IIC2413

AYUDANTÍA 9

LIMPIEZA DE DATOS

¿QUÉ VEREMOS?

HOY...

Restricciones de Integridad (IC) mas comunes

Teoría carga y limpieza de datos

PHP para el procesamiento

Limpieza en la practica

QUE ES UNA IC?

Las Restricciones de integridad (IC's), son las reglas que garantizan coherencia, calidad y validez de los datos en una BDD.

Estas se definen a nivel estructural (PKs o FKs) o a nivel de negocio (Limites de edad o formatos)

ERRORES EN LOS DATOS (IC)

01

No estandarizados

"Chile"
"chile"
"CHL"

02

Tipos de dato incorrecto

"20"
"veinte"
"20.0"

03

Datos nulos (cuando no se permite)

faltan datos en campos obligatorios

ERRORES EN LOS DATOS (IC)

04

Datos duplicados

Un cliente ingresado 2 veces con el mismo RUT

05

Incoherencia entre datos

Fecha de nacimiento: 02-02-1980 Edad:14

06

Datos fuera del formato permitido

mail: juanpgmail.cm

o una fecha MM/DD cuando solo acepto DD-MM

¿QUE HACER ANTE UN ERROR?

Principalmente nos vamos a enfocar en 3 opciones para corregir un dato.

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

Siempre tenemos que buscar la pérdida mínima de información

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. Tipos de 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?

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;
    }
}
```

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

CUAL ESCOGER?

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.

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, eliminando basura

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 la tabla de productos, los nombres aparecen con variaciones como " Pan Blanco ", "PAN blanco", "pan Blanco". Hint: Normalizar texto.

10- Correos electrónicos inválidos o temporales: En la tabla de usuarios aparecen emails malformados (usuario@@dominio.com, espacio delante/atrás), dominios inexistentes o correos de servicios temporales (ej. mailinator). Hint: Validar correo.

11- Problemas de codificación y caracteres no imprimibles: Nombres y textos que aparecen como "JosÃ©". Hint: Conversion a formato UTF-8

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

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).

¡MUCHAS GRACIAS!