

Projeto B2 Banco de Dados II

Participantes:

Rafael Christian Silva Wernesbach

Rafael Barcelos Azevedo

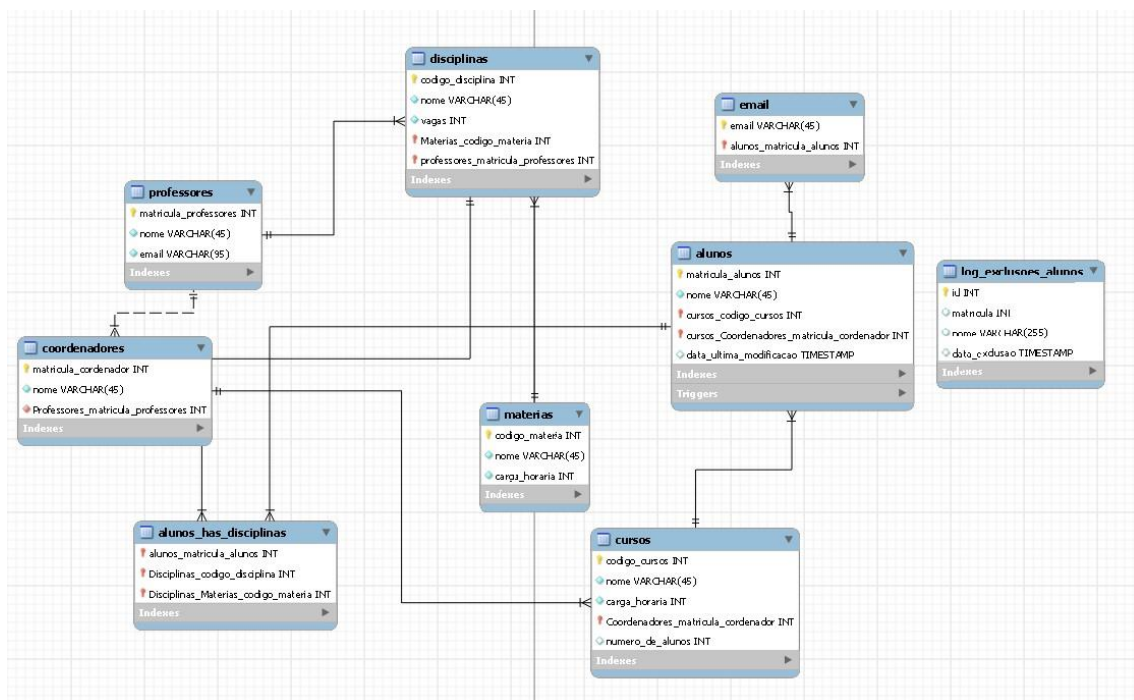
Pedro Henrique Souza Cravo

Pedro Henrique Pimentel da Silva

Cauã Agrisi Merizio

SCRIPT DO BD

Recriamos o banco de dados do minimundo, baseado no projeto do primeiro bimestre, realizamos algumas alterações como mudanças nas relações, cardinalidades, criação de novas tabelas e remoção de outras.

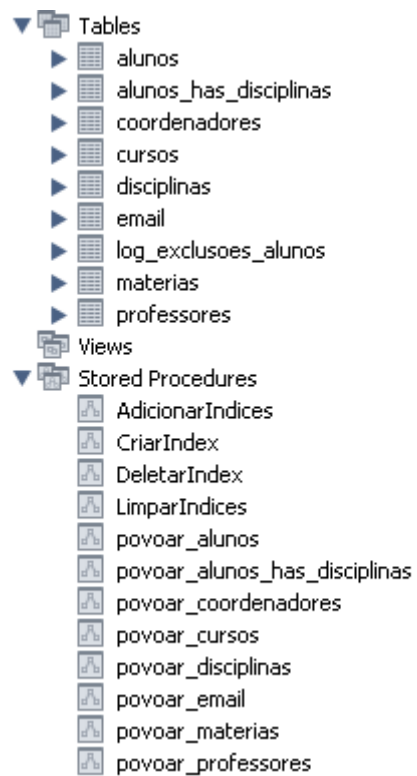


Script:

https://drive.google.com/drive/folders/1Dbm4yRFanSpi0Y0740Yh7WcInLLHuSy_?usp=sharing

Procedures de povoamento e outros recursos

Foram Criados Diversos procedures para povoar cada tabela presente, além de procedures para a criação dos índices e remoção dos mesmos quando necessário.



```

-----
-- Procedure Povoar Coordenadores
-----

-- Pedro Cravo

DROP PROCEDURE IF EXISTS povoar_coordenadores;

DELIMITER $$
CREATE PROCEDURE povoar_coordenadores(IN num_coord INT)
BEGIN
    DECLARE Id INT;
    SET Id = 1;
    WHILE Id <= num_coord DO
        INSERT INTO coordenadores(matricula_cordenador, nome, Professores_matricula_professores) VALUES (
            Id + 1000,
            CONCAT('Coordenador - ', CAST(Id AS CHAR)),
            (SELECT matricula_professores FROM professores ORDER BY RAND() LIMIT 1)
        );
        SET Id = Id + 1;
    END WHILE;
END $$
DELIMITER ;

-- Pedro Pimentel

-----
-- Procedure Povoar Cursos
-----

DROP PROCEDURE IF EXISTS povoar_cursos;

DELIMITER $$
CREATE PROCEDURE povoar_cursos(IN num_cursos INT)
BEGIN
    DECLARE Id INT;
    SET Id = 1;
    WHILE Id <= num_cursos DO
        INSERT INTO cursos(codigo_cursos, nome, Coordenadores_matricula_cordenador) VALUES (
            CONCAT(Id + 100),
            CONCAT('Curso - ', CAST(Id AS CHAR)),
            (SELECT matricula_cordenador FROM coordenadores ORDER BY RAND() LIMIT 1)
        );
        SET Id = Id + 1;
    END WHILE;
END $$
DELIMITER ;

```

```
-- Pedro Cravo
```

```
-- -----  
-- Procedure Povoar Professores  
-- -----
```

```
DROP PROCEDURE IF EXISTS povoarprofessores;
```

```
DELIMITER $$
```

```
CREATE PROCEDURE povoar_professores(IN num_prof INT)
```

```
BEGIN
```

```
    DECLARE Id INT;
```

```
    SET Id = 1;
```

```
    WHILE Id <= num_prof DO
```

```
        INSERT INTO professores(matricula_professores, nome, email) VALUES (  
            Id + 1000,  
            CONCAT('Professor - ', CAST(Id AS CHAR)),  
            CONCAT('professor_UW', CAST(Id AS CHAR), '@gmail.com.br')  
        );
```

```
        SET Id = Id + 1;
```

```
    END WHILE;
```

```
END $$
```

```
DELIMITER ;
```

-- Pedro Pimentel

-- Procedure Povoar Alunos

DROP PROCEDURE IF EXISTS povoar_alunos;

DELIMITER \$\$

CREATE PROCEDURE povoar_alunos(IN num_alunos INT)

BEGIN

DECLARE Id INT;

DECLARE idcurso INT;

DECLARE matriculacoord INT;

SET Id = 1;

WHILE Id <= num_alunos DO

SELECT codigo_cursos, Coordenadores_matricula_cordenador INTO idcurso, matriculacoord

FROM cursos

ORDER BY RAND()

LIMIT 1;

INSERT INTO alunos(matricula_alunos, nome, Cursos_codigo_cursos, cursos_Coordenadores_matricula_cordenador) VALUES (

Id + 1000,

CONCAT('Aluno - ', CAST(Id AS CHAR)),

idcurso,

matriculacoord

);

SET Id = Id + 1;

END WHILE;

END \$\$

DELIMITER ;

-- Pedro pimentel

-- Procedure Povoar Materias

DROP PROCEDURE IF EXISTS povoar_materias;

DELIMITER \$\$

CREATE PROCEDURE povoar_materias(IN num_mat INT)

BEGIN

DECLARE Id INT;

SET Id = 1;

WHILE Id <= num_mat DO

INSERT INTO materias(codigo_materia, nome, carga_horaria) VALUES (

Id + 100,

CONCAT('Materias - ', CAST(Id AS CHAR)),

40 + (RAND() * 21) -- OU FLOOR(40 + (RAND() * (60 - 40 + 1)))

);

SET Id = Id + 1;

END WHILE;

END \$\$

DELIMITER ;

-- Rafael Christian Silva Wernesbach

-- Procedure Povoar Disciplinas

DROP PROCEDURE IF EXISTS povoar_disciplinas;

DELIMITER \$\$

CREATE PROCEDURE povoar_disciplinas(IN num_dis INT)

BEGIN

DECLARE Id INT;

SET Id = 1;

WHILE Id <= num_dis DO

INSERT INTO disciplinas(codigo_disciplina, nome, vagas, Materias_codigo_materia, Professores_matricula_professores) VALUES (
Id + 100,
CONCAT('Disciplinas - ', CAST(Id AS CHAR)),
20 + (RAND() * 41), -- OU FLOOR(20 + (RAND() * (40 - 20 + 1)))
(SELECT codigo_materia FROM materias ORDER BY RAND() LIMIT 1),
(SELECT matricula_professores FROM professores ORDER BY RAND() LIMIT 1)
);

SET Id = Id + 1;

END WHILE;

END \$\$

DELIMITER ;

-- Pedro Cravo

-- Procedure Povoar Emails

DROP PROCEDURE IF EXISTS povoar_email;

DELIMITER \$\$

CREATE PROCEDURE povoar_email(IN num_email INT)

BEGIN

DECLARE Id INT;

SET Id = 1;

WHILE Id <= num_email DO

INSERT INTO email (email, alunos_matricula_alunos)

VALUES(
CONCAT('aluno_UV', CAST(Id AS CHAR), '@gmail.com.br'),
(SELECT matricula_alunos FROM alunos ORDER BY RAND() LIMIT 1)
);

SET Id = Id + 1;

END WHILE;

END \$\$

DELIMITER ;

```

DROP PROCEDURE IF EXISTS povoar_alunos_has_disciplinas;

-- Rafael Christian Silva Wernesbach

-- Procedure Povoar alunos_has_disciplinas
DELIMITER $$

CREATE PROCEDURE povoar_alunos_has_disciplinas(IN num_registros INT)
BEGIN
    DECLARE Id INT;
    DECLARE aluno_id INT;
    DECLARE disciplina_id INT;
    DECLARE materia_id INT;
    DECLARE exists_count INT;
    SET Id = 1;

    WHILE Id <= num_registros DO
        -- Seleciona um aluno de forma aleatória
        SELECT matricula_alunos INTO aluno_id
        FROM alunos
        ORDER BY RAND()
        LIMIT 1;

        -- Seleciona uma disciplina e sua matéria correspondente de forma aleatória
        SELECT codigo_disciplina, Materias_codigo_materia INTO disciplina_id, materia_id
        FROM disciplinas
        ORDER BY RAND()
        LIMIT 1;

        -- Verifica se a combinação já existe
        SELECT COUNT(*) INTO exists_count
        FROM alunos_has_disciplinas
        WHERE alunos_matricula_alunos = aluno_id
            AND disciplinas_codigo_disciplina = disciplina_id
            AND disciplinas_Materias_codigo_materia = materia_id;

        -- Insere a combinação apenas se ela não existir
        IF exists_count = 0 THEN
            INSERT INTO alunos_has_disciplinas (
                alunos_matricula_alunos,
                disciplinas_codigo_disciplina,
                disciplinas_Materias_codigo_materia
            )
            VALUES (
                aluno_id,
                disciplina_id,
                materia_id
            );

            SET Id = Id + 1;
        END IF;
    END WHILE;
END $$

```

```

-- Rafael Christian Silva Wernesbach
DROP PROCEDURE IF EXISTS CriarIndex;

DELIMITER //

CREATE PROCEDURE CriarIndex(p_tableName VARCHAR(255), p_indexName VARCHAR(255), p_indexColumns VARCHAR(255))
BEGIN
    DECLARE indexExists INT;
    SELECT COUNT(*)
    INTO indexExists
    FROM INFORMATION_SCHEMA.STATISTICS
    WHERE TABLE_NAME = p_tableName
    AND INDEX_NAME = p_indexName
    AND TABLE_SCHEMA = 'minimundouv';

    IF indexExists = 0 THEN
        SET @createIndexQuery = CONCAT('CREATE INDEX ', p_indexName, ' ON ', p_tableName, ' (', p_indexColumns, ')');
        PREPARE createIndexStatement FROM @createIndexQuery;
        EXECUTE createIndexStatement;
        DEALLOCATE PREPARE createIndexStatement;
    END IF;
END;
//

DELIMITER ;

-- Rafael Christian Silva Wernesbach
DROP PROCEDURE IF EXISTS AdicionarIndices;

DELIMITER //

CREATE PROCEDURE AdicionarIndices()
BEGIN
    CALL CriarIndex('coordenadores', 'idx_professores_matricula_professores', 'Professores_matricula_professores');
    CALL CriarIndex('cursos', 'idx_coordenadores_matricula_cordenador', 'Coordenadores_matricula_cordenador');
    CALL CriarIndex('alunos', 'idx_cursos_codigo_cursos', 'cursos_codigo_cursos, cursos_Coordenadores_matricula_cordenador');
    CALL CriarIndex('disciplinas', 'idx_materias_codigo_materia', 'Materias_codigo_materia');
    CALL CriarIndex('disciplinas', 'idx_professores_matricula_professores_disciplinas', 'Professores_matricula_professores');
    CALL CriarIndex('alunos_has_disciplinas', 'idx_alunos_matricula_alunos', 'alunos_matricula_alunos');
    CALL CriarIndex('alunos_has_disciplinas', 'idx_disciplinas_codigo_disciplina', 'Disciplinas_codigo_disciplina, Disciplinas_Materias_codigo_materia');
    CALL CriarIndex('alunos_has_disciplinas', 'idx_composto_alunos_disciplinas', 'alunos_matricula_alunos, Disciplinas_codigo_disciplina, Disciplinas_Materias_codigo_materia');
    CALL CriarIndex('email', 'idx_alunos_matricula_alunos_email', 'alunos_matricula_alunos');
    CALL CriarIndex('professores', 'idx_unico_email_professores', 'email');
END;
//

DELIMITER ;

```

Limpeza das tabelas + chamada dos procedures:


```

-- Rafael Christian Silva Wernesbach

-- Limpando os dados das tabelas

set foreign_key_checks=0;
TRUNCATE TABLE minimundouvv.alunos_has_disciplinas;
TRUNCATE TABLE minimundouvv.email;
TRUNCATE TABLE minimundouvv.disciplinas;
TRUNCATE TABLE minimundouvv.materias;
TRUNCATE TABLE minimundouvv.alunos;
TRUNCATE TABLE minimundouvv.cursos;
TRUNCATE TABLE minimundouvv.coordenadores;
TRUNCATE TABLE minimundouvv.professores;
set foreign_key_checks=1;

-- Limpando os indices e realocando os mesmos

CALL LimparIndices();
CALL AdicionarIndices();

-- Procedures para povoar as tabelas
CALL povoar_professores(20);
CALL povoar_materias(30);
CALL povoar_coordenadores(5);
CALL povoar_cursos(5);
CALL povoar_alunos(140);
CALL povoar_disciplinas(15);
CALL povoar_alunos_has_disciplinas(140);
CALL povoar_email(160);

```

TRIGGERS

Criamos Alguns triggers para operar em cima da tabela alunos, esses gatilhos indentificam e contabilizam modificações, remoções , adições de alunos, eles também verificam o numero de alunos por curso a cada atualização.

```
-- Rafael Christian Silva Wernesbach

-- Trigger para atualizar a data de modificação dos alunos
DROP TRIGGER IF EXISTS trg_alunos_update;

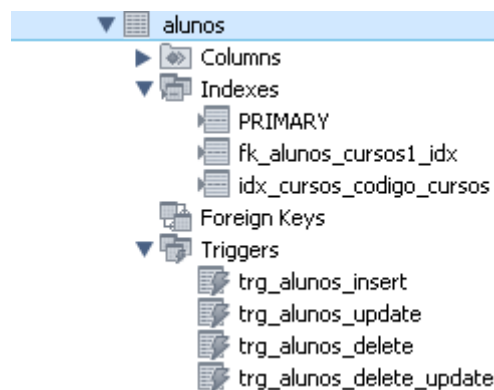
CREATE TRIGGER trg_alunos_update
BEFORE UPDATE ON alunos
FOR EACH ROW
SET NEW.data_ultima_modificacao = NOW();

-- Trigger para registrar remoções de alunos
DROP TRIGGER IF EXISTS trg_alunos_delete;
DELIMITER $$
CREATE TRIGGER trg_alunos_delete
AFTER DELETE ON alunos
FOR EACH ROW
BEGIN
    INSERT INTO log_exclusoes_alunos (matricula, nome, data_exclusao)
    VALUES (OLD.matricula_alunos, OLD.nome, NOW());
END $$
DELIMITER ;
```

```
-- Rafael Christian Silva Wernesbach

-- Trigger para contabilizar o numero de alunos em um curso
DROP TRIGGER IF EXISTS trg_alunos_insert;
DELIMITER $$
CREATE TRIGGER trg_alunos_insert
AFTER INSERT ON alunos
FOR EACH ROW
BEGIN
    UPDATE cursos
    SET numero_de_alunos = numero_de_alunos + 1
    WHERE codigo_cursos = NEW.cursos_codigo_cursos
    AND Coordenadores_matricula_cordenador = NEW.cursos_Coordenadores_matricula_cordenador;
END $$
DELIMITER ;

-- Trigger para decrementar o numero de alunos contabilizados
DROP TRIGGER IF EXISTS trg_alunos_delete_update;
DELIMITER $$
CREATE TRIGGER trg_alunos_delete_update
AFTER DELETE ON alunos
FOR EACH ROW
BEGIN
    UPDATE cursos
    SET numero_de_alunos = numero_de_alunos - 1
    WHERE codigo_cursos = OLD.cursos_codigo_cursos
    AND Coordenadores_matricula_cordenador = OLD.cursos_Coordenadores_matricula_cordenador;
END $$
DELIMITER ;
```



Consultas(Feitas por Rafael Azevedo e Cauã Agrisi)

Imagem 1: Listar todos os cursos com seus respectivos coordenadores e professores responsáveis

Imagem 2: Encontrar as disciplinas com mais de 50 vagas e listá-las com os professores responsáveis



Imagem 3: Encontrar professores que não estão associados a nenhuma disciplina

Imagem 4: Contar o número de disciplinas oferecidas por cada professor

```

1 • SELECT
2     p.matricula_professores,
3     p.nome AS nome_professor,
4     p.email
5 FROM
6     professores p
7 LEFT JOIN
8     disciplinas d ON p.matricula_professores = d.professores_matricula_professores
9 WHERE
10    d.professores_matricula_professores IS NULL
11 ORDER BY
12     p.matricula_professores;
13
14  /* Feita por Cauã Agrisi

```

Result Grid			
Filter Rows: <input type="text"/>			
Export: 			
Wrap Cell Content: 			
	matricula_professores	nome_professor	email
▶	1001	Professor - 1	professor_UVV_1@gmail.com.br
	1002	Professor - 2	professor_UVV_2@gmail.com.br
	1005	Professor - 5	professor_UVV_5@gmail.com.br
	1006	Professor - 6	professor_UVV_6@gmail.com.br
	1011	Professor - 11	professor_UVV_11@gmail.com.br
	1013	Professor - 13	professor_UVV_13@gmail.com.br
	1014	Professor - 14	professor_UVV_14@gmail.com.br
	1019	Professor - 19	professor_UVV_19@gmail.com.br
	1020	Professor - 20	professor_UVV_20@gmail.com.br

```

1 • SELECT
2     p.nome AS nome_professor,
3     COUNT(d.codigo_disciplina) AS numero_disciplinas
4 FROM
5     professores p
6 LEFT JOIN
7     disciplinas d ON p.matricula_professores = d.professores_matricula_professores
8 GROUP BY
9     p.matricula_professores
10 ORDER BY
11     numero_disciplinas DESC;
12
13 /* Feita por Rafael Azevedo

```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	nome_professor	numero_disciplinas			
▶	Professor - 7	2			
	Professor - 10	2			
	Professor - 16	2			
	Professor - 17	2			
	Professor - 3	1			
	Professor - 4	1			
	Professor - 8	1			
	Professor - 9	1			
	Professor - 12	1			
	Professor - 15	1			
	Professor - 18	1			
	Professor - 1	0			
	Professor - 2	0			
	Professor - 5	0			
	Professor - 6	0			
	Professor - 11	0			
	Professor - 13	0			
	Professor - 14	0			
	Professor - 19	0			

```

1 • CREATE INDEX idx_cursos_coordenadores ON cursos (Coordenadores_matricula_cordenador);
2
3 • CREATE INDEX idx_coordenadores_professores ON coordenadores (Professores_matricula_professores);
4
5 • CREATE INDEX idx_cursos_codigo ON cursos (codigo_cursos);
6
7 • SELECT
8     c.codigo_cursos,
9     c.nome AS nome_curso,
10    coord.nome AS nome_coordenador,
11    prof.nome AS nome_professor
12 FROM
13     cursos c
14 JOIN
15     coordenadores coord ON c.Coordenadores_matricula_cordenador = coord.matricula_cordenador
16 JOIN
17     professores prof ON coord.Professores_matricula_professores = prof.matricula_professores
18 ORDER BY
19     c.codigo_cursos;
20
21 /* Feita por Rafael Azevedo

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
codigo_cursos	nome_curso	nome_coordenador	nome_professor
101	Curso - 1	Coordenador - 4	Professor - 15
102	Curso - 2	Coordenador - 3	Professor - 18
103	Curso - 3	Coordenador - 5	Professor - 20
104	Curso - 4	Coordenador - 3	Professor - 18
105	Curso - 5	Coordenador - 4	Professor - 15

```

1 • CREATE INDEX idx_disciplinas_professores ON disciplinas (professores_matricula_professores);
2
3 • CREATE INDEX idx_disciplinas_vagas ON disciplinas (vagas);
4
5 • SELECT
6     d.codigo_disciplina,
7     d.nome AS nome_disciplina,
8     d.vagas,
9     p.nome AS nome_professor
10 FROM
11     disciplinas d
12 JOIN
13     professores p ON d.professores_matricula_professores = p.matricula_professores
14 WHERE
15     d.vagas > 50
16 ORDER BY
17     d.vagas DESC;
18
19 /* Feita por Cauã Agrisi

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
codigo_disciplina	nome_disciplina	vagas	nome_professor
105	Disciplinas - 5	58	Professor - 7
112	Disciplinas - 12	56	Professor - 7
104	Disciplinas - 4	56	Professor - 12
114	Disciplinas - 14	53	Professor - 10
102	Disciplinas - 2	52	Professor - 3

Para Povoar o Banco de Dados após rodar o script de criação, Utiliza se o seguinte código:

-- Limpando os dados das tabelas

```
set foreign_key_checks=0;
TRUNCATE TABLE minimundouv.v.alunos_has_disciplinas;
TRUNCATE TABLE minimundouv.v.email;
TRUNCATE TABLE minimundouv.v.disciplinas;
TRUNCATE TABLE minimundouv.v.materias;
TRUNCATE TABLE minimundouv.v.alunos;
TRUNCATE TABLE minimundouv.v.cursos;
TRUNCATE TABLE minimundouv.v.coordenadores;
TRUNCATE TABLE minimundouv.v.professores;
set foreign_key_checks=1;
```

-- Limpando os indices e realocando os mesmos

```
CALL LimparIndices();
CALL AdicionarIndices();
```

-- Procedures para povoar as tabelas

```
CALL povoar_professores(20);
CALL povoar_materias(30);
CALL povoar_coordenadores(5);
CALL povoar_cursos(5);
CALL povoar_alunos(140);
CALL povoar_disciplinas(15);
```


CALL povoar_alunos_has_disciplinas(140);

CALL povoar_email(160);