Proyecto de curso Bases de datos

Entrega 3 final

Implementación de una aplicación

Entrega 22 de noviembre 2023 a las 23:59 horas.

1. Objetivos

El objetivo de esta tarea es que el /la estudiante se familiarice con el uso, a través de una aplicación de bases de datos transaccionales para ello utilizarán los conceptos copmo SQL como contraint, triggers, stored procedures y manejo transaccional..vistos en clase para refinar su aplicación aumentando y mejorando funcionalidades al sistema de *Epic Prime*[1].

También conocer de primera mano los problemas de manejo de datos que ocurren al integrar bases de datos en el mundo real y los problemas al recibir bases de datos creadas por terceros. Los grupos deberán afrontar una serie de desafíos ligados a estos problemas.

2. Descripción General

En esta fase implementarán una aplicación que requiere consumir dos bases de datos distintas, conectarse con recursos del sistema operativo, generar transacciones de carga de datos, manejo de errores, transacciones de ventas registro de usuarios etc.

3. Requerimientos

Su aplicación deberá integrar la base de datos de la plataforma de *meta streaming* con la *meta tienda de videojuegos*. Para ello se deben manejar cada una en forma independiente en el servidor respectivo de modo mantener dos conexiones abiertas.

No se permite migrar el esquema a la base de datos de uno de los grupos excepto que se indique expresamente.

[1] Este enunciado es complementario a los enunciados de las entregas 1 y 2

Su aplicación debe implementar las siguientes acciones y requerimientos los que deben verse reflejado en sus bases de datos.

3.1. Usuarios y Acceso

Sistema de usuarios: Su página debe crear un sistema de usuarios, para lo que pueden ocupar la tabla qué guarda los usuarios en la base de datos del grupo **impar**. Las funcionalidades a proveer son las siguientes:

1. **Registrarse en la aplicación**: ingresando todos los datos de un Usuario (**username**, nombre, email, fecha de nacimiento). Se debe verificar que el username ingresado sea único en la BD. el id único a este nuevo usuario debe ser el sucesor del mayor id del sistema. Se deben asignar los permisos que correspondan (read, write, update, etc.) según corresponda, tanto a las tablas como al sistema operativo de modo de poder realizar todas las acciones.

2. **Información personal y *Login*:** Un usuario debe ingresar a la aplicación entregando sus credenciales (**username** y su **contraseña o password)** la cual estará en formato encriptada en la base de datos . Una vez *logeado* deberá ver (desde una página como *Mi Perfil*) la siguiente información:
    a. nombre, email y username.
    b. un listado de suscripciones activas ya sea a juegos o a servicios de streaming ordenado por fecha de compra.
    c. la suma de horas usadas viendo contenido o jugando por separado
    d. la edad
    e. La consulta debe estar en una vista materializada que se refresque en forma diaria.

3.2. Cargas de datos

El grupo deberá generar o modificar los scripts de cargas de datos desde los archivos CSV entregados teniendo en cuenta los tipos de datos y las restricciones que se aplican a los atributos, así como el manejo de errores que se pudieran producir. En caso de error el script debe tomar una de estas 3 opciones:

a) detenerse y deshacer completamente la transacción (es decir dejar las tablas involucradas vacías) corregir el dato y volver a cargar completa.
b) En forma alternativa puede detenerse en el primer error para corregir y continuar desde ese punto dejando los datos correctos
c) como tercera opción generar un log de errores para corregir y luego cargar solamente los esos registros.

1. Importación de usuarios: Usando php debe generar un cargador que extraiga todos los usuarios presentes en la base de datos del grupo **par** (si aun no están) en la tabla Usuarios del grupo **impar**). Para simplificar la tarea, los username, clave y correo coinciden en ambas bases para los mismos usuarios.

2. Adicionalmente todos los pagos en la base de datos del grupo **par** deben ser migrados a la base de datos **impar** para que haya una sola fuente centralizada con la información de los pagos.
3. Se deben encriptar todas las claves de los usuarios (crypt)

3.3. Navegación

Página de suscripciones: los usuarios deben poder ver una lista con todos los proveedores de streaming y videojuegos. Para cada uno, al hacer click en uno se debe desplegar una página con:

a. Valor de la suscripción y cantidad de películas, series y video juegos incluidas.
b. Un botón que al hacerle click se muestren las 3 películas, las 3 series más vistas o los juegos mas jugados en la plataforma que están incluidas en el plan de ese proveedor. Se deben separar claramente e indicar puntuación y cantidad de visualizaciones para cada una. Para las series, se debe considerar la suma de visualizaciones de sus capítulos.
c. Un buscador con dos campos, proveedor y nombre, que permita buscar para cada proveedor si la película, serie o videojuego ingresada en el nombre está incluida en el plan del proveedor o no. Se deben desplegar los resultados indicando visualmente si se incluye o no. La búsqueda debe ser *case insensitive* y con *matching parcial* como en la entrega anterior.

3.4. Compras, página de *one time purchases*:

En esta página los usuarios podrán hacer compras simultáneas en ambos servicios, para ello los usuarios deben poder ver una lista de aquellas películas o juegos que se compran fuera de un plan de suscripción. Para cada uno, se debe poder hacer click en ellos lo que llevará a una página con:

1. Los detalles del contenido y los precios en los distintos proveedores
2. Un campo con un dropdown para seleccionar el proveedor y un botón para comprar el contenido. Al seleccionar este botón debe ocurrir lo siguiente:

a. Se debe verificar que el proveedor ofrezca el contenido
b. Que la suscripción esté vigente (fecha de fin >= TODAY)
c. Se debe comprobar que el usuario no lo tenga en su librería con vigencia
d. Se debe generar una compra (o pago) en las tablas del grupo **impar** que estén involucradas. Se debe generar un **trigger** que actualice todas tablas afectadas por las ventas.

e. Se debe generar un **store procedure** al final de la compra que genere una página con el resumen de todas las compras ya validadas y contenga los botones cancelar y OK para cancelar toda la compra o ejecutarla respectivamente.

f. La compra debe estar contenida en una **transaction** de sql (BEGIN TRANSACTION—COMMIT

3.5. Página consulta inestructurada

En esta página Ud. deberá implementar una consulta inestructurada de tipo SELECT y recibir como parámetro los atributos de una tabla (incluyendo *), el nombre de la tabla y el criterio dejando el resultado en el archivo consulta.txt. Por ejemplo: SELECT * from proveedores where costo < 5000. Debe incluir todas las validaciones necesarias.

3.6. Cancelación de suscripciones

A las 00:00 de cada día se deben marcar como canceladas todas las suscripciones que vencieron el día anterior.
Una subscripción puede estar cancelada pero seguir estando activa hasta la fecha de finalización

3.7. Creatividad (Funcionalidad Adicional)

Debe agregar una funcionalidad adicional que no esté listada en el enunciado y que aporte valor a la aplicación. Esta consulta de hacer uso de un UNCLUSTERED INDEX.

3.8. BONUS

Bonus: No se evaluará la calidad de la página, ni su atractivo visual; sin embargo, los ayudantes tendrán la facultad de ofrecer un bonus a aquellos grupos que presenten una página con una navegación sobresaliente. Deberá notarse un trabajo adicional al de la entrega anterior.

Se evaluará que toda la información necesaria esté en tablas y no en código (en duro)

NOTA: Califican para este bonus solo los proyectos que implementen correctamente todo el proyecto.

4. Detalles Administrativos

Se deberá juntar un grupo **par** con uno **impar**. Los grupos que deseen trabajar juntos podrán inscribirse en este **formulario** hasta el día **25 de octubre**, de lo contrario serán asignadas al azar. Las parejas de grupos definitivas serán publicadas el día **27 de octubre.**

4.1. Ubicación de su entrega

1. La nueva versión de la app debe estar ubicada en el directorio **/home/grupoXX/Sites/**. Esto significa que el homepage debe ser **/home/grupoXX/Sites/index.php**. Usar cualquier otra ruta arriesga que el proyecto no sea corregido.

2. El procedimiento almacenado debe estar (o debe haber una copia) en la carpeta **/home/grupoXX/Entrega3/**

Donde XX es el número del **menor** de los dos grupos unidos. Se asumirá también que la página principal de su sitio esta está ubicada en **Bachman.ing.puc.cl/~grupoXX/index.php**

Adicionalmente, se les ha habilitado una base de datos denominada **grupoXXe3** que posee las tablas y datos de su base anterior en donde deberán trabajar a lo largo de esta entrega.

Importante: **Esta entrega requiere una coordinación entre los grupos que no siempre es sencilla. Se recomienda planificar con anterioridad y utilizar el todo el plazo**

4.2. Entrega

Debe entregar los siguientes items:

Aplicación Web: Todos los archivos necesarios para el correcto funcionamiento de la aplicación en su carpeta *Sites*.

Triggers y Procedimiento Almacenado: Archivo(s) con la definición de su procedimiento almacenado, en lenguage **PL/pgSQL**. Ubicado junto al Readme en la carpeta Entrega3.

Código de los cargadores de datos y archivos de datos utilizados (se evaluará con los archivos de datos originales y modificados si procede)

Readme.md o Reporte.pdf: Donde indique:

d. cómo *logearse* en la aplicación (contraseñas),
e. cómo evaluar la funcionalidad adicional que implementaron
f. cómo evaluar cada requerimiento de esta entrega si no es intuitivo desde la *homepage*.
g. especificar cómo reasignaron contraseñas a los usuarios existentes del grupo **impar**, y la derivación que usaron para asignar las nuevas a los del grupo **par**.
h. La estrategia seguida para corregir las inconsistencias y errores de los datos.

i. Este archivo debe quedar ubicado en la carpeta Entrega3[2](afuera de Sites).
j. Los ayudantes se reservan el derecho a descontar puntajes en una entrega en la que se haya dificultado la corrección.

4.3. Período de consultas

Las consultas al proyecto se recibirán hasta el día 20 de noviembre a las 11:59 AM exclusivamente a través de *issues* en GITHUB

4.4. Evaluación de pares intergrupo

Cómo aquí trabajarán en grupos, también existe una evaluación de pares, esto puede afectar a su nota (y disminuirla) si sus pares encuentran qué no aportaron al proyecto. Para aprobar el proyecto, deben contar con una evaluación promedio de al menos 50% entre todos sus pares.

5. Diccionario de datos

NOTAS:
cuando aparezca NN significa un CONSTRAIN NOT NULL
todos los ID son NOT NULL cuando definen la entidad

IMPARES
Genero, subgenero:
genero: TEXT NN Nombre que agrupa a las películas según su temática
subgenero: nombre que especifica una película según su temática dentro de un género
CONSTRAIN genero >< subgenero

Multimedia
pid: INT Id de la película CONSTRAIN las películas no tienen episodios. Los campos pid y sid+cid son excluyentes
sid, INT Id de la serie, pueden tener episodios con los mismos nombres y títulos diferentes. Los campos pid y sid+cid son excluyentes
cid,INT número del capítulo de la serie. Los campos pid y sid+cid son excluyentes
titulo : TEXT NN Título de la película/episodio
duracion, FLOAT NN duración de la la película/episodio en minutos
clasificación: NN Clasificación cinematográfica de la MPA CONSTRAIN valores posibles G, PG, PG-13,R,NC-17
puntuación: FLOAT valoración del (PÚBLICO/CRITICA)
año: INT NN Año de estreno CONSTRAIN en películas el año actual si es superior es solo preventa

[2] Si tienen dudas con respecto a la ubicación de esta carpeta les recomendados revisar el readme de la carpeta de la entrega ubicada al interior del repositorio del curso

numero: INT número de episodio CONSTRAIN solo válido para series
serie,: Nombre de la serie CONSTRAIN solo válido para series y debe tener lleno el campo título
genero: CONSTRAIN solo géneros o subgéneros existentes en BD

Pagos

pago_id: INT NN identificador del pago,
monto: FLOAT NN monto del pago
fecha: DATE NN fecha del pago CONSTRAIN debe ser igual o anterior a la actual
uid, INT NN id del usuario CONSTRAIN debe existir en la BD
subs_id, INT NN id de la suscripción CONSTRAIN debe existir en la BD
pid, INT Id NN de película CONSTRAIN debe existir en la BD
pro_id INT NN id de proveedor CONSTRAIN debe existir en la BD

Proveedores

id, INT NN id del proveedor
nombre: TEXT NN nombre del proveedor
costo: FLOAT costo de la serie CONSTRAIN (costo o precio debe existir en la BD)
sid: INT Id de la serie CONSTRAIN debe existir en la BD
pid,:INT id de la película CONSTRAIN debe existir en la BD
precio: FLOAT precio mensual de la suscripción
disponibilidad: INT Cantidad de películas disponibles (debe existir para cada pid)

Suscripciones

id: INT NN de la suscripción
estado: BOOLEAN NN (o TRUE/FALSE)
fecha_inicio: DATE NN fecha de inicio de la suscripción CONSTRAIN debe ser igual o menor de la fecha actual y fecha fin > fecha inicio >=1 mes
pro_id: INT NN id del proveedor CONSTRAIN debe existir en la BD
uid: INT NN id del usuario CONSTRAIN debe existir en la BD
fecha_termino: DATE NN fecha de término de la suscripción CONSTRAIN fecha fin > fecha inicio >=1 mes
proveedor: TEXT NN Nombre del proveedor CONSTRAIN debe existir en la BD
costo: INT NN Costo de la suscripción mensual

Usuarios

id: INT NN id del usuario
nombre: TEXT NN UNIQUE nombre del usuario UNICO EN EL SISTEMA
mail: TEXT NN correo electrónico CONSTRAIN mínimo largo 4 formato x@y.TLD
password: TEXT NN Password registrado en la BD encriptado
username: TEXT NN UNIQUE username
fecha de nacimiento: DATE NN fecha de nascimiento del usuario CONSTRAIN <= TODAY

Visualizaciones

uid:INT NN Id de usuario CONSTRAIN debe existir en la BD
pid, INT Id de película CONSTRAIN debe existir en la BD
sid: INT id de la serie, CONSTRAIN debe existir en la BD
cid: INT Número de capítulo de la serie CONSTRAIN el par sid-cid debe existir en la BD
fecha: DATE NN fecha de la visualización CONSTRAIN (fecha <= fecha actual) y ((suscripciones.fecha_inicio <=fecha <=suscripciones.fecha_termino) o (pagos.fecha <= fecha))

PARES

Genero_subgenero

genero: TEXT NN Nombre que agrupa a las películas según su temática
subgenero:TEXT nombre que especifica una película según su temática dentro de un género CONSTRAIN genero <> subgenero

Pagos

pago_id: INT NN identificador del pago,
monto: FLOAT NN monto del pago
fecha: DATE NN fecha del pago CONSTRAIN debe ser igual o menor de la fecha actual
id_usuario, INT NN id del usuario
preorden: BOOLEAN TRUE/FALSE
id_proveedor:INT NN id de proveedor CONSTRAIN debe existir en la BD
id_videojuego:INT Id de videojuego CONSTRAIN debe existir en la BD (id_videojuego y subs_id excluyente)
subs_id, INT id de la suscripción CONSTRAIN debe existir en la BD

Proveedores

id: INT NN id del proveedor
nombre: TEXT NN UNIQUE nombre del proveedor
plataforma: TEXT NN Nombre de la plataforma CONSTRAIN debe existir en la BD
id_videojuego: INT NN id del videojuego CONSTRAIN debe existir en la BD
precio, FLOAT NN costo del videojuego
precio_preorden FLOAT costo del videojuego en preorden

Subscripciones

id: INT NN id de la suscripción
estado: BOOLEAN NN ACTIVA/CANCELADA (o TRUE /FALSE)

fecha_inicio, DATE NN CONSTRAIN debe ser igual o menor de la fecha actual DATE fecha de inicio de la suscripción CONSTRAIN fecha fin > fecha inicio y fecha_fecha.termino-fecha_inicio>=1 mes

id_usuario: INT NN Id del usuario CONSTRAIN debe existir en la BD

fecha_termino: DATE NN fecha de inicio de la suscripción CONSTRAIN fecha fin > fecha inicio >=1 mes

id_videojuego: INT Id del video juego CONSTRAIN debe existir en la BD

mensualidad: FLOAT Precio de la mensualidad

Usuario proveedores

id_usuario: INT NN Id del usuario CONSTRAIN debe existir en la BD

id_proveedor: INT Id del proveedor CONSTRAIN debe existir en la BD

Usuarios_actividades

id_usuario: INT NN id del usuario:

nombre: TEXT NN nombre del usuario

mail, TEXT correo electrónico CONSTRAIN mínimo largo 4 formato x@y.TLD

password: TEXT Password debe almacenarse encriptado

username: TEXT UNIQUE username único en el sistema

fecha_de_nacimiento: DATE NN <= TODAY

id_videojuego INT NN Id del videojuego

fecha: DATE NN fecha de la visualización CONSTRAIN (fecha <= fecha actual) y ((suscripciones.fecha_inicio <=fecha <=suscripciones<=fecha_termino) o (pagos.fecha <= fecha))

cantidad: FLOAT NN cantidad de horas jugadas

veredicto: BOOLEAN POSITIVO/NEGATIVO (o true false) calificación general del juego

titulo: TEXT título del veredicto

texto: TEXT Descripción del veredicto

Video juego genero

id_videojuego: INT NN id del video juego

nombre: TEXT NN género o subgénero del video juego debe existir en la D

Videojuego

id_videojuego: INT NN id del video juego CONSTRAIN debe existir en la BD

titulo: TEXT NN nombre del video juego CONSTRAIN debe existir en la BD

puntuación: INT NN Puntuación del videojuego

clasificación: TEXT NN CONSTRAIN valores de ESRB o PEGI

fecha_de_lanzamiento: DATE NN fecha de lanzamiento del juego

beneficio_preorden TEXT beneficios de preorden

mensualidad: FLOAT NN valor de la suscripción
nombre: TEXT NN Género o subgénero CONSTRAIN debe existir en la BD

PUNTUACiÓN

ENTREGA FINAL

Evaluación

1. Modelo E/R de las bases de datos integradas 10pt
2. Bases de datos en 3NF o BCNF 20pt
3. Manejo de constrain de los atributos y entre atributos 10pt
4. Consultas integradas en ambas bases de datos 30pt
5. Implementación de cargadores de datos 10pt
6. Implementación de triggers 10 pt
7. Implementación de Store Procedures 10pt
8. Uso de transacciones 10pt
9. Creatividad 10 pt
10. BONUS: 10 pt

puntaje máximo sin bonus 120 pt

P3= (puntaje+bonus)/máximo*6+1, nota del grupo

$P3_i$= N3*evaluación de pares nota del estudiante i