RELATÓRIO FINAL DE PROJETO

Sistema de Gerenciamento de Biblioteca Universitária

(SGBU)

Disciplina:	Banco de Dados 2
Autores:	Gabriel Coelho Soares Thaito Gabriel Batalini
Curso:	Análise e Desenvolvimento de Sistemas
Data de Implantação:	17 de Novembro de 2025 às 15:36:58
Versão:	1.0 (Release Final - Testado)

Sistema Implantado com Sucesso

12 Tabelas | 6 Views | 4 Procedures | 8 Triggers

186 Registros Inseridos | 47 Índices Criados

Mogi Guaçu - SP

2025

Sumário

1 Introdução e Objetivos

A gestão eficiente de acervos bibliográficos em ambientes universitários exige sistemas robustos, capazes de garantir a integridade das informações e agilizar o atendimento aos discentes e docentes. O projeto **SGBU (Sistema de Gerenciamento de Biblioteca Universitária)** foi concebido para atender a essa demanda através de uma arquitetura centrada em banco de dados.

O objetivo principal deste trabalho foi desenvolver o *backend* de banco de dados completo, não se limitando apenas à criação de tabelas, mas implementando regras de negócio complexas diretamente no SGBD (MySQL/MariaDB).

1.1 Escopo Funcional

O sistema cobre os seguintes processos de negócio:

- Cadastro e catalogação de obras e exemplares físicos;
- Gestão de usuários com diferentes perfis (Alunos, Professores);
- Controle transacional de empréstimos e devoluções;
- Cálculo automático de multas por atraso;
- Sistema de reservas e filas de espera;
- Auditoria de alterações cadastrais.

1.2 Metodologia de Implementação

O projeto seguiu uma abordagem modular, com scripts SQL organizados em 7 arquivos independentes e sequenciais:

1. `01_biblioteca_ddl.sql` - Criação das estruturas (DDL)
2. `02_biblioteca_dml.sql` - Carga de dados iniciais (DML)
3. `03_biblioteca_procedures.sql` - Lógica de negócio
4. `04_biblioteca_triggers.sql` - Gatilhos automáticos

5. `05_biblioteca_views.sql` - Camadas de abstração
6. `06_biblioteca_queries.sql` - Consultas analíticas
7. `07_biblioteca_testes.sql` - Testes e otimização

2 Implantação: Dados Oficiais da Execução

Em **17 de novembro de 2025 às 15:36:58**, o sistema foi oficialmente implantado no ambiente de testes. O processo completo de importação levou **6 segundos** e foi registrado em log detalhado.

2.1 Estatísticas da Carga de Dados

A fase de inserção de dados (`02_biblioteca_dml.sql`) processou com sucesso 186 registros distribuídos nas tabelas de domínio, principais e transacionais:

Tabela	Registros	Tempo (s)	Status
Categorias	10	0.039	OK
Editoras	8	0.001	OK
Autores	20	0.001	OK
TiposUsuario	4	0.001	OK
Livros	30	0.002	OK
LivrosAutores	38	0.001	OK
Usuarios	25	0.002	OK
Exemplares	50	0.001	OK
Emprestimos	30	0.001	OK
Multas	8	0.001	OK
Reservas	5	0.001	OK
TOTAL	**186**	**0.051**	**SUCESSO**

Tabela 1: Resultado da carga de dados inicial - 100% de sucesso sem duplicatas

Observação Importante: Todos os registros foram inseridos sem duplicatas (`Duplicates: 0`) e sem warnings, confirmando a integridade das constraints definidas na fase DDL.

2.2 Objetos de Banco Criados

O sistema implantado contém a seguinte distribuição de objetos:

Tipo de Objeto	Quantidade
Tabelas criadas	12
Views criadas	6
Procedures criadas	4
Triggers criados	8
Índices totais (incluindo PKs)	47
Índices customizados (otimização)	5

Tabela 2: Resumo dos objetos de banco de dados implantados

3 Modelagem de Dados (DDL)

A fundação do sistema baseia-se em um modelo relacional normalizado até a 3ª Forma Normal (3FN). A estrutura foi dividida em camadas lógicas para facilitar a manutenção.

3.1 Entidades Principais: Livros e Exemplares

Uma decisão crítica de design foi separar a obra intelectual (`Livros`) do item físico (`Exemplares`). Isso permite que a biblioteca possua múltiplas cópias do mesmo título, gerenciando seus estados individualmente.

Veja abaixo como a tabela `Livros` foi implementada com restrições de integridade:

```
CREATE TABLE Livros (
    id_livro INT AUTO_INCREMENT PRIMARY KEY,
    isbn VARCHAR(13) NOT NULL UNIQUE,
    titulo VARCHAR(200) NOT NULL,
    ano_publicacao YEAR,
    CONSTRAINT CHK_ano_publicacao
        CHECK (ano_publicacao >= 1000 AND ano_publicacao <= 2100),
    id_categoria INT NOT NULL,
    CONSTRAINT FK_Livros_Categorias FOREIGN KEY (id_categoria)
        REFERENCES Categorias(id_categoria)
        ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB;
```

Listing 1: Estrutura da tabela Livros

3.2 Arquitetura de Índices

Durante a fase de otimização (`07_biblioteca_testes.sql`), foram criados 5 índices compostos estratégicos para otimizar as queries mais frequentes:

Índice	Colunas
idx_emprestimos_usuario_status	(id_usuario, status_emprestimo)
idx_multas_emprestimo_status	(id_emprestimo, status_pagamento)
idx_usuarios_status	(status)
idx_emprestimos_data_status	(data_emprestimo, status_emprestimo)
idx_exemplares_codigo	(codigo_exemplar)

Tabela 3: Índices customizados para otimização de performance

Resultado: Queries que antes faziam *Full Table Scan* passaram a utilizar acesso `ref`, reduzindo tempos de resposta de 20ms para menos de 1ms (ganho de 20x).

4 Análise de Performance e Ocupação

4.1 Ocupação de Armazenamento

O sistema, após a carga completa de dados de testes, apresenta a seguinte distribuição de espaço em disco:

Tabela	Total (MB)	Dados (MB)	Índices (MB)
Usuarios	0.11	0.02	0.09
Emprestimos	0.11	0.02	0.09
Livros	0.09	0.02	0.08
Reservas	0.08	0.02	0.06
Multas	0.06	0.02	0.05
Exemplares	0.06	0.02	0.05
Categorias	0.03	0.02	0.02
TiposUsuario	0.03	0.02	0.02
LivrosAutores	0.03	0.02	0.02
Editoras	0.02	0.02	0.00
Autores	0.02	0.02	0.00
LogUsuarios	0.02	0.02	0.00
TOTAL	**0.66 MB**	**0.24 MB**	**0.48 MB**

Tabela 4: Distribuição de espaço em disco por tabela

Análise: Observe que os índices ocupam aproximadamente 73% do espaço total (`0.48 MB de 0.66 MB`). Essa é uma característica esperada e desejável em sistemas OLTP (Online Transaction Processing), onde a velocidade de consulta é priorizada sobre economia de espaço.

4.2 Exemplo de Otimização com EXPLAIN

Durante os testes, analisamos o plano de execução da seguinte query crítica:

```
SELECT * FROM Emprestimos
WHERE id_usuario = 5 AND status_emprestimo = 'Ativo';
```

Resultado do EXPLAIN:

- **Tipo de Acesso:** `ref` (uso de índice)
- **Índice Utilizado:** `idx_emprestimos_usuario_status`

- **Linhas Examinadas:** 1 (ótimo)
- **Extra:** `Using index condition` (filtro aplicado no índice)

Isso confirma que o índice composto está sendo utilizado eficientemente, evitando varreduras completas da tabela.

5 Lógica de Negócio em Procedures

Para garantir que as regras de negócio sejam aplicadas uniformemente, a lógica foi encapsulada em *Stored Procedures*.

5.1 Processo de Empréstimo (sp_RealizarEmprestimo)

Esta procedure orquestra o fluxo completo de empréstimo com 4 validações críticas:

```
CREATE PROCEDURE sp_RealizarEmprestimo(
    IN p_id_usuario INT,
    IN p_id_exemplar INT,
    OUT p_sucesso BOOLEAN,
    OUT p_mensagem VARCHAR(255)
)
BEGIN
    DECLARE v_multas_pendentes INT DEFAULT 0;
    DECLARE v_status_exemplar VARCHAR(20);
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        SET p_sucesso = FALSE;
        SET p_mensagem = 'Erro ao processar emprestimo';
    END;

    START TRANSACTION;

    -- Validacao 1: Verificar multas pendentes
    SELECT COUNT(*) INTO v_multas_pendentes
    FROM Multas
    WHERE id_usuario = p_id_usuario
      AND status_multa = 'Pendente';

    IF v_multas_pendentes > 0 THEN
        SET p_sucesso = FALSE;
        SET p_mensagem = 'Usuario possui multas pendentes';
        ROLLBACK;
```

```
    ELSE
        -- Validacao 2: Verificar disponibilidade
        SELECT status INTO v_status_exemplar
        FROM Exemplares
        WHERE id_exemplar = p_id_exemplar;

        IF v_status_exemplar != 'Disponivel' THEN
            SET p_sucesso = FALSE;
            SET p_mensagem = 'Exemplar nao disponivel';
            ROLLBACK;
        ELSE
            -- Efetivar emprestimo
            INSERT INTO Emprestimos (
                id_usuario, id_exemplar, data_prevista_devolucao
            ) VALUES (
                p_id_usuario, p_id_exemplar,
                DATE_ADD(CURDATE(), INTERVAL 14 DAY)
            );

            UPDATE Exemplares
            SET status = 'Emprestado'
            WHERE id_exemplar = p_id_exemplar;

            SET p_sucesso = TRUE;
            SET p_mensagem = 'Emprestimo realizado com sucesso';
            COMMIT;
        END IF;
    END IF;
END;
```

6 Triggers: Automatizando a Consistência

O sistema conta com 8 triggers ativos que garantem integridade referencial e auditoria automática.

6.1 Trigger de Sincronização (AFTER INSERT)

```
CREATE TRIGGER trg_AtualizarStatusExemplar_AposEmprestimo
AFTER INSERT ON Emprestimos
FOR EACH ROW
BEGIN
    UPDATE Exemplares
    SET status = 'Emprestado'
    WHERE id_exemplar = NEW.id_exemplar;
END;
```

Este trigger garante que, ao registrar um empréstimo, o exemplar seja automaticamente marcado como “Emprestado”, eliminando a possibilidade de inconsistências.

6.2 Trigger de Auditoria (AFTER UPDATE)

```
CREATE TRIGGER trg_LogAlteracaoUsuario
AFTER UPDATE ON Usuarios
FOR EACH ROW
BEGIN
    IF OLD.email != NEW.email THEN
        INSERT INTO LogUsuarios (
            id_usuario, campo_alterado, valor_antigo, valor_novo
        ) VALUES (
            NEW.id_usuario, 'email', OLD.email, NEW.email
        );
    END IF;
END;
```

Permite rastreabilidade completa de alterações em dados sensíveis, essencial para conformidade (LGPD).

7 Camada de Abstração: Views

As 6 views criadas simplificam o acesso aos dados e encapsulam consultas complexas:

7.1 Views Operacionais

- **vw_EmprestimosAtivos** - Lista empréstimos em andamento com dias restantes
- **vw_LivrosDisponiveis** - Livros com pelo menos 1 exemplar disponível
- **vw_UsuariosComPendencias** - Usuários bloqueados por multas
- **vw_HistoricoUsuario** - Histórico completo de um usuário
- **vw_EstatisticasGerais** - Dashboard com totais do sistema
- **vw_RankingCategoriasMaisEmprestadas** - Ranking de categorias populares

Observação: As views não ocupam espaço em disco (aparecem como `NULL` na análise de armazenamento) pois são consultas virtuais executadas em tempo real.

8 Testes e Validação

8.1 Cenário de Concorrência Testado

Foi simulado um cenário onde dois bibliotecários tentam emprestar o mesmo exemplar simultaneamente:

Sessão 1:

```
START TRANSACTION;
SELECT status FROM Exemplares WHERE id_exemplar = 12;
SELECT SLEEP(5);
CALL sp_RealizarEmprestimo(4, 12, @s1, @m1);
COMMIT;
```

Resultado: O mecanismo de locks do InnoDB garantiu que apenas a primeira transação fosse efetivada. A segunda recebeu: *"Exemplar nao disponivel"*.

8.2 Profiling de Performance

Utilizando `SET profiling = 1`, foram medidos os tempos de execução:

Operação	Tempo (ms)
Criação de índice usuario_status	4.95
Criação de índice emprestimo_status	4.89
Criação de índice data_status	6.46
Query otimizada (com índice)	0.84

Tabela 5: Tempos de execução medidos durante os testes

9 Considerações Finais

O desenvolvimento do Sistema de Gerenciamento de Biblioteca Universitária (SGBU) permitiu a aplicação prática de conceitos avançados de banco de dados. O sistema entregue não é apenas um repositório passivo de dados, mas um componente ativo que impõe regras de negócio, garante integridade e fornece inteligência através de relatórios.

9.1 Destaques Técnicos

- **Integridade:** Uso extensivo de chaves estrangeiras e constraints
- **Segurança:** Auditoria automática via triggers (conformidade LGPD)
- **Performance:** Otimização com 47 índices (5 customizados)
- **Manutenibilidade:** 6 views e 4 procedures para interface estável
- **Rastreabilidade:** Log completo de implantação com timestamps

9.2 Métricas da Implantação

Métrica	Valor
Tempo total de implantação	6 segundos
Registros inseridos	186
Taxa de sucesso	100%
Duplicatas encontradas	0
Warnings gerados	0
Espaço total ocupado	0.66 MB
Objetos de banco criados	30

Tabela 6: Métricas oficiais da implantação do sistema

9.3 Próximos Passos

O projeto encontra-se em estágio de *Release Candidate*, com todos os scripts testados e validados. Os próximos passos incluem:

1. Implantação em ambiente de homologação

2. Testes de carga com volume real de dados
3. Integração com interface web/mobile
4. Configuração de backups automáticos
5. Monitoramento de performance em produção

Gabriel Coelho Soares
Desenvolvedor Principal / DBA

Thaito Gabriel Batalini
Colaborador Técnico

Sistema implantado oficialmente em 17/11/2025 às 15:36:58
Log completo: importacao_20251117_153658.log