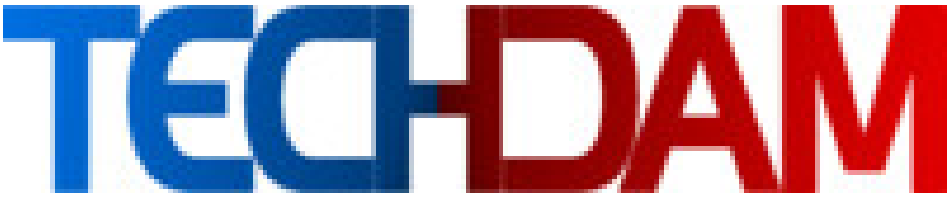
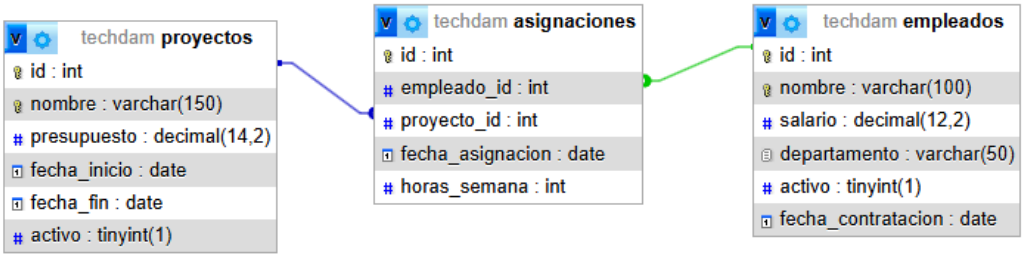
PROYECTO TechDAM - JDBC AVANZADO



Nombre del alumno: Verónica Tello Nozal
Fecha: 16:11:2025

1. DIAGRAMA DE BASE DE DATOS

Diagrama ER con las 3 Tablas



Indicar claves primarias y foraneas

CLAVES PRIMARIAS

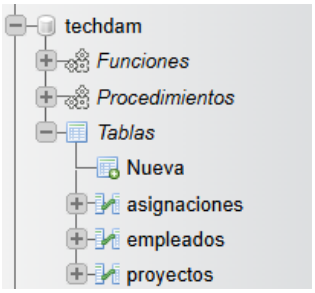
- Tabla empleados
    - id (INT AUTO_INCREMENT) - Clave primaria
- Tabla proyectos
    - id (INT AUTO_INCREMENT) - Clave primaria
- Tabla asignaciones
    - id (INT AUTO_INCREMENT) - Clave primaria

CLAVES FORÁNEAS

- Tabla asignaciones
    - empleado_id → Referencia a empleados(id)
    - proyecto_id → Referencia a proyectos(id)

2. CAPTURAS DE PANTALLA

Tablas creadas en MySQL Workbench / HeidiSQL



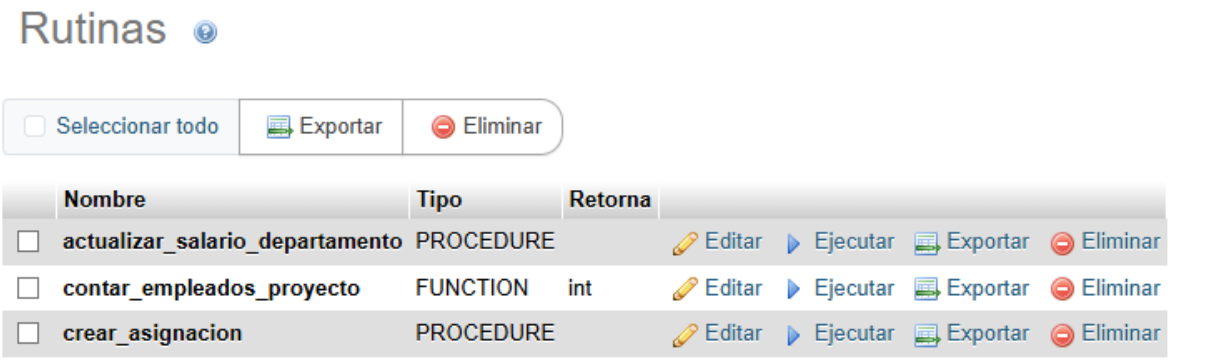
Ejecución exitosa de CRUD de empleados

```
Empleado creado con id: 6
Todos empleados:
Empleado{id=1, nombre='Ana López', salario=1500.00, departamento='Finanzas', activo=true, fechaContratacion=2019-06-10}
Empleado{id=2, nombre='Luis García', salario=2083.73, departamento='Desarrollo', activo=true, fechaContratacion=2018-03-20}
Empleado{id=3, nombre='Marta Ruiz', salario=2315.25, departamento='Desarrollo', activo=true, fechaContratacion=2020-01-15}
Empleado{id=4, nombre='Pedro Gómez', salario=1750.00, departamento='Ventas', activo=true, fechaContratacion=2017-11-01}
Empleado{id=5, nombre='Laura Fernández', salario=2200.00, departamento='Proyectos', activo=true, fechaContratacion=2021-09-01}
Empleado{id=6, nombre='Prueba Usuario', salario=1300.00, departamento='QA', activo=true, fechaContratacion=2025-11-16}
```

Ejecución exitosa de CRUD de proyectos

```
Proyecto creado id: 6
Transferencia completada: true
Empleados actualizados en Desarrollo: 2
Empleados no encontrados: [9999]
Asignaciones batch completadas: false
Empleados distintos en proyecto 1: 2
```

Invocación de procedimiento almacenado



Transacción con COMMIT

```
Transferencia completada: true
```

Mostrando filas 0 - 1 (total de 2, La consulta tardó 0.0003 segundos.)

```
SELECT presupuesto FROM proyectos WHERE id IN (1,2);
```

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Ac

Mostrar todo | Número de filas: 25 Filtrar filas: Buscar en e

Opciones extra

presupuesto
46000.00
34000.00

despues de ejecutarlo:

Mostrando filas 0 - 1 (total de 2, La consulta tardó 0.0002 segundos.)

```
SELECT presupuesto FROM proyectos WHERE id IN (1,2);
```

Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP

Mostrar todo | Número de filas: 25 Filtrar filas: Busc

Opciones extra

presupuesto
45000.00
35000.00

Transacción con ROLLBACK

Esto pasaria si con el presupuesto no se puede hacer una transferencia.

```
// Transacción: transferir presupuesto (Opción A)
boolean transfer = txService.transferirPresupuesto( proyectoOrigenId: 1, proyectoDestinoId: 2, new BigDecimal( val: "9999999.00"));
System.out.println("Transferencia completada: " + transfer);
```

```
Transferencia completada: false
```

Pool de conexiones configurado (logs o métricas)

```
[main] INFO com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Starting...
[main] INFO com.zaxxer.hikari.pool.HikariPool - HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@101df177
[main] INFO com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Start completed.
Empleado creado con id: 6
```

```
[main] INFO com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
[main] INFO com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown completed.
```

---

3. EXPLICACIÓN DE DECISIONES TÉCNICAS

1. ¿Por qué usar PreparedStatement en lugar de Statement?

Evidencia en código: Buscar en EmpleadoDAO.java y ProyectoDAO.java todos los usos de PreparedStatement

Respuesta:

- Prevención de SQL Injection: Los parámetros se enlazan de forma segura
- Mejor rendimiento: Reutilización de planes de ejecución
- Manejo seguro de tipos: Conversión automática de datos
- Código más limpio: Manejo simplificado de comillas y caracteres especiales

2. ¿Qué ventajas tiene el pool de conexiones?

Evidencia en código: DatabaseConnection.java - configuración HikariCP

Respuesta:

- Reutilización de conexiones: Reduce overhead de crear/cerrar conexiones
- Mejor performance: Conexiones listas para usar
- Control de recursos: Límite máximo de conexiones activas
- Manejo de timeouts: Previene conexiones huérfanas

3. ¿En qué casos usarías procedimientos almacenados?

Evidencia en código: Procedimiento en DatabaseSchema.sql

Respuesta:

- Lógica compleja de BD: Operaciones que requieren múltiples queries
- Seguridad: Control granular de permisos
- Performance: Ejecución en el servidor de BD
- Mantenimiento: Lógica centralizada en la base de datos

4. ¿Por qué es importante el control de transacciones?

Evidencia en código: Bloques con commit() y rollback()

Respuesta:

- Consistencia de datos: Garantiza que operaciones múltiples se completen todas o ninguna
- Integridad referencial: Mantiene las relaciones entre tablas consistentes
- Recuperación de errores: Permite revertir cambios en caso de fallos
- Operaciones atómicas: Trata múltiples operaciones como una sola unidad

---

4. PROBLEMAS ENCONTRADOS Y SOLUCIONES

Problema 1: Configuración del Pool de Conexiones

Evidencia: Buscar en DatabaseConnection.java o README.md

Descripción:

- Dificultad para configurar parámetros óptimos del pool HikariCP
- Errores de timeout en conexiones bajo carga

Solución:

- Ajuste de connectionTimeout y maxLifetime
- Configuración de minimumIdle y maximumPoolSize según carga esperada
- Implementación de reintentos automáticos

Problema 2: Manejo de Transacciones Distribuidas

Evidencia: Buscar en métodos que usan múltiples operaciones

Descripción:

- Inconsistencias al actualizar múltiples tablas simultáneamente
- Dificultad para mantener atomicidad en operaciones complejas

Solución:

- Implementación de patrón setAutoCommit(false) explícito
- Manejo estructurado de try-catch con rollback garantizado
- Validación de estado de transacción antes de commit

---

5. GitHUB

https://github.com/nika32231/TelloNozal_Veronica_techdam.git
