SQL Coderhouse - Entrega Final

01-08-2025
Nicolás Vera
Comisión 81855

1. Introducción

En el siguiente proyecto se desarrollará un esquema de base de datos para “Xue Xiao”, una empresa ficticia dedicada a la educación en línea. El objetivo es diseñar una estructura que permita organizar y administrar la información relacionada a los cursos, planes, usuarios y pagos dentro de su plataforma.

El **objetivo** será diseñar un modelo de base de datos relacional que permita a “Xue Xiao” administrar de forma eficiente su operación comercial, contable y estratégica. Para ello, se trabajarán los siguientes puntos:

- Identificar correctamente la información de los usuarios, incluyendo quiénes son, qué han comprado y qué han utilizado en la plataforma.
- Registrar y organizar todos los pagos realizados, con el fin de entregar esta información al área de contabilidad de manera clara y ordenada.
- Mantener visibilidad de los planes y precios disponibles en cada país, manteniendo así una uniformidad en el proceso de validación.
- Estructurar la información de manera que facilite la generación de informes útiles para el área de gerencia, apoyando así a la toma de decisiones estratégicas.

2. Situación Problemática

La empresa ficticia “Shue Chao” es una plataforma educativa que vende cursos asincrónicos de reforzamiento académico (ver más detalles en la sección “Modelo de Negocios”). Esta comenzó como algo muy pequeño, por lo que originalmente se administró todo a partir de registros manuales y sin una estructura determinada. En los últimos años esta empresa ha aumentado sus ventas significativamente, lo cual ha generado que la administración manual se vuelva inviable y complicada de manejar. Esta información es clave para validar las compras de los usuarios, y así determinar qué cursos deben tener habilitados los distintos usuarios. Por lo mismo, se necesita de manera urgente estructurar y normalizar la información, para así poder manejar el crecimiento de la demanda.

Al implementar este cambio, la empresa será capaz de:

- Consultar y validar la información de manera clara y rápida.
- Tener información consistente con los parámetros indicados, minimizando errores.

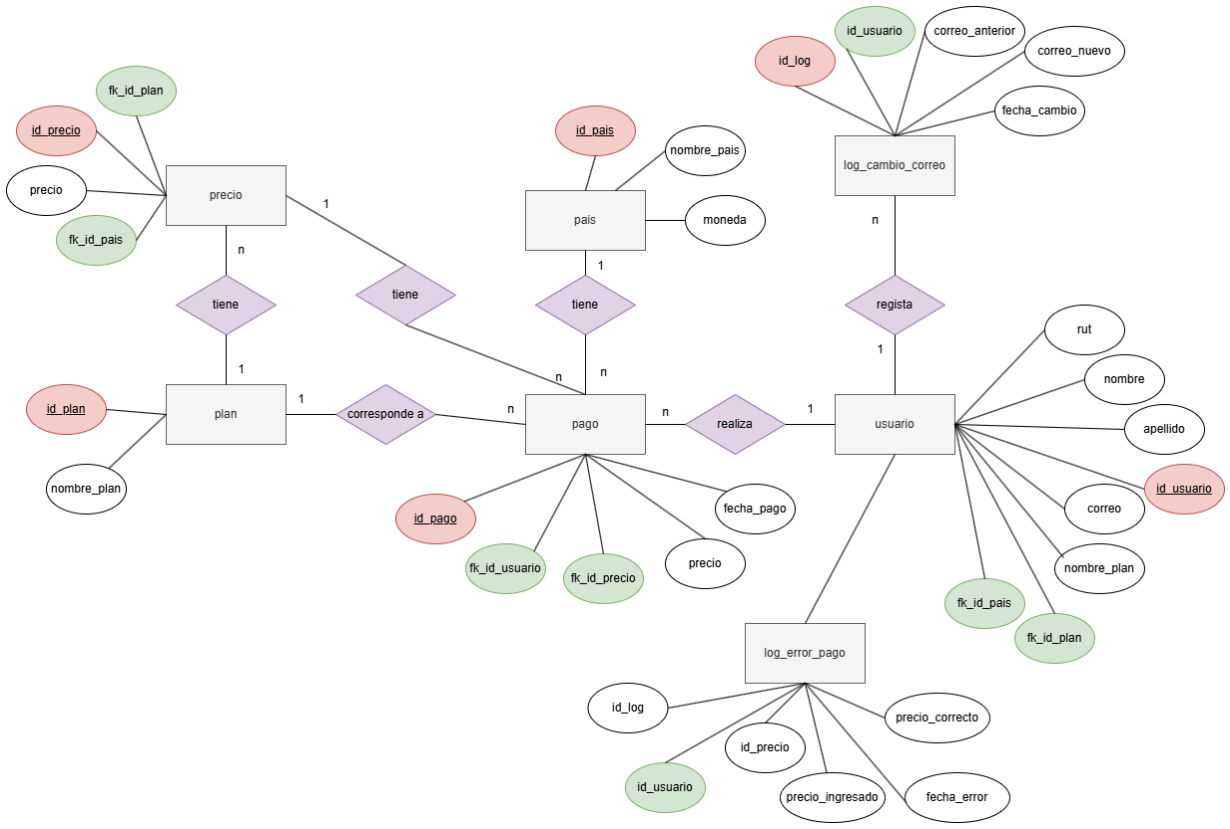
- Generar reportería para tomar mejores decisiones en las áreas de ventas, comercial y marketing.
- Mejorar el sistema contable de la empresa, al identificar correctamente los pagos y fechas respectivas.

3. Modelo de negocios

Empresa de educación que ofrece cursos asincrónicos en línea. Esta tiene una plataforma online en donde los usuarios pueden conectarse a ver estos cursos, realizar actividades y obtener calificaciones. Sus cursos están enfocados en estudiantes de primaria y secundaria en distintos países de latinoamérica, en donde su principal foco se encuentra en Chile y Argentina. Los estudiantes pagan una cantidad de dinero para obtener un curso, al cual una vez pagado puede acceder de por vida. Los cursos tienen el objetivo de reforzar distintas materias del colegio. De esta manera, se tiene una inversión inicial en la creación de cursos, para luego recuperar el dinero vendiendo muchos cursos a bajo costo.

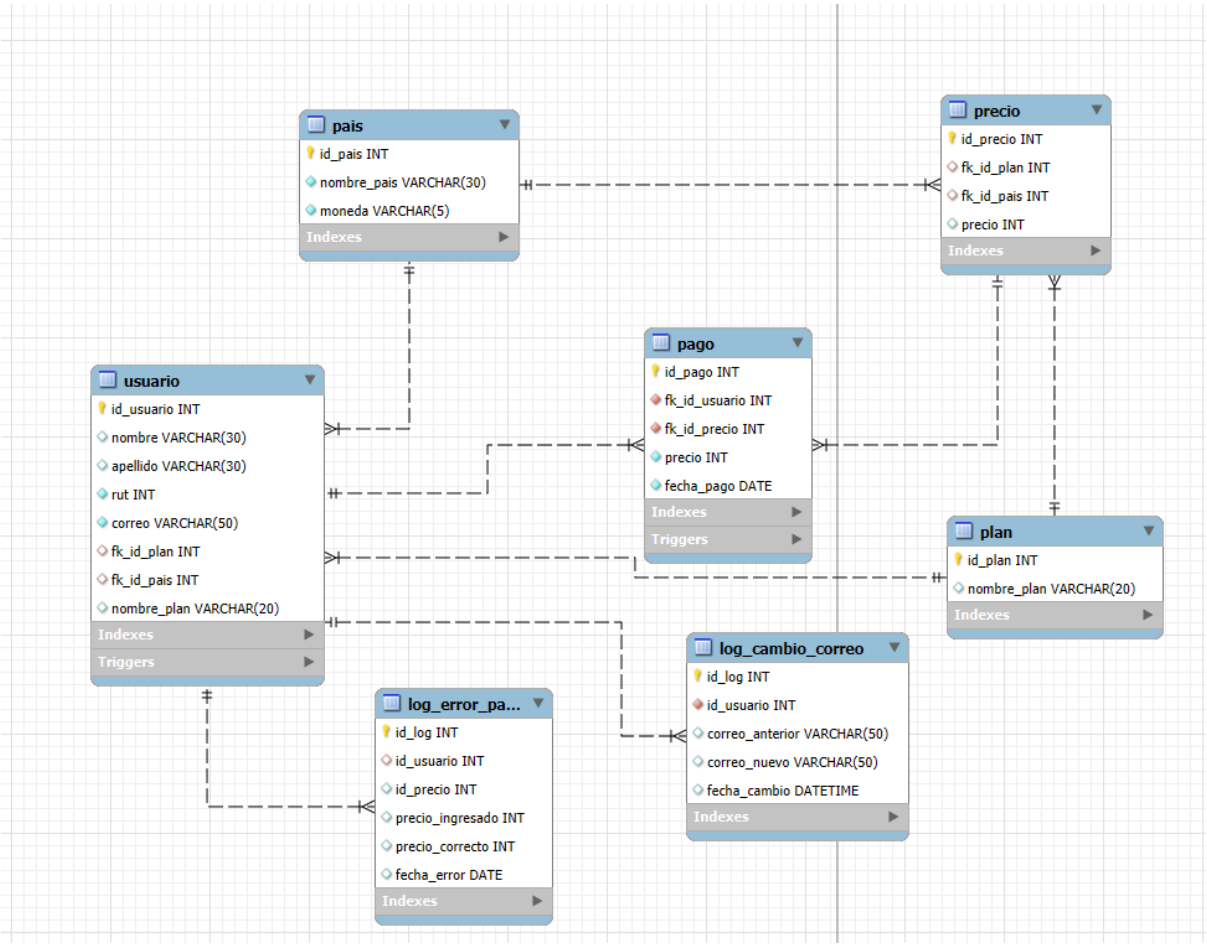
4. Diagrama E-R (Entidad-Relación)

A continuación, se adjunta el diagrama de la base de datos. En esta se marcó en rojo las Primary Key, en verde las Foreign Key y en blanco las entidades que no cumplen ninguna de las anteriores.



Rojo: *Primary Keys;* ***Verde:*** *Foreign Keys;* ***Blanco:*** *entidades sin PK o FK.*

Reverse Engineer



5. Listado de Tablas

Las tablas a crearse originalmente son las señaladas a continuación.

a) ***'plan'*:** En esta se identifican los cursos disponibles para los usuarios de la plataforma. Hay 3 opciones:
   i) basic
   ii) lite
   iii) full

Columna	Datatype	PK	FK	NN	UQ	B	AI
id_plan	INT	X					X
nombre_plan	VARCHAR(20)				X		

b) ***‘usuario”***: Información de los usuarios de la plataforma

Columna	Datatype	PK	FK	NN	UQ	B	AI
id_usuario	INT	X					X
nombre	VARCHAR(30)						
apellido	VARCHAR(30)						
rut	INT			X			
correo	VARCHAR(50)			X	X		
nombre_plan	VARCHAR(20)		X				
fk_id_plan	INT		X				
fk_id_pais	INT		X				

c) ***‘pais’:*** Notación de los paises y monedas de pago aceptadas en cada una.

Columna	Datatype	PK	FK	NN	UQ	B	AI
id_pais	INT	X					X
nombre_pais	VARCHAR(30)			X	X		
moneda	VARCHAR(5)		X	X	X		

d) ***‘precio’:*** Tabla que indica los precios de cada plan habilitado en la plataforma.

Columna	Datatype	PK	FK	NN	UQ	B	AI
id_precio	INT	X					X
fk_id_plan	INT		X				
fk_id_pais	INT		X				
precio	INT						
moneda	VARCHAR(5)		X		X		

e) ***'pago'*****:** Registro de los pagos generados por los usuarios según la fecha en la que se realizó.

Columna	Datatype	PK	FK	NN	UQ	B	AI
id_pago	INT	X					X
fk_id_usuario	INT		X	X			
fk_id_precio			X	X			
precio	INT			X			
fecha_pago	DATETIME			X			

f) ***'log_error_pago'***: Registro de los pagos ingresados con un precio incorrecto, distinto al vigente en la tabla 'precio'.

Columna	Datatype	PK	FK	NN	UQ	B	AI
id_log	INT	X					X
id_usuario	INT		X				
id_precio	INT						
precio_ingresado	INT						
precio_correcto	INT						
fecha_error	DATE						

g) ***'log_cambio_correo'***: Registro de los correos cambiados por usuarios durante el tiempo, para así no perder cualquier posible rastro del usuario asociado a este.

Columna	Datatype	PK	FK	NN	UQ	B	AI
id_log	INT	X					X
id_usuario	INT		X	X			
correo_anterior	VARCHAR(50)						
correo_nuevo	VARCHAR(50)						
fecha_cambio	DATE						

6. Vistas

a. ‘ventas_historicas_por_plan’: Esta vista presenta las ventas diarias agrupadas por tipo de plan y moneda. Permite identificar qué producto tuvo mayor demanda en una fecha determinada, aportando información útil para decisiones estratégicas de precios, promociones o expansión de planes.

b. ‘detalle_planes’: Vista descriptiva que enlaza cada plan con su respectiva moneda y país. Resulta útil para comprender cómo varían los planes según el contexto geográfico o económico.

c. **‘venta_usuario_ranking’**: Genera un ranking de usuarios ordenados según el total pagado en la plataforma. Esta vista facilita el reconocimiento de los clientes más activos o rentables, permitiendo, por ejemplo, aplicar programas de fidelización o promociones personalizadas.

7. Funciones

a. ‘obtener_nombre’: Función que recibe un verificador nacional (RUT) y devuelve el nombre completo del usuario. Si no se encuentra, se retorna un mensaje indicando que el usuario no fue hallado. Esta función mejora la utilización del código en la reportería.

b. **‘tabla_pagado_por_usuario’**: Función que calcula el total acumulado pagado por un usuario, identificándolo por su correo. Es especialmente útil para reportes personalizados o cálculos de facturación por cliente.

8. Store Procedures

a. ‘ventas_por_rango’: Store Procedure que permite consultas las ventas dentro de un rango de fechas y para una moneda específica. Complementa la vista ‘ventas_historicas_por_plan’ para facilitar la generación de reportes.

b. **‘correccion_monto_pago’:** Permite actualizar el valor de un pago registrado previamente en la tabla ‘pago’. Útil para corregir errores o aplicar descuentos posteriores.

9. Triggers

a. ‘validar_precio_pago’: Trigger que valida si la variable ‘precio’ de los datos ingresados en la tabla ‘pago’ coincide con el que está en ese momento en la tabla ‘precio’. Esto se hace debido a que los precios pueden ir cambiando constantemente en el tiempo, por lo que se debe ir validando si los nuevos ingresos tienen el precio actual, sin que esto implique un problema con los datos ya ingresados.

b. **‘trigger_log_cambio_correo’:** Trigger que se activa cuando un usuario modifica su correo. Inserta automáticamente un registro en la tabla ‘log_cambio_correo’, permitiendo rastrear cambios históricos y mantener control sobre la información de contacto del usuario. Esto es clave para procesos de auditoría y recuperación de cuentas.

10. Enlaces

- **Github**: https://github.com/exilium18/entrega_3_sql