UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

# FOLLOW

...

Guilherme Morone Araujo (gma2)
Lucca Morosini Gioia (lmg2)
Rafael Leite de Oliveira (rlo3)
Raul Pereira Coelho (rpc3)
Williams Santiago de Souza Filho (wssf)

# MINIMUNDO

Um centro de informática de uma faculdade deseja fazer um banco de dados para gerenciar seus **alunos** e **professores** com relação aos **cursos**, suas **disciplinas** e atividades extracurriculares. Assim, uma **pessoa** (CPF, nome, dt_nascimento, endereço(CEP, logradouro)) pode ser um **professor** (código_professor, telefones) ou um **aluno** (matrícula). Dessa maneira, um **aluno** deve estar matriculado em apenas um **curso** (cod_curso, nome) e um **curso** pode ter vários **alunos** matriculados.

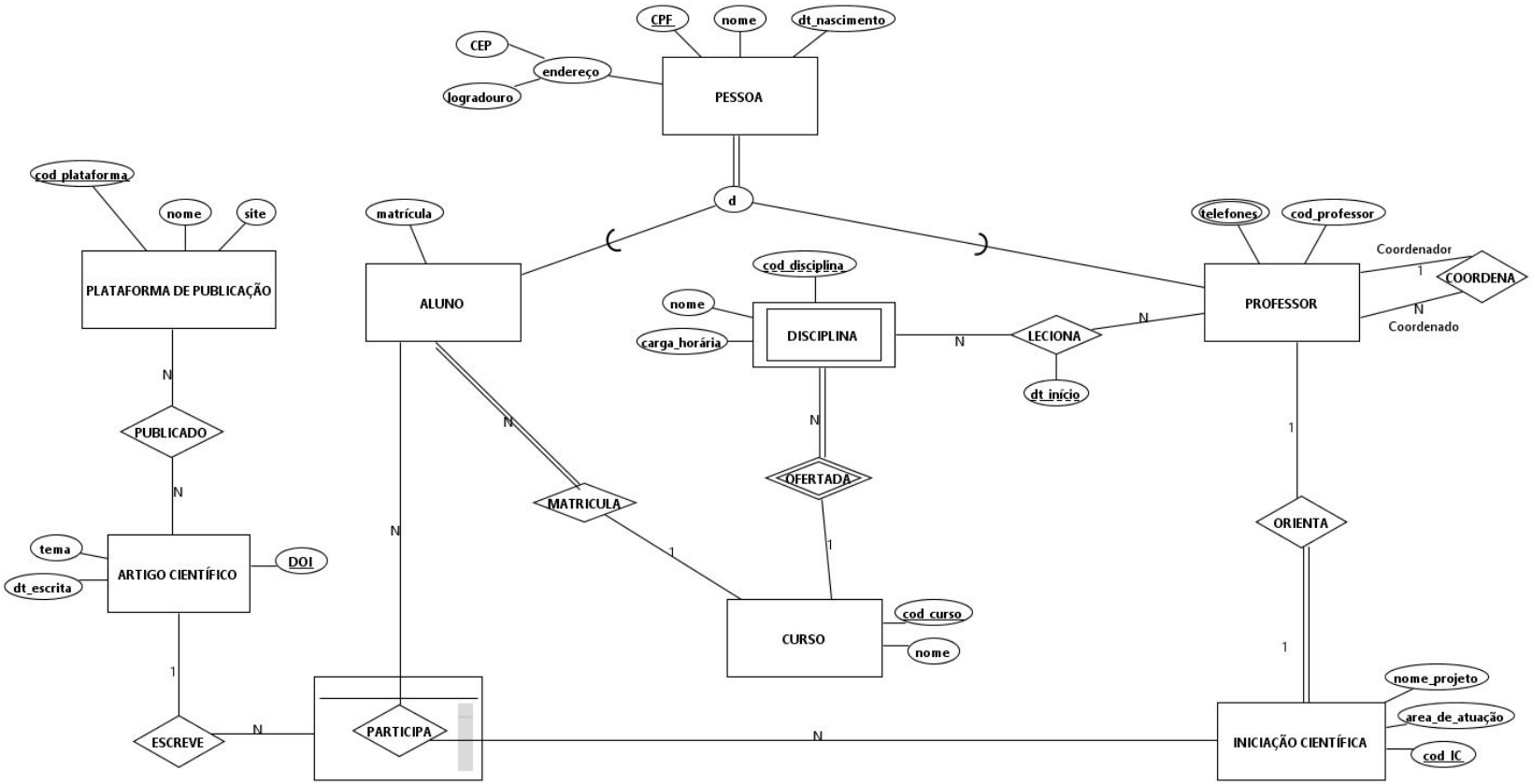
Sabendo disso, nota-se que um **aluno** pode participar de uma **Iniciação Científica** (código_IC, área_de_atuação, nome_projeto), a qual pode ser formada por vários **alunos**. Combinado a isso, um **professor** pode orientar **alunos** em apenas uma **Iniciação Científica**, que só pode ser orientada por um **professor**. Dessa forma, durante um ou mais projetos de **Iniciação Científica**, um **aluno** participante pode escolher entre escrever ou não um **artigo científico**. Contudo, tal **artigo científico** (DOI - Digital Object Identifier - identificador único para artigos, tema, dt_escrita) pode ser escrito por um ou mais **alunos** de uma **Iniciação Científica**.

É importante pontuar também que um **artigo científico** pode ser publicado em várias **plataformas de publicação** (cod_plataforma, site, nome), na qual podem ter vários **artigos científicos** publicados.

Com relação às atividades curriculares regulares, um **professor** pode lecionar várias **disciplinas** (código_disciplina, nome, carga_horária), as quais podem ter mais de um **professor**. Assim, já que um **professor** pode lecionar a mesma **disciplina** várias vezes ao longo do tempo, deve-se registrar a data em que ele iniciou o ensino da mesma. Além disso, sabe-se que uma **disciplina** é discriminada por um determinado **curso**, o qual pode ter várias **disciplinas**.

Finalmente, pontua-se que um **professor** pode ser coordenador de vários outros **professores**, os quais só podem ter um único coordenador.

# PROJETO CONCEITUAL



# PROJETO LÓGICO RELACIONAL

- PlataformaPublicação(cod_plataforma, nome, site);
- ArtigoCientífico(DOI, dt_escrita, tema);
- Curso(cod_curso, nome);
- Disciplina(cod_curso, cod_disciplina, nome, carga_horária);
- Aluno(CPF, nome, dt_nascimento, end_CEP, end_logradouro, matrícula, cod_curso!)
  - cod_curso → Curso(cod_curso);
- Professor(CPF, nome, dt_nascimento, end_CEP, end_logradouro, cod_professor, CPF_coordenador)
  - CPF_coordenador → Professor(CPF);
- Telefone(CPF, contato)
  - CPF → Professor(CPF);
- IniciaçãoCientífica(cod_IC, área_de_atuação, nome_projeto, [CPF_professor]!)
  - CPF_professor → Professor(CPF);
- Participa(CPF_Aluno, cod_IC, DOI_artigo)
  - CPF_Aluno → Aluno(CPF)
  - cod_IC → IniciaçãoCientífica(cod_IC)
  - DOI_artigo → ArtigoCientífico(DOI);
- Publicado(DOI, cod_plataforma)
  - DOI → ArtigoCientífico(DOI)
  - cod_plataforma → PlataformaPublicação(cod_plataforma);
- Leciona(CPF_professor, cod_curso, cod_disciplina, dt_início)
  - CPF_professor → Professor(CPF)
  - (cod_curso, cod_disciplina) → Disciplina(cod_curso, cod_disciplina);

# Consultas - Group By/Having

Agrupar, por curso, a quantidade de disciplinas que tem carga horária = 75

```
SELECT cod_curso, COUNT(*) AS Qtd
FROM Disciplina
WHERE CARGA_HORARIA = 75
GROUP BY cod_curso
ORDER BY Qtd DESC;
```

Agrupar, por CPF do Professor, a quantidade de disciplinas lecionadas após o ano de 2019 (Projetar apenas os professores que lecionam mais de 1 disciplina)

```
SELECT CPF_professor,
COUNT(cod_curso)
FROM Leciona
WHERE dt_inicio > DATE’2019-12-31’
GROUP BY CPF_professor
HAVING COUNT(cod_curso) > 1;
```

# Consultas - Inner Join

Projetar os Professores e os Códigos das Disciplinas que eles lecionam, sendo o código do curso dessas disciplinas = 'CC001'

```
SELECT p.nome AS Professor, l.cod_disciplina as Disciplina
FROM Professor p
INNER JOIN Leciona l
    ON p.CPF = l.CPF_professor
WHERE l.cod_curso = 'CC001';
```

# Consultas - Outer Join

`Projetar o nome do Professor e seu Coordenador, inclusive quando ele não tem nenhum Coordenador`

```
SELECT p1.nome AS Coordenado, p2.nome AS Coordenador
FROM Professor p1
LEFT OUTER JOIN Professor p2
    ON p1.CPF_coordenador = p2.CPF;
```

# Consultas - Semi Join

Projetar os Temas dos Artigos Científicos publicados, cujo código da plataforma de publicação = '167QP67850RQZ30'

```
SELECT a.tema as Tema
FROM ArtigoCientifico a
WHERE EXISTS (
    SELECT * FROM Publicado p
    WHERE p.DOI = a.DOI AND p.cod_plataforma = '167QP67850RQZ30'
);
```

# Consultas - Anti Join

Projetar os alunos que não tem nenhum artigo científico escrito

```
SELECT a.nome AS Nome
FROM Aluno a
WHERE a.CPF NOT IN (
   SELECT p.CPF_Aluno FROM Participa p
   WHERE p.DOI_artigo IS NOT NULL
);
```

# Consultas - Subconsulta do tipo escalar

`Projetar os nomes das Disciplinas com carga horária maior do que a carga horária média`

```
SELECT D.nome AS Nome, cod_curso AS CodigoDoCurso
FROM Disciplina D
WHERE D.carga_horaria > (SELECT AVG(D2.carga_horaria)
                         FROM Disciplina D2);
```

# Consultas - Subconsulta do tipo linha

Projetar os nomes dos alunos que nasceram no mesmo ano e fazem o mesmo curso que o(a) aluno(a) de CPF 500.896.666-30

```
SELECT A.nome
FROM Aluno A
WHERE (EXTRACT(YEAR FROM A.dt_nascimento), A.cod_curso) = (SELECT
EXTRACT(YEAR FROM A2.dt_nascimento), A2.cod_curso
                                        FROM Aluno A2
                                        WHERE A2.CPF = '500.896.666-30');
```

# Consultas - Subconsulta do tipo tabela

Projetar os nomes dos professores que são coordenadores

```
SELECT P.nome AS Nome
FROM Professor P
WHERE P.CPF IN (
    SELECT P2.CPF_coordenador
    FROM Professor P2
    WHERE P2.CPF_coordenador IS NOT NULL
);
```

# Consultas - Operação de Conjunto

Projetar o nome de todos alunos e professores cujo nome começa com a letra M

```
SELECT A.nome, 'Aluno(a)' as AlunoOuProfessor
FROM Aluno A
WHERE A.nome LIKE 'M%'
UNION
SELECT P.nome, 'Professor(a)' as AlunoOuProfessor
FROM Professor P
WHERE P.nome LIKE 'M%';
```

# PL/SQL - Procedimento

`Printar o CPF e nome dos alunos que cursam o curso com código = cursoCodigo`

```
CREATE OR REPLACE PROCEDURE alunos(cursoCodigo Aluno.cod_curso%TYPE) IS
  CURSOR cur_alunos IS
    SELECT CPF, nome
    FROM Aluno
    WHERE cod_curso = cursoCodigo;
  BEGIN
    DBMS_OUTPUT.PUT_LINE('CPF e nome dos alunos que cursam ' || cursoCodigo || ':');
    FOR reg_cursor IN cur_alunos LOOP
      DBMS_OUTPUT.PUT_LINE('CPF: ' || reg_cursor.CPF || ', nome: ' || reg_cursor.nome);
    END LOOP;
  END;
```

# PL/SQL - Procedimento

```
Printar o nome, o código da disciplina e o código do curso que o professor com CPF = CPFdoProfessor leciona
```

```
CREATE OR REPLACE PROCEDURE professoresEDisciplina(CPFdoProfessor Professor.CPF%TYPE) IS
  CURSOR cur_Professor IS
    SELECT D.nome, D.cod_disciplina, D.cod_curso
    FROM Disciplina D
    WHERE (D.cod_disciplina, D.cod_curso) IN (
      SELECT L.cod_disciplina, L.cod_curso
      FROM Leciona L
      WHERE L.CPF_Professor = CPFdoProfessor );
BEGIN
    DBMS_OUTPUT.PUT_LINE('Nome e códigos das disciplinas lecionadas pelo professor cujo CPF = ' ||
CPFdoProfessor || ':');
    FOR reg_cursor IN cur_Professor LOOP
      DBMS_OUTPUT.PUT_LINE('Nome: ' || reg_cursor.nome || ', cod_disciplina: ' || reg_cursor.cod_disciplina || ',
cod_curso: ' || reg_cursor.cod_curso);
    END LOOP;
  END;
```

# PL/SQL - Função

```
Conta a quantidade de pessoas cujo nome começa com a letra dada
```

```
CREATE OR REPLACE FUNCTION contaPessoas(letraAluno CHAR, letraProf CHAR) RETURN
INT IS
  contaAluno INT;
  contaProfessor INT;
  BEGIN
    SELECT COUNT(*) INTO contaAluno
    FROM Aluno
    WHERE nome LIKE CONCAT(letraAluno, '%');
    SELECT COUNT(*) INTO contaProfessor
    FROM Professor
    WHERE nome LIKE CONCAT(letraProf, '%');
    RETURN contaAluno + contaProfessor;
  END;
```

# PL/SQL - Função

```
Retorna o código do artigo mais velho

CREATE OR REPLACE FUNCTION artigoMaisAntigo RETURN VARCHAR2 IS
  doiMaisVelho ArtigoCientifico.DOI%TYPE;
  BEGIN
    SELECT A1.DOI INTO doiMaisVelho
    FROM ArtigoCientifico A1
    WHERE A1.dt_escrita = (
      SELECT MIN(A2.dt_escrita)
      FROM ArtigoCientifico A2
    );
    RETURN doiMaisVelho;
  END;
```

# PL/SQL - Gatilho(Trigger)

Evento que printa o nome antigo e novo do aluno ao ser modificado

```
CREATE OR REPLACE TRIGGER evento
AFTER DELETE OR INSERT OR UPDATE OF nome ON Aluno
FOR EACH ROW
   BEGIN
      IF UPDATING THEN
         DBMS_OUTPUT.PUT_LINE('Atualizando...');
         DBMS_OUTPUT.PUT_LINE('Nome antigo: ' || :OLD.nome || ', Nome novo: ' || :NEW.nome);
      ELSIF INSERTING THEN
         DBMS_OUTPUT.PUT_LINE('Inserindo...');
         DBMS_OUTPUT.PUT_LINE('Nome: ' || :NEW.nome);
      ELSE
         DBMS_OUTPUT.PUT_LINE('Deletando...');
         DBMS_OUTPUT.PUT_LINE('Nome deletado: ' || :OLD.nome);
      END IF;
   END;
```