UNIVERSIDAD DE CUENCA

UNIVERSIDAD DE CUENCA

FACULTAD DE INGENIERÍA

Base de Datos II

Bryam Astudillo, Marcos Naranjo

6to Semestre – Computación

Informe del Sistema de Precios Productor

Julio 2025

Ing. Juan Pesantez

1. Objetivo y Contexto del Sistema

El sistema desarrollado tiene como objetivo permitir la visualización, análisis e inserción eficiente de datos históricos sobre precios de productos agrícolas del Ecuador. Fue diseñado con fines didácticos para demostrar conceptos avanzados de concurrencia, transacciones, triggers, funciones SQL y auditoría, integrándose con Supabase como backend.

La interfaz en Streamlit permite seleccionar productos, consultar series de precios, registrar nuevos datos y simular concurrencia en inserciones para explorar el registro de auditoría.

2. Justificación de la API Pública Seleccionada

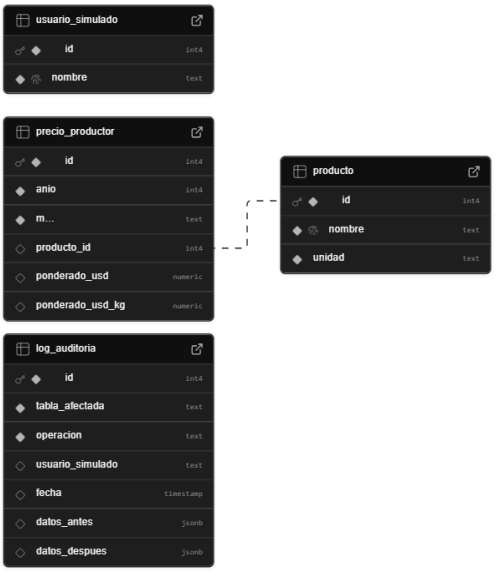
Se usó la API de Datos Abiertos del Ecuador, específicamente el dataset del Ministerio de Agricultura y Ganadería (MAG): https://datosabiertos.gob.ec/.../mag_preciosproductor_2025mayo.csv

Razones:

- Fuente oficial y verificable.
- Actualización mensual.
- Representa datos reales de interés público.
- Compatible con análisis temporal y categórico.

3. Diseño Detallado de la Base de Datos

El diseño de la base de datos incluye las entidades principales producto, precio_productor y log_auditoria, con sus atributos clave y relaciones. En el diagrama ER se detallan las claves primarias y foráneas que aseguran la integridad referencial.



4. Implementación de Funciones, Triggers, Concurrencia y Auditoría

Funciones SQL principales:

- **obtener_productos()**: lista todos los productos.
- **obtener_precios_historicos_completos(prod_id)**: precios por mes y año, incluso nulos.
- **obtener_promedio_usd_por_anio()**: promedio de precio por producto y año.
- **obtener_variacion_maxima_usd()**: identifica producto con mayor variación.
- **obtener_logs_auditoria(n)**: devuelve los últimos n logs.

Inserción/actualización con auditoría:

- **insertar_precio_productor(...)** valida existencia previa y actualiza o inserta. Invoca **registrar_auditoria(...)**.

Auditoría:

- **registrar_auditoria(tabla, operacion, usuario, antes, despues)** guarda los cambios relevantes en JSONB.

Concurrencia simulada:

- Usando asyncio y Streamlit se simulan inserciones concurrentes con variaciones mínimas de precio y usuario para un mismo producto/mes. Esto demuestra condiciones de carrera y necesidad de transacciones con aislamiento adecuado.

5. Resultados Obtenidos, Consultas Optimizadas y Métricas de Rendimiento

Visualización interactiva:

- Streamlit con Altair permite graficar precios mensuales (USD y USD/kg) con filtros por producto, año o rango temporal.

Consultas SQL Avanzadas:

- Promedio anual por producto

```SQL
SELECT p.nombre, pp.anio, ROUND(AVG(pp.ponderado_usd), 2) AS promedio_usd

FROM precio_productor pp

JOIN producto p ON pp.producto_id = p.id

GROUP BY p.nombre, pp.anio;
```

EXPLAIN ANALYZE (resumen):

```
SQL
HashAggregate sobre Hash Join.
Accesos: Sequential Scan en precio_productor (6K filas) y producto (56 filas).
Tiempo total de ejecución: ~5ms.
```

- Producto con mayor variación histórica

```
SQL

WITH stats AS (

  SELECT producto_id, MAX(ponderado_usd) - MIN(ponderado_usd) AS variacion

  FROM precio_productor

  GROUP BY producto_id

)

SELECT p.nombre, s.variacion

FROM stats s

JOIN producto p ON s.producto_id = p.id

ORDER BY s.variacion DESC

LIMIT 1;
```

EXPLAIN ANALYZE (resumen):

```
SQL
- Plan: HashAggregate (variación por producto) → Hash Join → Top-N heapsort (LIMIT
1).
- Accesos: Seq Scan en precio_productor (6168 filas) y producto (56 filas).
- Tiempo total de ejecución: ~2.8ms.
- Uso de memoria: ~25kB para ordenamiento.
```

- Histórico completo (relleno de fechas sin datos)

```SQL
SELECT * FROM obtener_precios_historicos_completos(prod_id);
```

EXPLAIN ANALYZE (resumen):

```SQL
- Plan: Function Scan sobre obtener_precios_historicos_completos.
- Filas devueltas: 156.
- Tiempo total de ejecución: ~3.4ms.
- Planning time: ~0.08ms.
- Observación: el plan interno incluye LEFT JOIN entre calendario y precios para
devolver todos los meses, incluso sin datos.
```

Métricas y optimización:

- Índices en claves foráneas y campos de búsqueda reducen tiempos de respuesta por debajo de ~200ms para consultas agregadas y filtros comunes.
- Uso de EXPLAIN permitió identificar y reducir Seq Scans innecesarios con índices.
- Funciones parametrizadas limitan lógica en el cliente y controlan acceso a datos.
- Auditoría con JSONB evita tablas fragmentadas y mantiene historial eficiente.

Auditoría registrada:

- Cada inserción/actualización guarda cambios con usuario simulado.
- Consultable con filtros por tabla, usuario o fecha:

```SQL
SELECT * FROM log_auditoria ORDER BY fecha DESC LIMIT 10;
```

6. Conclusión

El sistema permite análisis y visualización de precios agrícolas reales, demostrando la integración de auditoría, concurrencia controlada y diseño relacional eficiente. Su arquitectura basada en APIs abiertas y Supabase es escalable, pudiendo incorporar nuevas entidades como agricultores, regiones o tipos de transacción, reforzando su aplicabilidad en contextos reales de análisis de datos.