Modificações feitas no diagrama desde o último envio:

Agora a relação entre Aluno e Vaga “SE QUALIFICA” passou a ser “SE INSCREVE”, e a verificação se um aluno está ou não qualificado para uma vaga é feita com base nas habilidades desenvolvidas pelo aluno ao completar um curso.

Vaga passou a ter um nome.

Habilidade passou a ser uma entidade própria, evitando problemas possíveis de acontecer com o caso anterior com Vaga e Curso tendo atributos multivariados para as habilidades. Agora descobrimos a habilidade requisitada por uma vaga e a habilidade oferecida por um curso através de uma relação utilizando as chaves estrangeiras.

Consultas SQL

Seleciona o e-mail, o nome, a data de nascimento, o status do plano, o número de vagas no qual cada aluno se inscreveu e a idade para os alunos inscritos em uma vaga.

```
def get_alunos(self, nome: str = "", status_plano: str = "", email: str = ""):

        current_date = datetime.now().date()

        query = f"""

    SELECT A.email, A.nome, TO_CHAR(A.data_nascimento, 'YYYY-MM-DD') AS
    data_nascimento, A.status_plano, COUNT(S.id_vaga) AS numero_inscricoes,
    DATE_PART('year', AGE('{current_date}', A.data_nascimento))::INT AS idade

    FROM Aluno A LEFT JOIN Se_Inscreve S ON A.email = S.email_aluno

    """
```

```
    filtros = []
        if nome:
            filtros.append(f"LOWER(A.nome) LIKE '%{nome.lower()}%'")
        if status_plano:
            filtros.append(f"LOWER(A.status_plano) = '{status_plano.lower()}'")
        if email:
            filtros.append(f"LOWER(A.email) LIKE '%{email.lower()}%'")
        if filtros:
            query += " WHERE " + " AND ".join(filtros)
          query += """
          GROUP BY A.email, A.nome, A.data_nascimento, A.status_plano
          ORDER BY A.nome ASC
"""
return self.db.execute_select_all(query)
```

Lista o número de alunos com plano ativo.

```
SELECT COUNT(*) as count FROM Aluno
WHERE status_plano = 'ativo'"
```

Lista o número de alunos com plano inativo.

```
SELECT COUNT(*) as count FROM Aluno

WHERE status_plano = 'inativo'"
```

Lista o nome, a descrição, a duração, o nível, a data de lançamento, o número de alunos que concluiram e as habilidades oferecidas e seu nível para um curso.

```
def get_cursos(self, nome: str = "", duracao: int = None, nivel: str = ""):

    SELECT C.nome, C.descricao, C.duracao, C.nivel,
TO_CHAR(C.data_lancamento, 'DD/MM/YYYY') AS data_lancamento,
COUNT(E.email_aluno) AS numero_alunos_concluidos, STRING_AGG(DISTINCT
H.nome || ': ' || H.nivel, ', ') AS habilidades

    FROM Curso C LEFT JOIN Estuda E ON C.nome = E.nome_curso AND
E.data_conclusao IS NOT NULL LEFT JOIN habilidade_curso HC ON C.nome =
HC.nome_curso LEFT JOIN Habilidade H ON HC.id_habilidade = H.id

    filtros = []

    if nome:

        filtros.append(f"LOWER(C.nome) LIKE '%{nome.lower()}%'")

    if duracao is not None:

        filtros.append(f"C.duracao = {duracao}")



    if nivel:

        filtros.append(f"LOWER(C.nivel) = '{nivel.lower()}'")

    if filtros:

    query += " WHERE " + " AND ".join(filtros)

    query += """
```

```
    GROUP BY C.nome, C.descricao, C.duracao, C.nivel, C.data_lancamento
    """
    query += " ORDER BY C.nome ASC"
    return self.db.execute_select_all(query)
```

Lista o número de curso disponiblizados.

```
SELECT COUNT(*)
FROM Curso
```

Lista o nome, a localização, o setor e o número de vagas oferecidas das empresas.

```
def get_empresas(self, nome: str = "", setor: str = "", localizacao: str = ""):
    query = """
    SELECT E.nome AS empresa_nome, E.localizacao, E.setor, COUNT(V.id) AS numero_vagas
    FROM Empresa E LEFT JOIN Vaga V ON E.nome = V.empresa
    """
    filtros = []
    if nome:
        filtros.append(f"LOWER(E.nome) LIKE '%{nome.lower()}%'")
    if setor:
        filtros.append(f"LOWER(E.setor) LIKE '%{setor.lower()}%'")
    if localizacao:
        filtros.append(f"LOWER(E.localizacao) LIKE '%{localizacao.lower()}%'")
```

```
    if filtros:
        query += " WHERE " + " AND ".join(filtros)
    query += """
    GROUP BY
        E.nome, E.localizacao, E.setor
    ORDER BY
        E.nome ASC
    """
    return self.db.execute_select_all(query)
```

Lista o número de empresas parceiras.

```
SELECT COUNT(*)
FROM empresa
```

Lista o e-mail, o nome e a especialização dos professores.

```
def get_professores(self, nome: str = "", especializacao: str = "", email: str = ""):
    query = "SELECT *
    FROM Professor"
    filtros = []
    if nome:
        filtros.append(f"LOWER(nome) LIKE '%{nome.lower()}%'")
    if especializacao:
        filtros.append(f"LOWER(especializacao) LIKE '%{especializacao.lower()}%'")
    if email:
```

```
        filtros.append(f"LOWER(email) LIKE '%{email.lower()}%'")
    if filtros:
        query += " WHERE " + " AND ".join(filtros)
    query += " ORDER BY nome ASC"
    return self.db.execute_select_all(query)
```

Lista o número de professores cadastrados.

```
SELECT COUNT(*) as count
FROM professor
```

Lista o id, o nome, a descrição, a localização, o nome que empresa que oferece, o número de alunos inscritos, as habilidades necessárias e os seus respectivos níveis para as vagas.

```
def get_vagas(self, id: int = None, nome: str = "", empresa: str = "", requisitos: str = "", ordenar_por: str = "numero_inscritos", ordenar_ordem: str = "DESC"):
    query = """
    SELECT V.id, V.nome AS vaga_nome, V.descricao, E.localizacao, E.nome AS empresa_nome, COUNT(DISTINCT S.email_aluno) AS numero_inscritos, STRING_AGG(DISTINCT H.nome || ': ' || H.nivel, ', ') AS requisitos
    FROM Vaga V LEFT JOIN Empresa E ON V.Empresa = E.nome LEFT JOIN se_inscreve S ON V.id = S.id_vaga LEFT JOIN Habilidade_Vaga HV ON V.id = HV.id_vaga LEFT JOIN Habilidade H ON HV.id_habilidade = H.id
    """
    filtros = []
```

```
        if id is not None:
            filtros.append(f"V.id = {id}")
        if nome:
            filtros.append(f"LOWER(V.nome) LIKE '%{nome.lower()}%'")
        if empresa:
            filtros.append(f"LOWER(E.nome) LIKE '%{empresa.lower()}%'")
        if requisitos:
            filtros.append(f"""
            EXISTS (
                SELECT 1
                FROM Habilidade_Vaga HV JOIN Habilidade H ON HV.id_habilidade = H.id
                WHERE HV.id_vaga = V.id AND LOWER(H.nome) LIKE
'%{requisitos.lower()}%' )
            """)
        if filtros:
            query += " WHERE " + " AND ".join(filtros)
        query += """
        GROUP BY
            V.id, V.nome, V.descricao, E.localizacao, E.nome
        ORDER BY
            {ordenar_por} {ordenar_ordem}
        """.format(ordenar_por=ordenar_por, ordenar_ordem=ordenar_ordem)
        return self.db.execute_select_all(query)
```

Lista o número vagas disponíveis.

```
SELECT COUNT(*)

FROM Vaga
```

Lista os alunos inscritos para uma vaga, com os atributos úteis para essa busca.

```
def get_vagas_inscritas_por_aluno(self, email_aluno: str, vaga_nome: str = "", empresa_nome: str = "", localizacao: str = "", requisitos: str = "", ordenar_por: str = "numero_inscritos", ordenar_ordem: str = "DESC"):

    aluno_query = f"""

    SELECT nome AS aluno_nome

    FROM Aluno

    WHERE email = '{email_aluno}'

    """

    aluno_nome = self.db.execute_select_one(aluno_query)['aluno_nome']

    vagas_query = f"""

    SELECT V.id AS vaga_id, V.nome AS vaga_nome, E.nome AS empresa_nome, E.localizacao, (SELECT COUNT(*)

                FROM se_inscreve si_sub

                WHERE si_sub.id_vaga = v.id

                ) AS numero_inscritos, json_agg(DISTINCT (H.nome || ': ' || H.nivel)

     ) AS requisitos

    FROM Se_Inscreve S INNER JOIN Vaga V ON S.id_vaga = V.id LEFT JOIN Empresa E ON V.empresa = E.nome LEFT JOIN Habilidade_Vaga HV ON V.id = HV.id_vaga LEFT JOIN Habilidade H ON HV.id_habilidade = H.id
```

```
WHERE S.email_aluno = '{email_aluno}'
"""
if vaga_nome:
    vagas_query += f" AND LOWER(V.nome) LIKE '%{vaga_nome.lower()}%'"
if empresa_nome:
    vagas_query += f" AND LOWER(e.nome) LIKE '%{empresa_nome.lower()}%'"
if localizacao:
    vagas_query += f" AND LOWER(e.localizacao) LIKE '%{localizacao.lower()}%'"
if requisitos:
    vagas_query += f"""
    AND EXISTS (
        SELECT 1
        FROM habilidade_vaga hv
        JOIN habilidade h ON hv.id_habilidade = h.id
        WHERE hv.id_vaga = v.id
        AND LOWER(h.nome) LIKE '%{requisitos.lower()}%'
    )
    """
vagas_query += f"""
GROUP BY V.id, V.nome, E.nome, E.localizacao
ORDER BY {ordenar_por} {ordenar_ordem}
"""
vagas_inscritas = self.db.execute_select_all(vagas_query)
habilidades_query = f"""
SELECT STRING_AGG(h.nome || ': ' || h.nivel, ', ') AS habilidade
```

```
        FROM Estuda E INNER JOIN Habilidade_Curso HC ON E.nome_curso =
HC.nome_curso INNER JOIN Habilidade H ON HC.id_habilidade = H.id
        WHERE E.email_aluno = '{email_aluno}'
        """
        habilidades_aluno = self.db.execute_select_all(habilidades_query)
        return {
            "aluno_nome": aluno_nome,
            "habilidades_aluno": [{"habilidade": habilidades_aluno[0]['habilidade']}],
            "vagas_inscritas": vagas_inscritas if vagas_inscritas else []
        }
```

Lista o nome e o nível das habilidades de um curso.

```
SELECT H.nome, H.nivel
FROM Habilidade_Curso HC
JOIN Habilidade H ON HC.id_habilidade = H.id
WHERE HC.nome_curso = '{nome_curso}'
```

Lista o nome, o e-mail, a data de conclusão e a nota de um aluno inscrito em um dado curso.

```
SELECT A.nome AS aluno_nome, A.email AS aluno_email,
TO_CHAR(E.data_conclusao, 'DD/MM/YYYY') AS data_conclusao, E.nota
FROM Estuda E INNER JOIN Aluno A ON E.email_aluno = A.email
WHERE E.nome_curso = '{nome_curso}' AND E.data_conclusao IS NOT NULL
```

Lista o nome e o nível de um habilidade requisitada por uma vaga com base no ID da vaga.

```
SELECT H.nome, H.nivel

FROM Habilidade_Vaga HV JOIN Habilidade H ON HV.id_habilidade = H.id

WHERE HV.id_vaga = {id_vaga}
```

Lista o e-mail, o nome, a data de nascimento e o status do plano de um aluno inscrito em uma vaga com base no ID da vaga.

```
SELECT A.email, A.nome, A.data_nascimento, A.status_plano

FROM Aluno A INNER JOIN Se_Inscreve S ON A.email = S.email_aluno

WHERE S.id_vaga = {id_vaga}
```

Lista o nome, o nível e a duração dos cursos feitos por um aluno com base no e-mail do aluno.

```
SELECT C.nome, C.nivel, C.duracao

FROM Curso C INNER JOIN Estuda E ON C.nome = E.nome_curso

WHERE E.email_aluno = '{aluno["email"]}'
```

Lista o nome e o nível das habilidades oferecidas pelos cursos realizados por um aluno com base no seu e-mail.

```
SELECT DISTINCT H.nome, H.nivel

FROM Habilidade_Curso HC JOIN Habilidade H ON HC.id_habilidade = H.id INNER JOIN Estuda E ON HC.nome_curso = E.nome_curso

WHERE E.email_aluno = '{aluno["email"]}'
```

Lista o nome, o nível, a duração, a data de lançamento e as habilidades oferecidas dos cursos sugeridos para um aluno para que ele se qualifique a uma vaga.

```
SELECT subquery.nome, subquery.nivel, subquery.duracao, subquery.data_lancamento,  subquery.habilidade

FROM ( SELECT DISTINCT c.nome, c.nivel, c.duracao, TO_CHAR(c.data_lancamento, 'DD/MM/YYYY') AS data_lancamento, H.nome AS habilidade

        FROM curso c

        INNER JOIN habilidade_curso hc ON c.nome = hc.nome_curso

        INNER JOIN habilidade h ON hc.id_habilidade = h.id

        WHERE h.nome IN ('{habilidades_str}')
```

Lista o e-mail, nome e status plano de um aluno de acordo com o email.

```
SELECT email, nome, status_plano

FROM Aluno

WHERE LOWER(email) = LOWER('{email_aluno}')
```

Lista o número de curso concluídos por um aluno.

```
SELECT COUNT(*) AS cursos_concluidos

FROM Estuda

WHERE LOWER(email_aluno) = LOWER('{email_aluno}') AND data_conclusao IS NOT NULL
```

Lista o número de vagas que um aluno se inscreveu.

```
SELECT COUNT(*) AS vagas_inscritas

FROM Se_Inscreve

WHERE LOWER(email_aluno) = LOWER('{email_aluno}')
```

Lista as habilidades de um aluno.

SELECT DISTINCT LOWER(h.nome || ': ' || h.nivel) AS habilidade

FROM Estuda E INNER JOIN Habilidade_Curso HC ON E.nome_curso = HC.nome_curso INNER JOIN Habilidade H ON HC.id_habilidade = H.id

WHERE LOWER(E.email_aluno) = LOWER('{email_aluno}') AND E.data_conclusao IS NOT NULL

Lista o nome, o nível, a data de conclusão e a nota dos cursos completados for um aluno.

SELECT C.nome, C.nivel, TO_CHAR(e.data_conclusao, 'DD/MM/YYYY') AS data_conclusao, E.nota

FROM Curso C INNER JOIN Estuda E ON C.nome = E.nome_curso

WHERE LOWER(E.email_aluno) = LOWER('{email_aluno}') AND E.data_conclusao IS NOT NULL

if filtro_nome_curso:

query_detalhes_cursos_concluidos += f" AND LOWER(c.nome) LIKE '%{filtro_nome_curso}%'"

if filtro_habilidade:

query_detalhes_cursos_concluidos += f""" AND LOWER(c.nome) IN (

SELECT LOWER(nome_curso)

FROM Habilidade_Curso HC

INNER JOIN Habilidade H ON HC.id_habilidade = H.id

WHERE LOWER(H.nome) LIKE '%{filtro_habilidade}%')

"""

if filtro_nivel:

query_detalhes_cursos_concluidos += f" AND LOWER(c.nivel) LIKE '%{filtro_nivel}%'"

query_detalhes_cursos_concluidos += f" ORDER BY e.nota {ordem_nota}"

Lista o nome e o nível das habilidades oferecidas por um curso com base no nome do curso.

SELECT LOWER(H.nome || ': ' || H.nivel) AS habilidade

FROM Habilidade_Curso HC INNER JOIN Habilidade H ON HC.id_habilidade = H.id

WHERE LOWER(HC.nome_curso) = LOWER('{curso["nome"]}')