Trabajo Final Integrador (TFI) – Bases de Datos I

Gestión de Empresas y Domicilios Fiscales

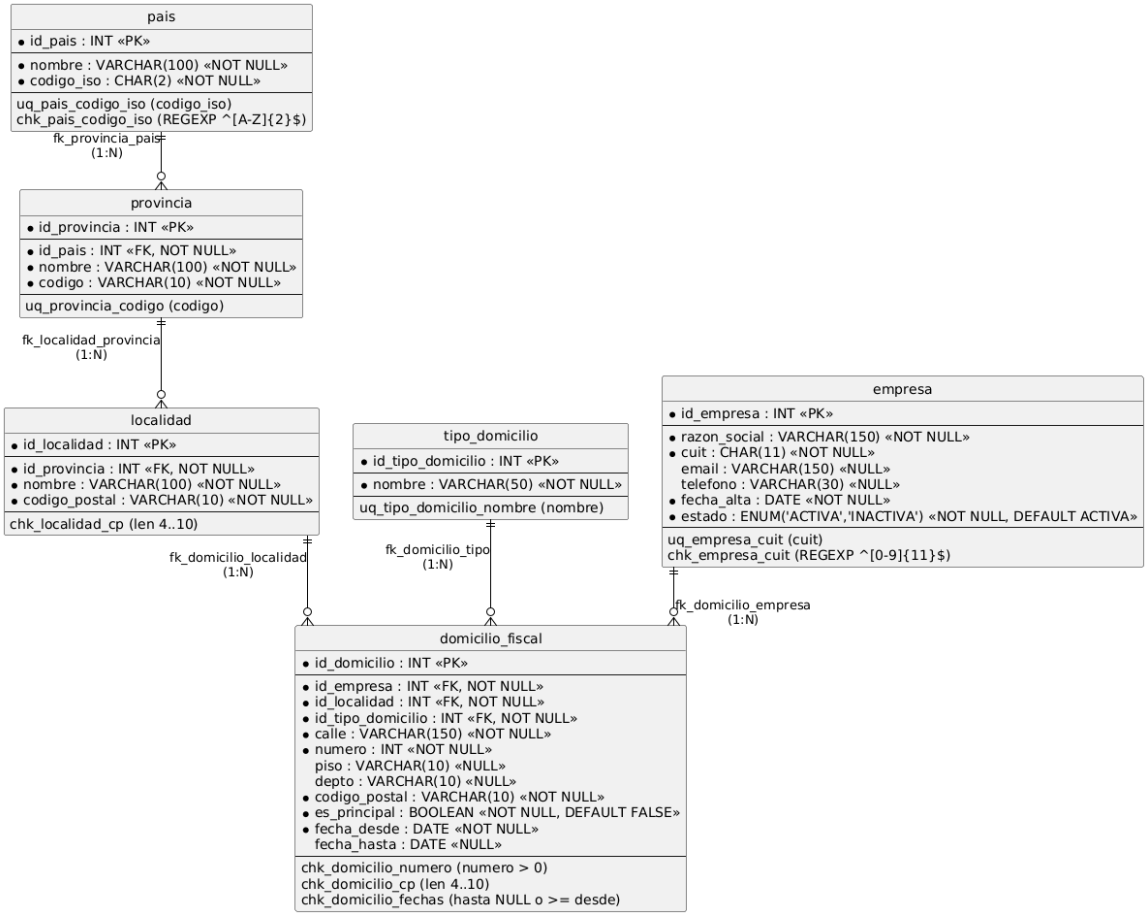
Alumno: Kenyi Meza

Comisión: 17

SGBD: MariaDB (InnoDB) – Entorno Linux

Entrega: Scripts SQL + Informe

Índice



1. Introducción

El objetivo de este TFI es consolidar los contenidos de la materia Bases de Datos I mediante el diseño e implementación de una base de datos orientada a la gestión de empresas y sus domicilios fiscales. El enfoque del trabajo es “primero integridad, luego volumen, luego consultas/optimización, y finalmente seguridad y concurrencia”.

El modelo está diseñado para mantener consistencia mediante constraints (PK, FK, UNIQUE, CHECK) y para permitir reportes multi-tabla (JOIN, GROUP BY, HAVING, subconsultas), con optimización mediante índices y análisis de planes de ejecución.

2. Entorno y herramientas

- Sistema operativo: Linux.
- Motor: MariaDB (tablas InnoDB).
- Scripts entregados: 01_esquema.sql a 09_concurrencia_guiada.sql.
- El trabajo se ejecuta sobre la base de datos: tfi_empresa_domicilio.

Nota: algunos editores SQL pueden marcar advertencias de sintaxis cuando se usan extensiones específicas del motor. La validación real se realiza con la ejecución en el servidor MariaDB.

3. Etapa 1 – Modelado y constraints

El esquema implementa un dominio con 6 entidades: pais, provincia, localidad, tipo_domicilio (catálogos) y empresa, domicilio_fiscal (tablas principales). Se aplican restricciones de dominio para asegurar calidad de datos antes de cualquier carga masiva.

3.1 Tablas principales y reglas de integridad

Tabla	Clave primaria	Constraints relevantes
pais	id_pais	UNIQUE(codigo_iso), CHECK formato ISO
provincia	id_provincia	FK a pais, UNIQUE(codigo)
localidad	id_localidad	FK a provincia, CHECK largo CP
tipo_domicilio	id_tipo_domicilio	UNIQUE(nombre)
empresa	id_empresa	UNIQUE(cuit), CHECK 11

		dígitos, ENUM estado
domicilio_fiscal	id_domicilio	FK a empresa/localidad/tipo, CHECK número>0, CHECK fechas

3.2 Pruebas de constraints (inserciones correctas/erróneas)

En el script 01_esquema.sql se incluyen inserciones de prueba y ejemplos de errores esperados: violación de UNIQUE en CUIT, violación de FK (empresa inexistente) y violaciones de CHECK (código postal y número). Estas pruebas demuestran que la base valida reglas críticas desde el nivel físico.

4. Etapa 2 – Catálogos y carga masiva

4.1 Catálogos (02_catalogos.sql)

Primero se cargan los catálogos mínimos (países, provincias, localidades y tipos de domicilio). Esto garantiza que, en la carga masiva posterior, todas las claves foráneas apunten a registros existentes.

4.2 Carga masiva (03_carga_masiva.sql)

Para simular un entorno real se generan aproximadamente 50.000 empresas y ~100.000 domicilios. Como el entorno de ejecución es MariaDB, la generación de secuencias se realiza con una tabla temporal (seq) y un procedimiento fill_seq(max_n), evitando dependencias de CTE recursivas específicas de MySQL 8.

- Integridad referencial: los domicilios se insertan siempre con FK válidas hacia empresa/localidad/tipo_domicilio.
- Coherencia de dominio: CUIT y email se generan con el número de secuencia para mantener unicidad; fechas y estado siguen reglas reproducibles.
- Cardinalidad: 1 domicilio por empresa + un segundo domicilio para ~50% de empresas.

4.3 Verificaciones de consistencia

Al final del script se ejecutan verificaciones: conteo de empresas y domicilios, chequeo de domicilios sin empresa (debe ser 0), y distribución de empresas por provincia.

5. Etapa 3 – Consultas, reportes y vistas

Las consultas del script 05_consultas.sql fueron diseñadas para ser útiles al dominio y para cumplir el requisito de SQL avanzado: JOINs múltiples, agregaciones con GROUP BY/HAVING y subconsultas.

5.1 Consultas incluidas (05_consultas.sql)

Consulta	Características SQL	Utilidad práctica
1) Empresa + domicilio principal	JOIN múltiple + filtro es_principal	Obtiene el domicilio principal y ubicación completa (localidad/provincia/país).
2) Empresas por provincia (HAVING)	GROUP BY + HAVING + ORDER BY	Reporte de concentración de empresas por provincia.
3) Empresas en provincias con alta densidad	Subconsulta con GROUP BY/HAVING + IN	Identifica provincias “críticas” y lista empresas asociadas.
4) Empresas activas recientes por provincia	JOIN + filtro por estado + rango de fechas	Reporte temporal de altas recientes por provincia.

5.2 Vistas (05_consultas.sql y 06_vistas.sql)

Se crean vistas para facilitar reporting y para exponer datos de manera controlada:

- vw_empresas_con_domicilio_principal: centraliza empresa + domicilio principal + ubicación completa.
- vw_empresas_por_provincia: resumen por provincia.
- vw_empresas_activas_recientes: empresas activas con fecha de alta dentro del último año.
- vw_empresas_cuit_enmascarado: CUIT parcialmente oculto para consumo por usuarios con permisos mínimos.

6. Etapa 3 – Índices y análisis con EXPLAIN

6.1 Índices (04_indices.sql)

Se crean índices para acelerar patrones típicos: igualdad por CUIT, JOIN por id_empresa, y reportes por localidad y código postal. El script es idempotente en MariaDB mediante CREATE INDEX IF NOT EXISTS.

6.2 EXPLAIN (05_explain.sql)

Para analizar performance se ejecutan EXPLAIN sobre tres categorías: igualdad (cuit), rango (fecha_alta) y JOIN multi-tabla. Esto permite observar el uso de índices, el tipo de acceso (ALL/range/ref), y la estimación de filas.

6.3 Tabla de medición (para completar con capturas)

Según la consigna, se ejecuta cada consulta 3 veces y se reporta la mediana de tiempos con y sin índice (o forzando el plan). Los tiempos dependen del equipo, por lo que se adjuntan capturas de ejecución en el anexo correspondiente.

Consulta	Sin índice (mediana, ms)	Con índice (mediana, ms)	Conclusión breve
Igualdad por CUIT	___	___	___
Rango por fecha_alta	___	___	___
JOIN empresa-domicilio-localidad-provincia	___	___	___

7. Etapa 4 – Seguridad e integridad

7.1 Usuario con mínimos privilegios (07_seguridad.sql)

Se crea un usuario de aplicación (app_user@localhost) con permisos mínimos. El usuario solo puede hacer SELECT sobre vistas específicas y SELECT/INSERT/UPDATE sobre domicilio_fiscal, evitando exposición directa de tablas sensibles y reduciendo la superficie de ataque.

7.2 Vistas para ocultar datos sensibles

La vista vw_empresas_cuit_enmascarado oculta parcialmente el CUIT (solo últimos 4 dígitos). Esto permite reportes sin exponer el identificador completo cuando no es necesario.

7.3 Pruebas de integridad

Las pruebas de constraints se incluyen en 01_esquema.sql con ejemplos de inserciones inválidas que disparan errores distintos (UNIQUE, FK y CHECK). Se adjuntan capturas de dichas ejecuciones como evidencia.

8. Etapa 5 – Transacciones

8.1 Procedimiento almacenado transaccional (08_transacciones.sql)

Se implementa el procedimiento almacenado sp_actualizar_domicilio_principal para asegurar atomicidad: 1) desmarca domicilios principales vigentes y 2) inserta el nuevo domicilio principal. Ambas operaciones se ejecutan dentro de START TRANSACTION/COMMIT y ante cualquier error se ejecuta ROLLBACK mediante un handler.

- Propiedad ACID destacada: atomicidad (no quedan estados intermedios).
- Consistencia: se mantiene la regla de “un domicilio principal vigente” por empresa según fecha.

9. Etapa 5 – Concurrencia (locks, timeouts y deadlocks)

Para concurrencia se utiliza un guion de pruebas manuales (09_concurrencia_guiada.sql) ejecutado en dos sesiones simultáneas. El motor InnoDB gestiona bloqueos a nivel de fila (row-level locking) y detecta deadlocks para evitar esperas indefinidas.

9.1 Lock wait timeout (1205)

Un escenario típico observado es el error 1205 (Lock wait timeout exceeded) cuando una sesión intenta modificar/insertar sobre datos relacionados a un registro bloqueado por otra sesión. Este comportamiento es esperado y demuestra protección de integridad.

9.2 Deadlock (1213) – recomendación de prueba

Para forzar deadlock (1213) se recomienda el patrón clásico de “orden cruzado” sobre dos filas distintas: Sesión A bloquea fila 1 y luego intenta fila 2; Sesión B bloquea fila 2 y luego intenta fila 1. El motor detecta el ciclo y aborta una transacción.

```
Ejemplo (ejecutar en dos sesiones casi en paralelo):
Sesión A: START TRANSACTION; UPDATE empresa SET razon_social='A1' WHERE
id_empresa=1; SELECT SLEEP(3); UPDATE empresa SET razon_social='A2' WHERE
id_empresa=2;
Sesión B: START TRANSACTION; UPDATE empresa SET razon_social='B2' WHERE
id_empresa=2; SELECT SLEEP(3); UPDATE empresa SET razon_social='B1' WHERE
id_empresa=1;
```

9.3 Niveles de aislamiento

El script incluye guía para comparar READ COMMITTED vs REPEATABLE READ, observando diferencias en lecturas dentro de una transacción mientras otra sesión inserta datos. Se adjuntan evidencias (capturas/resultado) según consigna.

Nota práctica: en algunos entornos gráficos, las consultas bloqueadas pueden cortar la conexión por timeouts del cliente. Para pruebas confiables se recomienda usar dos consolas/terminales o ajustar los timeouts del cliente.

10. Orden de ejecución de scripts

Orden	Script	Propósito
1	01_esquema.sql	Crea la base y las tablas con constraints + pruebas de integridad.
2	02_catalogos.sql	Carga catálogos/maestras.
3	03_carga_masiva.sql	Genera volumen (empresas y domicilios) + verificaciones.
4	04_indices.sql	Crea índices para performance y verifica.
5	05_consultas.sql	Consultas avanzadas + vista principal.
6	05_explain.sql	EXPLAIN para análisis de planes.
7	06_vistas.sql	Vistas adicionales para reportes y seguridad.
8	07_seguridad.sql	Usuario de aplicación + privilegios mínimos.
9	08_transacciones.sql	Procedimiento almacenado transaccional.
10	09_concurrencia_guiada.sql	Guion para pruebas en 2 sesiones (locks/aislamiento/deadlock).

11. Uso responsable de IA (anexo)

Durante el desarrollo utilicé IA generativa como herramienta de apoyo pedagógico para validar decisiones de modelado, documentar el DER en

PlantUML, revisar constraints, y mejorar la explicación de transacciones y concurrencia. El uso de IA se mantuvo como tutoría: las decisiones finales se validaron ejecutando los scripts y contrastando con resultados reales.

Según la consigna, se adjunta el anexo con evidencias de las interacciones con IA (chat completo/organizado por etapa).

12. Conclusiones

El trabajo implementa un modelo normalizado con integridad fuerte, generación de datos masivos coherentes, consultas útiles para el dominio y elementos clave de un sistema real: índices, seguridad por mínimos privilegios, vistas, transacciones y pruebas de concurrencia. De esta forma, el proyecto demuestra no solo sintaxis SQL, sino también criterios de diseño, robustez y operación en un contexto multiusuario.