usuario >.e_3 directorio E3 del servidor stonebraker.ing.uc.cl.
- Seguridad: SIEMPRE guarde un respaldo del trabajo (por ejemplo en Onedrive). No se otorgará tiempo extra por problemas o indisponibilidad de los computadores personales. Recuerde que trabajar en el computador personal es OPCIONAL, se puede hacer todo el trabajo en el server.
- **Corrección del trabajo:** Los programas PHP y SQL que se usará para su entrega serán 'main.php' y 'carga.sql' que serán ejecutadas en forma secuencial mediante la línea de comandos de linux de su cuenta en el servidor.

DCColita de rana

1. Construir la base de datos del centro medico a partir de datos exportados de la base de datos real del centro medico

2. Recibiran una seria de archivos similares a los de la E1, para de esta manera construir la vase de datos desde cero

3. Estos datos estaran almacenados con criterios diferentes a los que se requieren en este nuevo sistema (otras IC), por lo que pueden tener errores de formato, tipo, duplicaciones, datos faltentes o incoherentes.

4. Su mision es tomar el input de los .csv y mediante el uso de PHP y sql realizar un proceso de limpieza y carga de los datos en la bdd con buscando el minimo de perdida de informacion

OBJETIVO

Realizar un proceso de limpieza de datos usando PHP previo a la carga de los datos en la base de datos la cual cumple con sus propias restricciones

Priorizando!:

1. Correción de datos
2. Asignación de nulo
3. Eliminación

Siempre tenemos que buscar la pérdida mínima de información a la hora de resolver las restricciones de integridad (IC)

ERRORES EN LOS DATOS (IC)

01

No estandarizados

02

Tipos de dato incorrecto

03

Datos nulos (cuando no se permite)

ERRORES EN LOS DATOS (IC)

04

Datos duplicados

05

Incoherencia entre datos

06

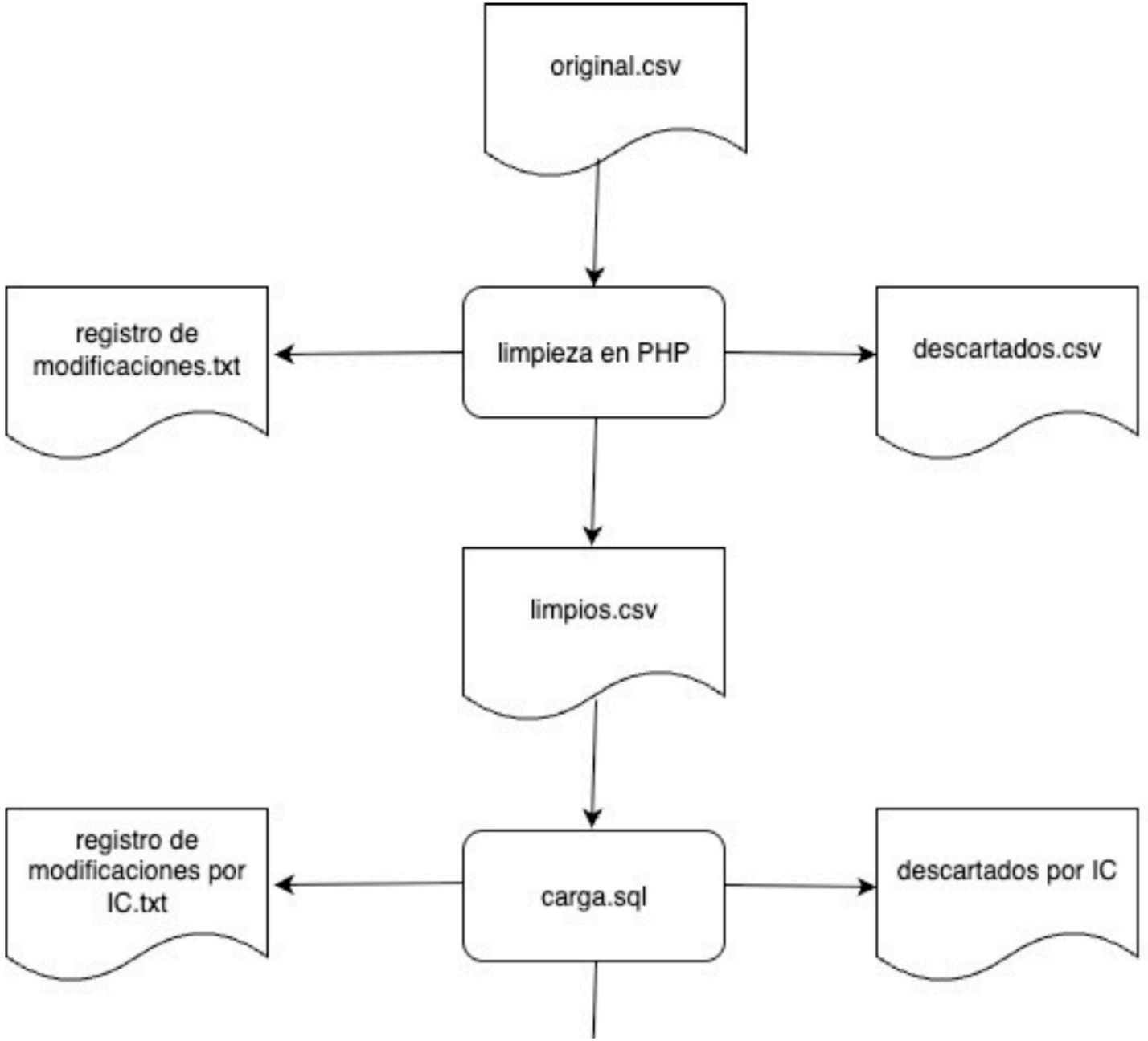
Datos fuera del formato permitido

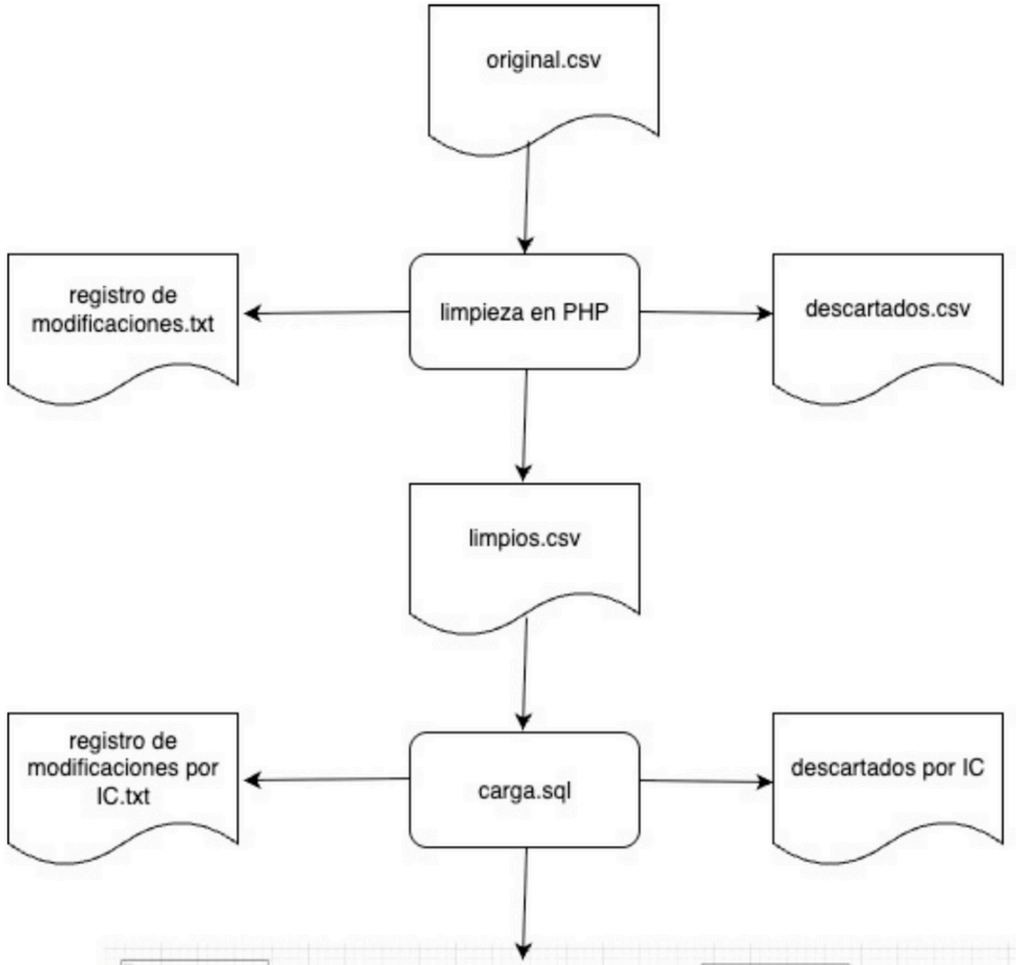
COMO RESOLVEMOS?

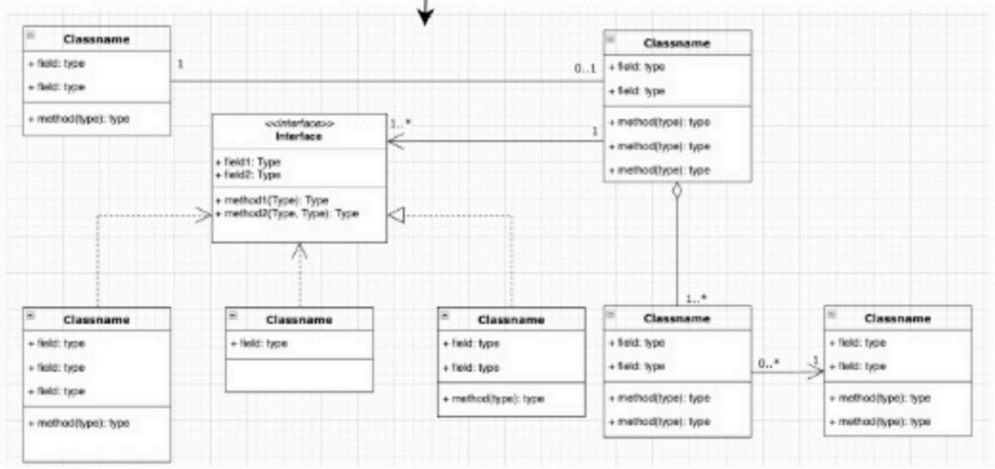
IC	Mejor corregir en	Ejemplo
Datos duplicados	SQL	UNIQUE o PRIMARY KEY
Datos no estandarizados	PHP	Cambiar “EEUU” → “USA”
Tipos de datos incorrectos	SQL	Tipos y CHECK
Datos nulos indebidos	SQL	Telefono VARCHAR(30) NOT NULL.
Incoherencia entre datos	SQL	FK o TRIGGER ej: un paciente debe existir antes de registrar su atencion
Datos fuera del formato permitido	PHP o SQL	Validar telefono o validar correo
Conversion de unidades o formatos	PHP	Pasar de libras a kilos, MM/DD/YYYY → YYYY-MM-DD etc.

DCColita de rana

Esto como se ve...


original.csv
limpieza en PHP
registro de modificaciones.txt
descartados.csv
limpios.csv
carga.sql
registro de modificaciones por IC.txt
descartados por IC


original.csv
registro de modificaciones.txt
limpieza en PHP
descartados.csv
limpios.csv
registro de modificaciones por IC.txt
carga.sql
descartados por IC


Classname
+ field: type
+ field: type
+ method(type): type
<<Interface>>
Interface
+ field1: Type
+ field2: Type
+ method1(Type): Type
+ method2(Type, Type): Type

h

DCColita de rana

Tareas a realizar



Modificar el esquema

Basandose en la E2, deberan crear/modificar su esquema en 3NF o BCND para que pueda almacenar los datos provenientes de los archivos <nombre_archivo>.csv entregados



Limpieza de datos

1.Usando PHP deberan revisar, identificar y corregir, los datos de cada archivo <archivo>.csv y entregar un archivo con datos <archivo>OK.csv

2.Cada modificacion debera ser registrada en el archivo <archivo>LOG.txt.

3.Si una modificacion se repite varias veces, esta debe ser reportada una sola vez.

EJEMPLO EN CODIGO

```php
function parse_correo($str){
    /**
     * Correo
     * Consideraciones:
     * Los dominios de correos permitidos en la pagina web son 'viajes.cl', 'tourket.com',
     * 'wass.com', 'marmol.com', 'outluc.com', 'edubus.cal' y 'viajesanma.com'.
     * Un correo es una letra @ un dominio ejemplo:
     * a@dominio.cl
     */
    if ($str != "" AND strpos($str,"@") != FALSE){
        $arr = explode("@", $str);
        if (strlen($arr[0]) > 0 && in_array($arr[1], [
            "viajes.cl",
            "tourket.com",
            "wass.com",
            "marmol.com",
            "outluc.com",
            "edubus.cal",
            "viajesanma.com"
        ])){
            return $str;
        }else{return FALSE;}
    }else{return FALSE;}
}

```

EJEMPLO EN CODIGO

```php
function parse_fecha($str){
    /**
     * Fecha
     * Consideraciones:
     * ADMITE NULO
     * Debe tener formato YYYY-MM-DD
     * Para preservar la mayor cantidad de datos si un string
     * Documentacion:
     * https://www.php.net/manual/es/function.preg-replace.php
     */
    if (empty($str)){
        return $str;
    }
    $numero_saneado = preg_replace('/[^0-9]/', '', $str);
    $largo_num = strlen($numero_saneado);
    $numero_saneado =substr($numero_saneado, 0, 4)."-".substr($numero_saneado,4,2)."-".substr(
        $numero_saneado, 6);
    return $numero_saneado;

```

EJEMPLO EN CODIGO

```php
function parse_telefono($str){
    /** Telefono
     * Consideraciones:
     * NO ADMITE NULO
     * Debe tener largo 9
     * Si un numero es no nulo pero no es de largo 9 y/o no contiene codigo pais se agregaran 0
     * hasta completar 9 numeros
     * Se tomaran todos los numeros del string y a partir de ahi se ordenara el numero de manera correcta
     * Si un numero es menor que 9 se agregan 0 hasta completar y se asume codigo +569 todo esto con el objetivo de
     * preservar la mayor cantidad de informacion
     * Documentacion:
     * https://www.php.net/manual/es/function.preg-replace.php
     */
    if (empty($str)){
        /*cualquier otro caso para mantener la mayor cantidad de datos, se ignoran los primeros numeros
        y se toman en cuenta solo los 9 ultimos agregando siempre el codigo de pais
        */
        $numero_saneado = preg_replace('/[^0-9]/', '', $str);
        $largo_num = strlen($numero_saneado);
        if ($largo_num < 11){
            $numero_saneado = "569".(8-$largo_num)*"0" . $str;
        }
        $numero_saneado = "+".substr($numero_saneado, 0, 2)." ".substr($numero_saneado,2,1)." ".substr(
            $numero_saneado, 3, 4)." ".substr($numero_saneado, 7);
        return $numero_saneado;



    }else{
        return FALSE;
    }
}
```



Cargar en tablas



Cargar en tablas

1.Crear el esquema con todas sus tablas, atributos, dominios, restricciones, PK, FK, SK, etc.

2.Carga de datos mediante sentencias SQL almacenadas en uno o mas archivos SQL (solo se ejecutara carga.sql, usar \i funciones.sql)



Cargar en tablas

3.El programa carga.sql recibira <archivo>OK.csv, creara el esquema y cargara los datos a la bdd controlando que cumplan con las restricciones de integridad declaradas en este.

4.Las tuplas descartadas deben ser traspasadas al archivo cargaERR.csv y las modificaciones realizadas deben reportarse en cargaLOG.txt



Cargar en tablas

5.Tanto cargaLOG.txt como cargaERR.csv pueden ser uno solo, en este caso deben incluir la identificacion de la tabla en la que se produjo el error y la IC que lo causo.

DCColita de rana

Informe

Informe

1.README.md justificando:

a) el modelo de bdd creado

b)las modificaciones a cada <archivo>.csv (las mismas de <archivo>LOG.txt

c)todas las acciones realizadas a los datos de <archivo>OK.csv efectuadas para ser cargadas en la bdd

2. Todos los .csv originales y procesados, ademas de los .txt y .sql solicitados

3.Instrucciones para ejecutar los dos programas PHP y SQL

DCColita
de rana

Reglas de Negocio

REGLAS DE NEGOCIO

1.Una persona puede ser paciente, trabajar en el centro médico o ambos. Los trabajadores se dividen en staff médico o administrativo.

2.Solo los médico/as tienen especialidad y realizan atenciones médicas, emiten recetas y órdenes.

3.Un staff médico también puede ser paciente, pero en este caso solo puede ser titular en la Institución de salud previsional.

REGLAS DE NEGOCIO

4.Una persona (independiente del rol) puede estar afiliada a una institución de salud previsional (una ISAPRE o FONASA) o no tener previsión, en cuyo caso se considera como particular.

5.Cada ISAPRE tiene un solo plan de salud que especifica el porcentaje de bonificación por cada grupo del Arancel FONASA. Es decir, todos los beneficiarios de esa ISAPRE tienen el mismo plan. Sin embargo, las bonificaciones por grupo varían de una ISAPRE a otra.

REGLAS DE NEGOCIO

6.Si el beneficiario (titular o no titular) está en ISAPRE (no FONASA) paga el valor de la atención y del arancel del centro médico menos la bonificación. Si el beneficiario (titular o no titular) está en FONASA paga solo el arancel FONASA, no tiene bonificación. Si la persona es titular paga el valor completo de la atención. Los medicamentos se pagan al precio de lista de la farmacia independiente de su tipo y previsión de salud.

REGLAS DE NEGOCIO

7.En el contexto de la institución de salud previsional, una persona puede ser Beneficiario titular o solo beneficiario. En este segundo caso la persona tiene asociado un beneficiario titular. Un beneficiario solo puede estar registrado en una institución y el beneficiario está en la misma institución que el titular.

REGLAS DE NEGOCIO

- 8.Toda atención comienza con una consulta médica, en esta el médico realiza un diagnóstico y pueden generar una o más recetas médicas y una o más órdenes de procedimientos o exámenes. Todos los medicamentos de las recetas están registrados en el Maestro farmacia y todos los procedimientos o exámenes están registrados en el Arancel FONASA.

REGLAS DE NEGOCIO

- 9.Las órdenes pueden contener muchos exámenes y procedimientos

- 10.Las recetas de medicamentos no psicotrópicos pueden tener muchos medicamentos, los psicotrópicos van uno por receta.

DCColita de rana

Formato de archivos de datos

PERSONA

Atributo	Tipo de Dato	Longitud/Formato	Descripción
ID	integer	Único	Identificador único de la persona.
RUN	text	10 (XDDDDD-A)	Identificador nacional (Rol Único Nacional).
nombre	text	30	Primer(os) nombre(s).
apellido	text	30	Apellido(s).
dirección	text	100	Dirección de residencia.
correo electrónico	text	30	Nombre de la institución de salud previsional.
telefono	text	30	Nombre de un archivo de imagen (firma digital).
tipo beneficiario	text	30	Nombre de la institución de salud previsional.
titular	text	30	Nombre de un archivo de imagen (firma digital).
rol	text	30	Nombre de la institución de salud previsional.
profesion	text	30	Nombre de un archivo de imagen (firma digital).
especialidad	text	30	Nombre de la institución de salud previsional.
firma	text	30	Nombre de un archivo de imagen (firma digital).
InsSalPrev	text	30	Nombre de la institución de salud previsional.

INSTITUCIONES PREVISONALES DE SALUD

Atributo	Tipo de Dato	Longitud/Formato	Descripción
codigo	integer	MINSAL	Código registrado en el Ministerio de Salud (MINSAL).
nombre	text	30	Nombre de la institución.
tipo	text	abierta, cerrada	Tipo de institución de salud previsional.
RUT	text	12 (XD.DDD.DDD-A)	Rol Único Tributario de la institución.
enlace	text	URL, NULL	Dirección del servidor (ej: http://servidor.TLD).

FARMACIA

Atributo	Tipo de Dato	Longitud/Formato	Descripción
cod	integer	MINSAL	Código genérico del MINSAL.
nombre	text	100	Nombre del producto genérico.
descripcion	text	256	Descripción del producto genérico.
tipo	text	Alimentos, Equipamiento,	Tipo de producto.
codonu	integer	Acepta nulos	Código ONU (Organización de las
clasonu	text	30	Clasificación ONU (acepta nulos si
clasificacion	text	50	Clasificación interna.
estado	text	activo, inactivo	Estado del código genérico.
esencial	integer	0, 1	Indica si está en la Canasta Esencial de
precio	*No especificado*	*No especificado*	Precio del producto genérico.

ARANCEL DCCOLITA

Atributo	Tipo de Dato	Longitud/Formato	Descripción
codigo	integer	Único	Código interno DCColita.
codFonasa	text	codF-adicional	Código Fonasa (con o sin adicional).
atencion	text	100	Nombre de la Consulta y Atención Médica.
valor	integer	Valor	Valor DCColita de la Consulta y Atención Médica.

ARANCEL FONASA

Atributo	Tipo de Dato	Longitud/Formato	Descripción
codF	integer	Único	Código único de la Consulta y Atención Médica
CodA	integer	Adicional	Código adicional (tienen algunos codF)
atencion	text	100	Nombre de la Consulta y Atención Médica
valor	integer	Valor	Valor FONASA de la Consulta y Atención Médica
grupo	text	30	Grupo de Consultas y Atención Médica.
tipo	text	30	Tipo de Consultas y Atención Médica.

ATENCION

Atributo	Tipo de Dato	Longitud/Formato	Descripción
ID	integer	Único	Identificación de la atención.
runpaciente	text	RUN	RUN del paciente.
runmédico	text	RUN	RUN del médico.
diagnostico	text	100	Diagnóstico (solo si la consulta se efectuó). NULL.
efectuada	boolean	true, false	true si la consulta se efectuó (implica que tiene diagnóstico).

MEDICAMENTO

Atributo	Tipo de Dato	Longitud/Formato	Descripción
IDAtencion	integer	ID de la Atención	Clave foránea a la tabla Atencion.
nombre	text	100	Nombre del producto genérico recetado
posología	text	100	Dosis y frecuencia de administración.
psicotropico	boolean	true, false	Indica si el medicamento es psicotrópico
Nota			*Los psicotrópicos van uno por receta (por IDAtencion)*

ORDEN

Atributo	Tipo de Dato	Longitud/Formato	Descripción
IDAtencion	integer	ID de la Atención	Clave foránea a la tabla Atencion.
IDArancel	integer	Código Interno	Clave foránea al codigo de Arancel DCColita.
consulta	text	100	Nombre de la Consulta y Atención Médica.
Nota			*Una orden puede contener muchos exámenes y*

¡MUCHAS GRACIAS!

1.CORRECCION DE DATOS

I. Estandarizacion de formato:

- Datos que represetan lo mismo pero en diferente forma. Deben ser estandarizados. Ejemplo: DD/MM/YY, MM/DD/YYYY
- Datos con errores menores. Son fáciles y rápidos de solucionar

1.CORRECCION DE DATOS

II. Tiposde datos incorrectos:

- El tipo de dato recibido no corresponde al tipo de dato definido

Ejemplo: Se espera un int (500) y se recibe un float (500.0)

- El dato no representa lo pedido, se coloca donde corresponde y se elimina donde no debe

Ejemplo Se espera un apellido (Parra) y se recibe un correo (parra@uc.cl).

2. ASIGNACIÓN DE NULO

Se utiliza en casos que el dato no puede ser corregido, pero no es fundamental para la tupla

Esto se realizan en casos que el valor por default no pueda ser nulo, por lo que se asigna valor decidido por ustedes.
Ejemplo: numero de telefono 5427 se reemplaza por 000000000.

3. ELIMINACIÓN DE DATO

- Esta medida debe aplicarse solamente como último recurso
- ¿Por qué? Siempre debemos piorizar la pérdida mínima de datos
- Consdiderar esta opción solo en casos donde los datos hayan perdido casi todo su valor
- Si se llega a optar por esta opción se debe de eliminar en cascada

DETECCION Y CORRECCION

Las IC pueden ser detectadas y corregidas de 2 maneras:

1. A nivel de aplicacion (en PHP)

2. A nivel de base de datos (con SQL)

PHP Y CARGADORES

¿CÓMO SE HACE?

1. La correccion de los datos con PHP ocurre antes de enviarlos a la BDD con SQL

2. Esto se puede hacer con un script o codigo que valide los datos antes de cargarlos en las tablas

¿CÓMO SE HACE?

CARGADORES

1.Programa encargado de leer los datos e insertarlos

en tablas SQL

2. Revisa que los datos sean correctos para poder insertarlos en las tablas

3. En caso de que un dato este mal, se guarda en un archivo para luego corregirlo o en ULTIMO caso descartarlo.

¿CÓMO SE HACE?

EJEMPLO EN CODIGO

Conexión con la base de datos:

```php
// Inicializamos las variables para la conexión
$host = "localhost"; // dominio de su grupo en ssh
$dbname = "database_name"; // nombre de la base de datos
$user = "username"; // el usuario
$password = "password"; // contraseña

// Establecemos la conexión con la base de datos.
$conn = pg_connect("host=$host dbname=$dbname user=$user password=$password");

// Comprobamos si la conexión se pudo efectuar
// caso contrario terminamos el programa arrojando un error.
if (!$conn) {
    die("Connection failed: " . pg_last_error());
}
```

EJEMPLO EN CODIGO

Creamos las tablas y comprobamos si se crean correctamente:

```php
$sql = "CREATE TABLE IF NOT EXISTS users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)";

// Ejecutamos la query
$result = pg_query($conn, $sql);

// Se comprueba si la query se ejecutó correctamente
if ($result) {
    echo "Tabla 'users' creada correctamente\n";
} else {
    echo "Error creando la tabla: " . pg_last_error($conn) . "\n";
}
```

EJEMPLO EN CODIGO

Insertamos los valores a la tabla y comprobamos su inserción:

```php
// Insertar un valor en la tabla 'users'
$insert_sql = "INSERT INTO users (username, email) VALUES ('testuser', 'testuser@example.com')";
$insert_result = pg_query($conn, $insert_sql);

// Se comprueba si la inserción se ejecutó correctamente
if ($insert_result) {
    echo "Valor insertado correctamente en la tabla 'users'\n";
} else {
    // Incluir lógica adicional para la corrección de errores:
    // añadir a un archivo y luego procesar dichos errores sobre los datos.
    echo "Error insertando el valor: " . pg_last_error($conn) . "\n";
}

// Cerramos la conexión con el servidor
pg_close($conn);
```

EJEMPLO EN CODIGO

Insertamos los valores a la tabla y comprobamos su inserción:

```php
// Insertar un valor en la tabla 'users'
$insert_sql = "INSERT INTO users (username, email) VALUES ('testuser', 'testuser@example.com')";
$insert_result = pg_query($conn, $insert_sql);

// Se comprueba si la inserción se ejecutó correctamente
if ($insert_result) {
    echo "Valor insertado correctamente en la tabla 'users'\n";
} else {
    // Incluir lógica adicional para la corrección de errores:
    // añadir a un archivo y luego procesar dichos errores sobre los datos.
    echo "Error insertando el valor: " . pg_last_error($conn) . "\n";
}

// Cerramos la conexión con el servidor
pg_close($conn);
```

IMPORTANTE!!!

1.La correccion de se hace aparte del cargador!

2.En otras palabras tienen que hacer funciones que “limpien" los datos antes de cargarlos

¿CÓMO SE HACE?

VENTAJAS

1. Flexible, se pueden programar restricciones especificas
2. Evita sobrecargar la bdd con validaciones repetitivas

DESVENTAJAS

1. Depende del codigo del programador, puede fallar y entrar el dato sucio igual

2. No garantiza consistencia total, los datos pueden entrar por otros medios a la bdd (API) o ya estar en esta.

SQL

¿CÓMO SE HACE?

1. Las restricciones se hacen directamente en la estructura o usando comando de SQL

2. Aca la bdd protege su propia integridad, sin depender de factores externos

¿CÓMO SE HACE?

EJEMPLO EN CODIGO

```
22
21
20 -- Crear una tabla Persona donde el telefono sean solo numeros
19 CREATE TABLE Persona (
18   run VARCHAR(9) PRIMARY KEY,
17   telefono CHAR(9) CHECK (telefono ~ '^[0-9]{9}$')
16 );
15
14 -- Revisar que una nota este entre 1 y 5
13 CHECK (calificacion BETWEEN 1 AND 5)
12
11 -- Revisar que un correo sea unico y de no mas de 100 caracteres
10 email VARCHAR(100) UNIQUE NOT NULL
 9
```

EJEMPLO EN CODIGO

```
13
14
15 --Mas complejo, un trigger que revise en cada insercion o update
16 -- que si una receta sea de tipo psicotropica esta tenga codigo de autorizacion
17 CREATE TRIGGER validar_psicotropica
18 BEFORE INSERT OR UPDATE ON Receta
19 FOR EACH ROW
20 WHEN (NEW.es_psicotropica = TRUE AND NEW.codigo_autorizacion IS NULL)
21 BEGIN
22     RAISE EXCEPTION 'Receta psicotropica debe tener codigo de autorizacion';
23 END;
24
```

VENTAJAS

1. Garantiza integridad total, ningun dato sucio puede ingresar independiente de cual sea su origen
2. Estandariza las reglas del negocio a nivel global, las mismas reglas se aplican en toda la bdd
3. Mejora la confiabilidad y consistencia a largo plazo

DESVENTAJAS

1. Menos flexible, modificar las reglas implica modificar la bdd y la estructura SQL

2. Validaciones y/o restricciones mas complejas son exponencialmente mas dificiles de programar

3. Peor rendimiento si se hacen muchas comprobaciones de integridad en tiempo real

ACTIVIDAD: ANALISIS LIMPIEZA

1.

Errores en identificadores de aviones: En una base de datos de vuelos, los códigos de avión presentan inconsistencias: algunos tienen números faltantes (B EING- 47), la marca que existe es (BOEING-747). Hint: Se necesita información adicional.

2.

Uso de diferentes unidades de medida: En un base de datos de productos, algunos pesos están en kilogramos (kg), otros en gramos (g) y algunos en libras (lb). Hint: Hacer la conversión.

ACTIVIDAD: ANALISIS LIMPIEZA

3.
Inconsistencia en nombres de categorías: En una base de datos de clientes, el campo "País" tiene valores como USA, United States y EEUU para referirse a lo mismo. Hint: estandarizar un valor.

4.
Errores en identificadores de vehículos: En una base de datos de automóviles, algunos registros presentan números de chasis escritos con distintos formatos (1HGCM82633A123456, 1HG CM8-2633A 123456 o 1HGCM8263A12345 con un dígito faltante), lo que impide validar el identificador correctamente. Hint: análisis de caso por caso

ACTIVIDAD: ANALISIS LIEMPIEZA

5.
Errores en números de teléfono: En una base de datos de contactos, algunos números de teléfono celular tienen menos de 9 dígitos (12345678) o más de lo esperado (9123456789)

6.
Formato de números incorrecto: En una base de datos financiera, algunos valores monetarios están escritos con coma como separador decimal (1,500.75), mientras que otros usan punto (1500,75), siendo este último el formato correcto.

ACTIVIDAD: ANALISIS LIMPIEZA

7.
Atributo opcional con datos erróneos: En una base de datos de reseñas de productos, algunos registros tienen valores incorrectos en la columna "Calificación", donde se encuentran valores fuera del rango permitido como 8, dado que el formato es de 1 a 5 y la calificación no es un dato obligatorio para el resto de la información. Hint: Anular el dato, datos fuera de rango igual a 0.

8-En la base de datos de empleados, algunas fechas de ingreso están en distintos formatos (2023/01/10, 10-01-2023, enero 10 2023). Hint: usar un solo formato

ACTIVIDAD: ANALISIS LIMPIEZA

9-En latabla de productos,los nombres aparecencon variaciones como " Pan Blanco ", "PAN blanco", "pan Blanco".
Hint: Normalizar texto.

CIERRE

1. Se arregla lo que se puede arreglar, muchas veces obteniendo información adicional. Ejemplo: RUN sin dígito verificador.
2. Si no se puede arreglar:

- Dejarlo en NULL.
- En caso de campo obligatorio (no NULL): valor que “signifique" nulo en nuestro contexto. Ejemplo: teléfono: 999999999.
- Si no es crítico: borrarlo.
- Si es crítico: borrar la tupla (ON DELETE CASCADE).

¿CÓMO SE HACE?

1. Analisis de los datos

   Analizar los datos para ver la detección de errores.

2. Limpieza de Datos.

   Leer los datos desde los archivos y limpiar los datos con un programa de PHP. Entregar los datos en nuevos archivos.

¿CÓMO SE HACE?

¿CÓMO SE HACE?

1. Carga dedatos

   Leer datos desde archivos (.csv) con codigo PHP

2. Insercion en tablas

   Desde el codigo hacemos los INSERTS en las tablas SQL

3. Correccion de datos

   A partir de las violaciones de IC (Integrity Constraints) optar por una de las 3 opciones de correccion

IIC2413

AYUDANTÍA 9

LIMPIEZA DE DATOS