Sistema de Auxílio à Secretaria do Departamento de Informática

INF1383

Clara Szwarcman

Daniel Siles

Fernando Roisman

Lucas Borges

Guilherme Simas

Simon Travancas

Índice

I. Introdução

O trabalho de administração e gerência dos cursos oferecidos pela Universidade não é trivial, e ferramentas que auxiliem neste processo são sempre bem vindas. Oferecer uma forma simples e expressiva de realizar consultas ao universo de informações disponíveis sobre os aspectos funcionais da vida acadêmica ao Departamento de Informática é o principal objetivo deste trabalho.

O sistema permitirá o acesso e operações a informações tais como: quais alunos dão monitoria no departamento e em quais cadeiras; quais alunos estagiam em quais laboratórios do departamento; qual professor orienta o projeto final do aluno e qual é o tema do projeto; qual professor orienta a tese do aluno e qual é o tema da tese e quais alunos são bolsistas e de que tipo.

Para a construção do banco de dados será necessário conhecer as seguintes informações:

Para cada aluno sua matrícula e seu nome. Se for de graduação, seu curso. Se for de pós-graduação serão necessárias mais informações pessoais, como os dados do passaporte, identidade, endereço e uma conta bancária, que poderá estar associada a mais de um aluno, assim como se o aluno é de mestrado ou doutorado. Para cada professor, seu nome e sua matrícula.

Um aluno pode ser monitor de até uma disciplina, enquanto uma disciplina pode possuir vários monitores. Sobre a disciplina é necessário conhecer seu código e nome. Também é preciso saber de qual turma o aluno foi monitor, a data de início e de fim da monitoria.

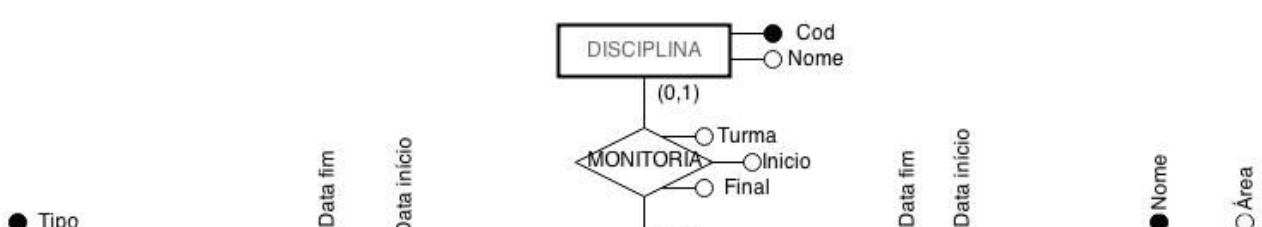
Um professor pode orientar e co-orientar vários alunos de pós-graduação. Um aluno de pós-graduação pode possuir até no máximo um orientador e um co-orientador.

Um professor também pode administrar vários projetos orientados, nos quais podem participar vários alunos de graduação. Um projeto possui um tema, número de créditos e um código. Entretanto, para um mesmo projeto o número de créditos pode variar para os alunos participantes. Também será conhecida a nota de cada aluno no projeto. Cada projeto é administrado por somente um professor. Um aluno de graduação pode participar de vários projetos.

Um aluno pode ser estagiário em até um laboratório. É necessário saber a data de início e de fim do estágio. Um laboratório possui, além de um nome e uma área, um professor responsável, e professores membros sem restrição de quantidade. Um professor só pode pertencer a um laboratório, seja como responsável ou membro.

Cada aluno pode possuir até duas bolsas simultaneamente, uma bolsa possui um tipo e pode estar associada a diversos alunos. Será conhecida a data de início do recebimento da bolsa e, se a bolsa não for mais vigente, a data de término.

II. Diagrama de Entidade-Relacionamento



III. Descrição Entidades e Atributos

Entidades e atributos

1. Aluno – Contém os alunos que se necessita cadastrar.

    - Matrícula – número de 7 dígitos gerado que identifica o aluno
    - Nome – texto que se refere ao nome do aluno (máximo de 50 caracteres)

    1.1 Graduação – Contém os alunos de graduação cadastrados

        - Curso –texto que se refere ao curso do aluno (máximo de 30 caracteres)

    1.2 Pós-Graduação – Contém os alunos de pós-graduação cadastrados

        - Tipo – texto referente a que tipo de pós-graduação o aluno cursa (máximo de 9 caracteres)
            - Pode ser “doutorado”,”mestrado”.
        - Área – texto que se refere à área de estudo do aluno (máximo de 50 caracteres)
        - Situação - texto que se refere a situação do aluno na pós-graduação (máximo de 10 caracteres)
            - Pode ser: “afastou-se”, “em curso”, “atrasado”, “formando”,”formado”.
        - CPF – Combinação de 14 caracteres (seguindo o modelo: ***.***.***-**) referente ao Cadastro de Pessoa Física
        - Anotação – texto referente à alguma anotação, se desejado (máximo de 200 caracteres)
            - Campo de observações
        - E-mail – texto referente ao e-mail do aluno (máximo de 30 caracteres)
            - Obrigatoriamente o da PUC (@inf.puc-rio.br)
        - Login – texto gerado para acesso do aluno (máximo de 20 caracteres)
        - Lattes – link referente ao currículo lattes do aluno (máximo de 50 caracteres)
            - Formato: “http://lattes.cnpq.br/xxxxxxxxxxxx”
        - Período Início – número de 4 digitos na parte antes do separador decimal e 1 depois, referente ao período de início
            - “2014.1”
        - Estimativa – número do período que é estimado para a conclusão

    - Mesmo formato de período início
- Status matrícula – texto referente a situação atual da matrícula (máximo de 12 caracteres)
    - Pode ser: “prorrogação”, “marcou banca”, “matriculado”, “reabrindo”, “afastou-se”, “trancamento”,”formado”
- Data Nascimento – Data referente ao nascimento do aluno (AAAA-MM-DD)
- Sexo – Caracter que indica o sexo do aluno (‘F’ para feminino e ‘H’ para masculino)
- País Nascimento – texto referente ao país em que o aluno nasceu (máximo de 20 caracteres)
    - Pode ser null
- Estado Origem – texto referente ao estado em que o aluno nasceu (máximo de 30 caracteres)
    - Pode ser null
- Nacionalidade – texto referente à nacionalidade do aluno (máximo de 30 caracteres)
- Naturalidade – texto referente à naturalidade do aluno (máximo de 30 caracteres)
- Identidade – número de até 10 dígitos referente ao número do Registro Geral
- Órgão Emissor – texto referente ao órgão que emitiu a identidade (máximo de 30 caracteres)
- Data de emissão – data referente à emissão da identidade (AAAA-MM-DD)
- Número Passaporte (se estrangeiro) – texto referente ao número do passaporte do aluno (máximo de 20 caracteres)
- País Passaporte – texto referente ao país do passaporte do aluno (máximo de 30 caracteres)
- Validade Passaporte – data referente a validade do passaporte (AAAA-MM-DD)
- Logradouro – texto referente ao endereço residencial (máximo de 30 caracteres)
- Cidade – texto referente a cidade em que o aluno mora (máximo de 30 caracteres)
- Estado – texto referente ao estado em que o aluno mora (máximo de 30 caracteres)
- CEP – número de 8 dígitos referente ao Código de Endereçamento Postal
- Telefone – número de até 11 dígitos referente ao telefone residencial do aluno
- Celular – número de até 11 dígitos referente ao celular do aluno
- Dedicação - texto que se refere ao tipo de dedicação do aluno (máximo de 9 caracteres)
    - Pode ser: “parcial”, “exclusiva”.

2 Disciplina – Contém as disciplinas do departamento de informática

- Código – texto gerado para identificar uma disciplina (3 letras maiúsculas seguidas de 4 números)
- Nome – texto referente ao nome da disciplina (máximo de 50 caracteres)

3 Professor – Contém os professores que se necessita cadastrar

- Matrícula – número de 7 dígitos que identifica um professor
- Nome – texto referente ao nome do professor (máximo de 50 caracteres)
- Departamento – texto referente ao departamento que o professor pertence

4 Laboratório – Contém os laboratórios do departamento de informática

- Nome – texto referente ao nome do laboratório (máximo de 50 caracteres)
- Área – texto referente a área de atuação do laboratório (máximo de 50 caracteres)

5 Bolsa – Contém as bolsas cadastradas na faculdade

- Tipo/Órgão – texto referente ao tipo da bolsa (máximo de 50 caracteres)

6 Conta Banco – Contém as contas de banco cadastradas

- Nº do banco – texto referente ao número do banco do aluno (máximo de 4 caracteres)
- Nº da agência – número referente ao número da agência do aluno (máximo de 8 algarismos)
- Nº da conta – número referente ao número da conta do aluno (máximo de 4 algarismos)
- Nome da Agência – texto referente ao nome da agência do aluno (máximo de 30 caracteres)

7 Projeto Orientado – Contém os Projetos Orientados cursados pelos alunos do departamento

- Código – texto gerado que identifica um projeto orientado (3 letras maiúsculas seguidas de 4 números)
- Tema – texto referente ao tema do projeto orientado (máximo de 50 caracteres)
- Número de Créditos – número referente à quantidade de créditos do projeto orientado (máximo de 2 algarismos)
- Período – número do período que foi realizado o projeto
    - Formato: 2014.1
- Nota – número da nota obtida no projeto (máximo de 2 algarismos antes da vírgula e 1 depois)

Relacionamentos

1. Monitoria – Relação entre um Aluno e uma Disciplina, na qual o aluno é monitor da disciplina. O aluno só pode ser monitor de uma disciplina, a disciplina pode ter vários monitores.

    - Turma – texto referente a qual turma da disciplina o aluno é monitor
    - Data Início – data referente ao início da monitoria
    - Data Fim – data referente ao término da monitoria, se já tiver ocorrido o término

2. Possui – Relação entre um Aluno e uma Bolsa, na qual o aluno possui uma bolsa. Um aluno pode possuir até 2 bolsas e cada bolsa pode estar atribuída a diversos alunos.

    - Data Início – data referente ao início do recebimento da bolsa
    - Data Fim – data referente ao término do recebimento da bolsa, se já tiver ocorrido o término

3. Estágio – Relação entre um aluno e um laboratório, na qual um aluno estagia no laboratório. Um laboratório pode ter vários estagiários, mas um aluno só estagia em até um laboratório.

    - Data Início – data referente ao início do estágio pelo aluno
    - Data Fim – data referente ao término do estágio pelo aluno, se já tiver ocorrido o término

4. Membro – Relação entre um Professor e um Laboratório, na qual o professor é membro do laboratório. Um laboratório pode ter vários membros, e um professor só pode ser membro de até um laboratório.

    - Data Início – data referente à quando o professor passou a ser membro do laboratório
    - Data Fim – data referente à quando o professor deixou de ser membro do laboratório, se já tiver deixado de ser membro

5. Responsável - Relação entre um Professor e um Laboratório, na qual o professor é responsável pelo laboratório. Um professor pode ser responsável por até um laboratório, e um laboratório contém exatamente um responsável.

    - Data Início – data referente à quando o professor passou a ser responsável pelo laboratório
    - Data Fim – data referente à quando o professor deixou de ser responsável pelo laboratório, se já tiver deixado de ser responsável

6. Participa – Relação entre um Aluno e um Projeto Orientado na qual o aluno participa do projeto. O projeto tem um aluno participante, e o aluno pode participar de vários projetos.

7. Orienta Mestrando – Relação entre um Aluno de Mestrado e um Professor, na qual o professor orienta o aluno. Um mestrando pode ter até 1 orientador, e um professor pode orientar vários mestrandos.

8. Orienta Mestrando – Relação entre um Aluno de Doutorado e um Professor, na qual o professor orienta o aluno. Um doutorando deve ter 1 orientador, e um professor pode orientar vários doutorandos.

9. Co-orienta - Relação entre um Aluno de Doutorado e um Professor, na qual o professor co-orienta o aluno. Um doutorando pode ter até 1 co-orientador, e um professor pode co-orientar vários doutorandos.

10. Administra – Relação entre um Professor e um Projeto Orientado, na qual o professor administra o projeto. Um projeto tem somente um professor que o administra, e um professor pode administrar vários projetos.

11. Vinculado – Relação entre uma Conta de Banco e um aluno de Pós-Graduação, na qual o aluno está vinculado a uma conta. Cada aluno possui exatamente uma conta de banco, e cada conta de banco está vinculada a pelo menos um aluno.

IV. Esquema Lógico - Relacional

Tabelas

- ALUNO (Matricula ,Nome)

- GRAD (Matricula , Curso)
    - Matricula referencia ALUNO

- POSGRAD (*Matricula*, Tipo, Area, Situacao, CPF, Anotacao, E-mail, Login, Lattes, PeriodoInicio, Estimativa, StatusMatricula, DataNascimento, Sexo, PaisNascimento, EstadoOrigem, Nacionalidade, Naturalidade, Identidade, OrgaoEmissor, DataEmissao, NumPassaporte, PaisPassaporte, ValidadePassaporte, Logradouro, Cidade, Estado, CEP, Telefone, Celular, NumBanco, NumConta, NumAgencia, NomeAgencia, Dedicação, *Orientador*, *Co-orientador*).

    - Matricula referencia ALUNO

    - Orientador referencia PROFESSOR (Relação Orienta). (Pode ser nulo)
    - Co-orientador referencia PROFESSOR (Relação Co-orienta). (Pode ser nulo)

- PROFESSOR (Matricula, Nome, Departamento, *NomeLab*, DataIniMembro, DataFimMembro).
    - NomeLab referencia LABORATORIO (Relação Membro). (Pode ser nulo)

- DISCIPLINA (Codigo, Nome).

- LABORATORIO (Nome, Area, *Responsavel*).
    - Responsavel referencia PROFESSOR. (Relação responsável)

- BOLSA (Tipo, *Aluno*, DataIni, DataFim).
    - Aluno referencia ALUNO (Relação Possui). (Pode ser Nulo)

- MONITORIA (*Aluno*, *CodDisc*, DataIni, DataFim)
    - Aluno referencia ALUNO.
    - CodDisc referencia Disciplina

- ESTAGIOLAB (*Aluno*, *Laboratorio*, DataIni, DataFim)
    - Aluno referencia ALUNO.
    - Laboratorio referencia LABORATORIO

- PROJETOORIENTADO (Codigo, *Aluno*, Tema, NumCred, Nota, Periodo, *MatriculaProf*).
    - Aluno referencia GRAD.
    - MatriculaProf referencia PROFESSOR (Relação Administra).

V. Restrições de Integridade Semântica

Foram definidas as seguintes Regras de Negócio:

- Um aluno de pós-graduação cursando um doutorado tem obrigatoriamente um orientador.

- Um aluno de pós-graduação cursando um mestrado pode não ter um orientador durante a primeira metade do programa (o mestrado dura 4 períodos).

- Um aluno não pode ser responsável por mais de uma monitoria ao mesmo tempo. Não deve haver interseção entre o fim de uma monitoria e o início de outra.

- Um aluno não pode estagiar em mais de um laboratório ao mesmo tempo. Não deve haver interseção entre o fim de um estágio e o início de outro.

- Um doutorando possui dedicação exclusiva ao curso obrigatoriamente.

VI. Criação das Tabelas

A seguir estão os códigos em SQL referentes à criação das tabelas especificadas na seção IV.

```
CREATE TABLE aluno
(
matricula numeric(7) NOT NULL,

Constraint pk_aluno Primary Key (matricula),

nome varchar(50) NOT NULL

);


CREATE TABLE grad
(
matricula numeric(7) NOT NULL,

Constraint fk_graduacao Foreign Key (matricula)
References aluno (matricula),

curso varchar(30) NOT NULL

);


CREATE TABLE posgrad
```

```
(
matricula numeric(7) NOT NULL,

Constraint fk_posgrad_matricula Foreign Key (matricula)
References aluno (matricula),
CONSTRAINT pk_posgrad Primary Key (matricula),

tipo varchar(9) NOT NULL,

Constraint CKC_tipo_posgrad
Check (tipo IN ('doutorado','mestrado')),

area varchar(50) NOT NULL,
situacao varchar(10) NOT NULL,

Constraint CKC_situacao_posgrad
Check (situacao IN ('afastou-se','em curso','atrasado','formando',’formado’)),

CPF char(14) NOT NULL,

CONSTRAINT CKC_cpf CHECK (cpf like ('___.___.___-__')),
CONSTRAINT UNI_cpf UNIQUE (cpf),

anotacao varchar(200) ,
email varchar(30) NOT NULL,

Constraint CKC_email
CHECK (email like ('%@inf.puc-rio.br')),

login varchar(20) NOT NULL,

CONSTRAINT UNI_login UNIQUE (login),

lattes varchar(50) NOT NULL,

CONSTRAINT CKC_lattes CHECK (lattes like ('http://lattes.cnpq.br/%')),
CONSTRAINT UNI_lattes UNIQUE (lattes),

periodoinicio numeric(5,1) NOT NULL,
estimativa numeric(5,1) NOT NULL,
status_matricula varchar(12) NOT NULL,
```

```
Constraint CKC_status_posgrad
Check (status_matricula IN ('prorrogação','marcou banca','matriculado',
'reabrindo','afastou-se','trancamento',’formado’)),

data_nascimento date NOT NULL,
sexo char(1) NOT NULL,

Constraint CKC_sexo_posgrad
Check (sexo IN ('F','H')),

pais_nascimento varchar(20),
estado_origem varchar(30),
nacionalidade varchar(30) NOT NULL,
naturalidade varchar(30) NOT NULL,
id varchar(10),
orgao_emissor varchar(30),
data_emissao date,
num_passaporte varchar(20),
pais_passaporte varchar(30),
validade_passaporte date,
logradouro varchar(30) NOT NULL,
cidade varchar(30) NOT NULL,
estado varchar(30) NOT NULL,
cep numeric(8) NOT NULL,
telefone numeric(11),
celular numeric(11),
num_banco char(4) NOT NULL,
num_conta numeric(8) NOT NULL,
num_agencia numeric(4) NOT NULL,
nome_agencia varchar(30) NOT NULL,
dedicacao varchar(9) NOT NULL,

Constraint CKC_dedicacao_posgrad
Check (dedicacao IN ('parcial','exclusiva')),

orientador numeric(7),

Constraint fk_posgrad_orientador Foreign Key (orientador)
References professor (matricula),

coorientador numeric(7),
```

```
Constraint fk_posgrad_coorientador Foreign Key (coorientador)
References professor (matricula)

);



CREATE TABLE professor
(
matricula numeric(7) NOT NULL,

Constraint pk_professor Primary Key (matricula),

nome varchar(50) NOT NULL,
nome_lab varchar(50),


data_ini_membro date ,
data_fim_membro date

);

ALTER TABLE professor ADD Constraint fk_professor_nome_lab Foreign Key
(nome_lab)
References laboratorio (nome);



CREATE TABLE disciplina
(
codigo char(7) NOT NULL,

Constraint pk_disciplina Primary Key (codigo),

nome varchar(50) NOT NULL

);


CREATE TABLE laboratorio
(
nome varchar(50) NOT NULL,
```

```
Constraint pk_laboratorio Primary Key (nome),

area varchar(50) NOT NULL,
responsavel numeric(7) NOT NULL,

Constraint fk_laboratorio_responsavel Foreign Key (responsavel)
References professor (matricula)

);


CREATE TABLE bolsa
(
tipo varchar(50) NOT NULL,

aluno numeric(7) NOT NULL,

Constraint fk_bolsa_aluno Foreign Key (aluno)
References aluno (matricula),

data_ini date NOT NULL,

Constraint pk_bolsa Primary Key (aluno,tipo,data_ini),

data_fim date

);


CREATE TABLE monitoria
(
aluno numeric(7) NOT NULL,

Constraint fk_monitoria_aluno Foreign Key (aluno)
References aluno (matricula),

codigo char(7) NOT NULL,

Constraint fk_monitoria_codigo Foreign Key (codigo)
References disciplina (codigo),
```

```
data_ini date NOT NULL,

Constraint pk_monitoria Primary Key (codigo,aluno,data_ini),

data_fim date

);


CREATE TABLE estagiolab
(
aluno numeric(7) NOT NULL,

Constraint fk_estagiolab_aluno Foreign Key (aluno)
References aluno (matricula),

laboratorio varchar(50) NOT NULL,

Constraint fk_estagiolab_laboratorio Foreign Key (laboratorio)
References laboratorio (nome),

data_ini date NOT NULL,

Constraint pk_estagiolab Primary Key (laboratorio,aluno,data_ini),

data_fim date
);


CREATE TABLE projetoorientado
(
aluno numeric(7) NOT NULL,

Constraint fk_projeto_orientado_aluno Foreign Key (aluno)
References aluno (matricula),

codigo char(7) NOT NULL,

Constraint pk_projeto_orientado Primary Key (codigo,aluno),

tema varchar(50) NOT NULL,
numcred numeric(2) NOT NULL,
```

```
nota numeric(3,1) NOT NULL,
periodo numeric(5,1) NOT NULL,
matricula_prof numeric(7) NOT NULL,

Constraint fk_projeto_orientado_matricula_prof Foreign Key (matricula_prof)
References professor (matricula)
);
```

VII. Consultas em SQL

1 - Todos os atributos (*), Subconsulta aninhada, IN, NOT LIKE '%', ORDER BY, IS NULL

Quais são os alunos de graduação que dão monitoria e não fazem Engenharia?

```
SELECT *
FROM Aluno
WHERE A.Matricula IN ( SELECT G.matricula
                       FROM Graduacao as G, Monitoria as M
                       WHERE G.matricula = M.aluno
                       AND G.curso NOT LIKE '%Engenharia%'
                       AND M.datafim IS NULL)
ORDER BY Nome
```

2 - LEFT OUTER JOIN

Listar todos os monitores, e se estagiarem em algum laboratório, o nome do laboratório.

```
SELECT M.Matricula, M.Nome, E.Nome
FROM MONITORIA as M left outer join ESTAGIOLAB as E on M.Aluno = E.Aluno
```

3 - SUM , INNER JOIN

Qual a quantidade de créditos administrados pelo professor "Sergio Lifschitz" na disciplina Projeto Orientado?

```
SELECT SUM NumCred
FROM PROJETOORIENTADO INNER JOIN PROFESSOR ON
PROJETOORIENTADO.MatriculaProf = PROFESSOR.Matricula
WHERE PROFESSOR.nome = 'Sergio Lifschitz'
```

4- COUNT

Quantos alunos de Pos-Graduação se matricularam em 2013?

```
SELECT COUNT (*)
FROM POSGRAD
WHERE POSGRAD.PeriodoInicio like '2013.?'
```

5 - INTERSECT

Quais professores são orientadores e co-orientadores?

```
SELECT Nome
FROM PROFESSOR
WHERE Matricula IN
        (SELECT P.Matricula
        FROM PROFESSOR as P, POSGRAD as PG
        WHERE PG.Co-orientador=P.Matricula)
        INTERSECT
        (SELECT P.Matricula
        FROM PROFESSOR as P, POSGRAD as PG
        WHERE PG.Orientador=P.Matricula)
```

6 - EXCEPT

Quais os atuais monitores de Programação I?

```
SELECT Aluno
FROM MONITORIA
WHERE CodDiscp = ‘INF1005’
EXCEPT(
        SELECT Aluno
        FROM MONITORIA
        WHERE DataFim IS NULL
        )
```

7 - UNION

Quais alunos são monitores, estagiários, ou ambos?

```
SELECT Aluno
FROM MONITORIA
WHERE DataFim is NULL
UNION (
        SELECT Aluno
        FROM ESTAGIOLAB
        WHERE DataFim is NULL
        )
```

8 - SOME

Quais laboratórios tem 3 ou 7 membros?

```
SELECT NomeLab
FROM ( SELECT NomeLab, COUNT(Matrícula) AS NumMembros
        FROM Professor GROUP BY NomeLab ) AS A
WHERE NumMembros = SOME (3,7)
```

9 - ALL

Quais laboratórios nao tem 2 nem 5 membros?

```
SELECT NomeLab
FROM ( SELECT NomeLab, COUNT(Matrícula) AS NumMembros
         FROM Professor GROUP BY NomeLab ) AS A
WHERE NumMembros != ALL (2,5)
```

10 - EXISTS

Quais professores deram um projeto orientado no período de 2014.1?

```
SELECT P.matricula, P.nome
FROM professor as P
WHERE EXISTS ( SELECT *
                    FROM projetoorientado as PJO
                    WHERE PJO.matriculaprof = P.matricula
```

```
                AND PJO.periodo=2014.1 )
```

11 - '????'

Quais os alunos que tem o primeiro nome com 5 letras?

```
SELECT A.matricula, A.nome
FROM Aluno as A
WHERE A.nome LIKE '????? %'
```

12 - Having

Quantos alunos deram mais de uma monitoria?

```
SELECT aluno.matricula, aluno.nome, count(*)
FROM aluno, monitoria
WHERE aluno.matricula=monitoria.aluno
AND EXISTS (SELECT *
                FROM monitoria
                WHERE monitoria.aluno=aluno.matricula
                AND monitoria.data_fim IS NULL)
GROUP BY aluno.matricula, aluno.nome
HAVING count(*)>1
```

13 – Unique

Quantos alunos deram mais de uma monitoria?

```
SELECT DISTINCT aluno.matricula,aluno.nome
FROM aluno
WHERE NOT UNIQUE (
        SELECT (*)
        FROM monitoria
        WHERE aluno.matricula=monitoria.aluno
)
```

VIII. Consultas em Álgebra Relacional

1 – "÷" (divisão), "|><|" (junção natural)

Listar todos os alunos que foram ou são monitores de prog 1 e prog 2.

DISCIPLINAS ← π <Codigo> (σ <Nome = Prog I II Nome = Prog II (DISCIPLINA))
ALUNOS ← MONITORIA ÷ DISCIPLINAS
RESP ← π <Matricula,Nome > (Alunos |><| (Aluno=Matricula) (ALUNO))

2 - “U” (união), “×” (produto cartesiano)

Quais alunos são monitores, estagiários ou ambos?

MONITORES ← π <Aluno> (σ <DataFim = null> (MONITORIA))
ESTAGIARIOS ← π <Aluno> (σ <DataFim = null> (ESTAGIOLAB))
AMBOS ← MONITORES ∪ ESTAGIARIOS
TEMP1 ← σ<Aluno = Matricula> (AMBOS X ALUNO)
RESP ← π <Matricula, Nome > (TEMP1)

3 - “G” (aggregate functions)

Qual a quantidade de créditos administrados pelo professor "Sergio Lifschitz" na disciplina Projeto Orientado?

TEMP1 ← MatriculaProf G sum (NumCred) (PROJETOORIENTADO)
TEMP2 ← TEMP1 |><|(MatriculaProf = Matricula) PROFESSOR
TEMP3 ← σ <Nome = ‘Sergio Lifschitz’> (TEMP2)
RESP ← π <Nome, NumCred> (TEMP3)

4 - “∩” (interseção)

Quais alunos de pós-graduação já deram monitoria?

MONITORES ← π <Aluno> (MONITORIA)
POS ← π <Matricula> (POSGRAD)
TEMP1 ← MONITORES ∩ POS
TEMP2 ← σ <Aluno = Matricula> (TEMP1 X ALUNO)
TEMP3 ← π <Matricula, Nome> (TEMP2)

5 - “=|><|” (junção externa à esquerda)

Listar todos os alunos dizendo se eles são monitores, de que disciplina, a data do início, e data do fim da monitoria

R1 ← σ <Codigo = CodDisc> (DISCIPLINA X MONITORIA)
R2 ← σ <Nome = NomeDisc> (R1)
R3 ← Aluno =|><| <Matricula = Aluno> R2
Resp ← π <Nome, NomeDisc, DataIni, DataFim> (R3)

6 - Projeção generalizada

R1← G sum <numcred>(PROJETOORIENTADO)
R2← RENAME<R2(numcredtotal)>(R1)
R3← <matricula_prof> G sum <numcred>(PROJETOORIENTADO)

R4← R1 X R2
RF← π <matricula_prof,numcred/numcredtotal>(R4)

IX. Views

Obtem todos os alunos de graduação e seus respectivos nomes:

```
create view alunos_pos as
(select *
 from aluno natural join grad)
```

Obtem todos os alunos de pos graduação e seus respectivos nomes:

```
create view alunos_pos as
(select *
from aluno natural join posgrad)
```

Obtem todos os alunos de doutorado e seus respectivos nomes:

```
create view alunos_doutorado as
(select *
from aluno natural join posgrad
where tipo = 'doutorado')
```

Obtem todos os alunos de mestrado e seus respectivos nomes:

```
create view alunos_mestrado as
(select *
from aluno natural join posgrad
where tipo = 'mestrado')
```

Obtem todos os monitores e seus respectivos nomes:

```
create view monitores as
(select *
```

```
from monitoria, aluno
where aluno = matricula)
```

Obtem todos os monitores da graduação que não fazem engenharia:

```
create view monitores_grad_nao_eng as
(
SELECT *
FROM aluno
WHERE matricula IN ( SELECT G.matricula
FROM grad as G, monitoria as M
WHERE G.matricula = M.aluno
AND G.curso NOT LIKE '%Engenharia%'
AND M.data_fim IS NULL)
ORDER BY nome
where aluno = matricula)
```

Obtem todos os monitores que também estagiam:

```
create view monitor_e_estagiario as
(
SELECT M.matricula, M.nome, E.laboratorio
FROM monitores as M left outer join estagiolab as E on M.aluno = E.aluno
)
```

Obtem todos os pós graduados que iniciaram em 2013:

```
create view posgrad_2013 as
(
SELECT COUNT (*)
FROM posgrad
WHERE PeriodoInicio like '2013.?'
)
```

Obtem a quantidade de créditos de projeto orientado dados pelo professor Sergio Lifschitz:

```
create view cred_sergio as
```

```
(
SELECT SUM(numcred)
FROM projetoorientado INNER JOIN professor ON
projetoorientado.matriculaprof = professor.matricula
WHERE professor.nome = 'Sergio Lifschitz'
)
```

Obtem todos os professores que são orientadores e coorientadores:

```
create view orient_e_coorient as
(
SELECT nome
FROM professor
WHERE matricula IN
(
(SELECT P.matricula
FROM professor as P, posgrad as PG
WHERE PG.coorientador=P.matricula)
INTERSECT
(SELECT P.matricula
FROM professor as P, posgrad as PG
WHERE PG.orientador=P.matricula)
)
)
```

Obtem todos os monitores de programação I:

```
create view monitores_progI as
(
SELECT aluno
FROM monitoria
WHERE codigo = 'INF1005'
EXCEPT(
SELECT aluno
FROM monitoria
WHERE data_fim IS NOT NULL
)
)
```

Obtem todos os alunos que são monitores e/ou estagiam:

```
create view monitor_eou_estagiario as
```

```
(
SELECT aluno
FROM monitoria
UNION (
SELECT aluno
FROM estagiolab
)
)

create view projeto2014 as
(
SELECT P.matricula, P.nome

FROM professor as P

WHERE EXISTS ( SELECT *

FROM projetoorientado as PJO

WHERE PJO.matricula_prof = P.matricula

AND PJO.periodo=2014.1 )
)
```

X. Triggers

Verifica que um aluno de pós graduação de doutorado possui um orientador:

```
CREATE OR REPLACE FUNCTION tem_orientador()

RETURNS TRIGGER AS $tg_tem_orientador$

BEGIN

        IF new.tipo='doutorado' AND new.orientador is NULL THEN

                RAISE EXCEPTION 'Doutorando % deve obrigatoriamente
possuir um orientador.', NEW.matricula;

                RETURN NULL;

        END IF;
```

```
        RETURN NEW;
END;
$tg_tem_orientador$ LANGUAGE plpgsql;

CREATE TRIGGER tg_tem_orientador BEFORE INSERT OR UPDATE ON
posgrad
FOR EACH ROW EXECUTE PROCEDURE tem_orientador();
```

Verifica um aluno de pós graduação de doutorado possui dedicação exclusiva:

```
CREATE OR REPLACE FUNCTION chk_dedicacao()
RETURNS TRIGGER AS $tg_chk_dedicacao$
BEGIN
        IF new.tipo='doutorado' AND new.dedicacao!='exclusiva' THEN
                RAISE EXCEPTION 'Doutorando % deve obrigatoriamente
possuir dedicacao exclusiva.', NEW.matricula;
                RETURN NULL;
        END IF;
        RETURN NEW;
END;
$tg_chk_dedicacao$ LANGUAGE plpgsql;

CREATE TRIGGER tg_chk_dedicacao BEFORE INSERT OR UPDATE ON
posgrad
FOR EACH ROW EXECUTE PROCEDURE chk_dedicacao();
```

Verifica que ao inserir uma monitoria, o aluno não possui outra em andamento:

```
CREATE OR REPLACE FUNCTION chk_monitoria_insert()
RETURNS TRIGGER AS $tg_chk_monitoria_insert$
DECLARE
        aluno monitoria.aluno%TYPE;
BEGIN
        SELECT monitoria.aluno into aluno
        FROM monitoria
        WHERE new.aluno = monitoria.aluno
        AND (monitoria.data_fim>new.data_ini
        OR monitoria.data_fim is NULL);

        IF aluno IS NOT NULL THEN
                RAISE EXCEPTION 'Aluno % já possui/possuia uma monitoria em
andamento.', new.aluno;
                RETURN NULL;
        END IF;
        RETURN NEW;
END;
$tg_chk_monitoria_insert$ LANGUAGE plpgsql;

CREATE TRIGGER tg_chk_monitoria_insert BEFORE INSERT ON monitoria
FOR EACH ROW EXECUTE PROCEDURE chk_monitoria_insert();
```

Verifica que ao atualizar uma monitoria, não aconteça de haver duas monitorias em um mesmo período de tempo:

```
CREATE OR REPLACE FUNCTION chk_monitoria_update()
RETURNS TRIGGER AS $tg_chk_monitoria_update$
```

```
DECLARE
        aluno monitoria.aluno%TYPE;
BEGIN
        SELECT monitoria.aluno into aluno
        FROM monitoria
        WHERE new.aluno = monitoria.aluno
        AND monitoria.data_fim>new.data_ini
        AND monitoria.data_ini!=new.data_ini;

        IF aluno IS NOT NULL THEN
                RAISE EXCEPTION 'Aluno % já possui/possuia uma monitoria em
andamento.', new.aluno;
                RETURN NULL;
        END IF;
        RETURN NEW;
END;
$tg_chk_monitoria_update$ LANGUAGE plpgsql;

CREATE TRIGGER tg_chk_monitoria_update BEFORE UPDATE ON monitoria
FOR EACH ROW EXECUTE PROCEDURE chk_monitoria_update();
```

Verifica que ao inserir um estágio, o aluno não possui outro em andamento:

```
CREATE OR REPLACE FUNCTION chk_estagiolab_insert()
RETURNS TRIGGER AS $tg_chk_estagiolab_insert$
DECLARE
        aluno estagiolab.aluno%TYPE;
```

```
BEGIN
        SELECT estagiolab.aluno into aluno
        FROM estagiolab
        WHERE new.aluno=estagiolab.aluno
        AND (estagiolab.data_fim>new.data_ini
        OR estagiolab.data_fim is NULL);

        IF aluno IS NOT NULL THEN
                RAISE EXCEPTION 'Aluno % já possui/possuia um estagio em
laboratorio em andamento.', new.aluno;
                RETURN NULL;
        END IF;
        RETURN NEW;
END;
$tg_chk_estagiolab_insert$ LANGUAGE plpgsql;

CREATE TRIGGER tg_chk_estagiolab_insert BEFORE INSERT ON estagiolab
FOR EACH ROW EXECUTE PROCEDURE chk_estagiolab_insert();
```

Verifica que ao atualizar um estágio, , não aconteça de haver dois estágios em um mesmo período de tempo:

```
CREATE OR REPLACE FUNCTION chk_estagiolab_update()
RETURNS TRIGGER AS $tg_chk_estagiolab_update$
DECLARE
        aluno estagiolab.aluno%TYPE;
BEGIN
```

```
        SELECT estagiolab.aluno into aluno
        FROM estagiolab
        WHERE new.aluno=estagiolab.aluno
        AND estagiolab.data_fim>new.data_ini
        AND estagiolab.data_ini!=new.data_ini;

        IF aluno IS NOT NULL THEN
                RAISE EXCEPTION 'Aluno % já possui/possuia um estagio em
laboratorio em andamento.', new.aluno;
                RETURN NULL;
        END IF;
        RETURN NEW;
END;
$tg_chk_estagiolab_update$ LANGUAGE plpgsql;

CREATE TRIGGER tg_chk_estagiolab_update BEFORE UPDATE ON
estagiolab
FOR EACH ROW EXECUTE PROCEDURE chk_estagiolab_update();
```

Verifica que um aluno de pós-graduação brasileiro possui um número de identidade ,um órgão emissor e uma data de emissão:

```
CREATE OR REPLACE FUNCTION chk_brasileiro()
RETURNS TRIGGER AS $tg_chk_brasileiro$
DECLARE
        nacionalidade posgrad.nacionalidade%TYPE;
BEGIN
        SELECT posgrad.nacionalidade INTO nacionalidade
```

```
        FROM posgrad
        WHERE new.matricula=posgrad.matricula;

        IF lower(new.nacionalidade) like 'brasileir_' OR lower(nacionalidade) like
'brasileir_' THEN
                IF new.id IS NULL THEN
                        RAISE EXCEPTION 'Aluno % brasileiro precisa de um
número de id.', new.matricula;
                        RETURN NULL;
                END IF;
                IF new.orgao_emissor IS NULL THEN
                        RAISE EXCEPTION 'Aluno % brasileiro precisa de um
órgão emissor.', new.matricula;
                        RETURN NULL;
                END IF;
                IF new.data_emissao IS NULL THEN
                        RAISE EXCEPTION 'Aluno % brasileiro precisa de uma
data de emissão.', new.matricula;
                        RETURN NULL;
                END IF;
        END IF;
        RETURN NEW;
END;
$tg_chk_brasileiro$ LANGUAGE plpgsql;

CREATE TRIGGER tg_chk_brasileiro BEFORE INSERT OR UPDATE ON
posgrad
FOR EACH ROW EXECUTE PROCEDURE chk_brasileiro();
```

Verifica que um aluno de pós-graduação estrangeiro possui um número de passaporte ,um país do passaporte e uma data de validade do passaporte:

```
CREATE OR REPLACE FUNCTION chk_estrangeiro()

RETURNS TRIGGER AS $tg_chk_estrangeiro$

BEGIN

        IF lower(new.nacionalidade) not like 'brasileir_' THEN

                IF new.num_passaporte IS NULL THEN

                        RAISE EXCEPTION 'Aluno % estrangeiro precisa de um
número de passaporte.', new.matricula;

                        RETURN NULL;

                END IF;

                IF new.pais_passaporte IS NULL THEN

                        RAISE EXCEPTION 'Aluno % estrangeiro precisa de um
país do passaporte.', new.matricula;

                        RETURN NULL;

                END IF;

                IF new.validade_passaporte IS NULL THEN

                        RAISE EXCEPTION 'Aluno % estrangeiro precisa de uma
data de validade do passaporte.', new.matricula;

                        RETURN NULL;

                END IF;

        END IF;

        RETURN NEW;

END;

$tg_chk_estrangeiro$ LANGUAGE plpgsql;


CREATE TRIGGER tg_chk_estrangeiro BEFORE INSERT OR UPDATE ON
posgrad
```

```
FOR EACH ROW EXECUTE PROCEDURE chk_estrangeiro();
```

XI. Procedimentos

Altera as “data_fim” na tabela monitoria que estão como NULL para a data atual:

```
CREATE OR REPLACE FUNCTION finaliza_monitorias()
RETURNS void AS $$
BEGIN
        UPDATE monitoria
        SET data_fim=current_date
        WHERE data_fim IS NULL;

END;
$$ LANGUAGE plpgsql;
```

Exibe quantos alunos de pós graduação estão em cada status da matrícula:

```
CREATE OR REPLACE FUNCTION relatorio_status()
RETURNS void as $$
DECLARE
        status varchar(12);
        num int;
        tuplas cursor is
                SELECT posgrad.status_matricula, count(*)
                FROM posgrad
                GROUP BY posgrad.status_matricula;
```

```
BEGIN
        OPEN tuplas;
        RAISE INFO 'STATUS DA MATRICULA - QUANTIDADE';
        loop
                fetch tuplas into status, num;
                if not found then exit;
                end if;
                RAISE INFO '% - %', status, num;
        end loop;
        CLOSE tuplas;
END;
$$ LANGUAGE plpgsql;
```

XII. Funções

Obtem o período a partir da data:

```
CREATE OR REPLACE FUNCTION obtem_periodo(data_in date)
RETURNS numeric(5) AS $$
DECLARE
        mes smallint;
        ano smallint;
        resultado numeric(4,1);

BEGIN
        SELECT EXTRACT(YEAR FROM data_in) into ano;
```

```
        SELECT EXTRACT(MONTH FROM data_in) into mes;


        IF (mes <=6)THEN
                resultado:=0.1;
        ELSE
                resultado:=0.2;
        END IF;
        RETURN resultado+ano;
END;
$$ LANGUAGE plpgsql;
```

Obtem todos os alunos de pós graduação que se matricularam em um período específico:

```
CREATE OR REPLACE FUNCTION obtem_matriculados (periodo
posgrad.periodoinicio%TYPE , tipopg posgrad.tipo%TYPE)
RETURNS bigint as $$
DECLARE
        numero int;
BEGIN
        SELECT COUNT (*) into numero
        FROM posgrad
        WHERE posgrad.periodoinicio = periodo AND posgrad.tipo = tipopg;
        RETURN numero;
END;
$$ LANGUAGE plpgsql;
```

XIII. Índices

Para o nosso banco de dados, prevemos que a maior parte das consultas seram feitas sobre matriculas e códigos de discilinas, os quais já são indexados implicitamente por serem chaves primárias. Na tabela posgrad, o CPF, link do currículo Lattes e login, também já possuem índices por serem UNIQUE. Analisamos que outras consultas que podem vir a ser relevantes são aquelas em cima da situação ou sttatus da matrícula de um pós-graduando e sobre as disciplinas de monitoria, assim criamos também os seguintes índices:

```
CREATE INDEX INX_situacao ON posgrad (situacao)

CREATE INDEX INX_status ON posgrad (status_matricula)

CREATE INDEX INX_discip_monitoria ON monitoria (codigo)
```