SISTEMA DE GERENCIAMENTO DE VAGAS (SGV) PARA HOSPITAL DAS CLÍNICAS DA UFPE

Equipe:

Felipe Aguiar
Roberta Baudel Francisco
Maxwell Volnei
Pedro de Souza
José Lamartine

DESCRIÇÃO DE MINIMUNDO

Este minimundo define a estrutura do SGV, permitindo a gestão eficiente e organizada das atividades dos estudantes e residentes no Hospital das Clínicas da UFPE, garantindo um uso otimizado dos recursos e uma melhor gestão e coordenação das operações acadêmicas e assistenciais.

1. Descrição do mundo real a ser modelado

O Hospital da Clínicas (HC) oferece uma ampla gama de serviços de alta complexidade e mantém uma diversidade de atividades práticas de ensino e pesquisa, incluindo programas de residências em saúde, pós-graduação e atividades de extensão.

Os estudantes e residentes envolvidos nas atividades são provenientes de diversas áreas da saúde, como Medicina, Enfermagem, Fisioterapia, Nutrição, e outras áreas afins. Isso reflete a abrangência e a diversidade dos programas oferecidos pela instituição.

A gestão eficaz das vagas no hospital é essencial para garantir uma distribuição equitativa de oportunidades de aprendizado e prática clínica, além de assegurar o cumprimento dos padrões de qualidade exigidos pela instituição e pelos órgãos reguladores da saúde.

A gestão e coordenação das atividades dos graduandos e pós-graduandos (residentes) dentro do hospital é um aspecto de grande relevância para o desenvolvimento profissional e acadêmico, contribuindo diretamente para a excelência no atendimento aos pacientes e na produção de conhecimento científico na área da saúde.

2. Descrição dos objetivos

O SGV irá oferecer ferramentas e recursos para uma gestão eficiente e organizada das atividades dos graduandos e residentes, promovendo assim um ambiente de aprendizado e prática clínica de alta qualidade no Hospital das Clínicas da UFPE. O HC tem a necessidade de ter um banco de dados que

contemple todas as informações relevantes para o controle e a gestão de suas atividades de prática de ensino e pesquisa.

O SGV será desenvolvido para o Hospital das Clínicas da UFPE, visando centralizar e otimizar a gestão de estudantes em diversas atividades, como estágios, práticas, participação em ligas acadêmicas, extensão e coordenação de rodízios de residentes.

3. Descrição informal dos dados:

3.1. Entidades

Colaborador: São pessoas que trabalham no Hospital das Clínicas da UFPE.

Atributos:

- Matrícula
- CPF
- Nome
- Lotação

Enfermeiro: É uma especialização da entidade Colaborador que ocupa o cargo de enfermeiro.

- COREN

Médico: É uma especialização da entidade Colaborador que ocupa o cargo de Médico.

- CRM

Estudante: Representam os graduandos e os pós-graduandos (residentes). Os cursos de graduação ou Programas de Residências podem ser oferecidos pela instituição UFPE ou conveniadas.

Atributos:

- CPF
- CNS
- Nome
- Obs
- Período
- Curso
- CBO
- Programa
- Registro_conselho
- Tipo_estudante

Instituição: São as entidades educacionais ou de saúde às quais os graduandos/ residentes estão associados.

Atributos

- CNPJ
- Nome
- Telefone - *Multivalorado*
- Endereço (Rua, Bairro, Cidade, Estado, CEP) - *Composto*

Serviço: São os serviços oferecidos pelo Hospital das Clínicas da UFPE.

Atributos:

- Código
- Nome
- Capacidade
- Medico
- Enfermeiro

Vaga: Representam as oportunidades disponíveis nos serviços para os estudantes (graduandos e pós-graduandos) de diferentes cursos/programas.

Atributos:

- Codigo
- Vigencia_vaga (data_inicio, data_fim) - *Composto*

3.2. Relacionamentos

Vinculado

É um relacionamento binário entre Estudante e Instituição. Indica que um estudante pode estar vinculado a apenas uma instituição e que uma instituição pode ter vários estudantes vinculados à ela.

Cardinalidade: N:1

Ocupa

É um relacionamento binário entre Estudante e Vaga. Indica que um estudante pode estar ocupando diversas vagas e uma vaga só pode ser ocupada por um estudante.

Cardinalidade: 1:N

Oferece

É um relacionamento binário entre Vaga e Serviço. Indica que uma vaga só pode ser oferecida por um serviço e que um serviço pode oferecer diversas vagas.

Cardinalidade: N:1

Trabalha

É um relacionamento binário entre Colaborador e Serviço. Indica que um colaborador trabalha em apenas um serviço e que um serviço pode ter diversos colaboradores trabalhando nele.

Cardinalidade: N:1

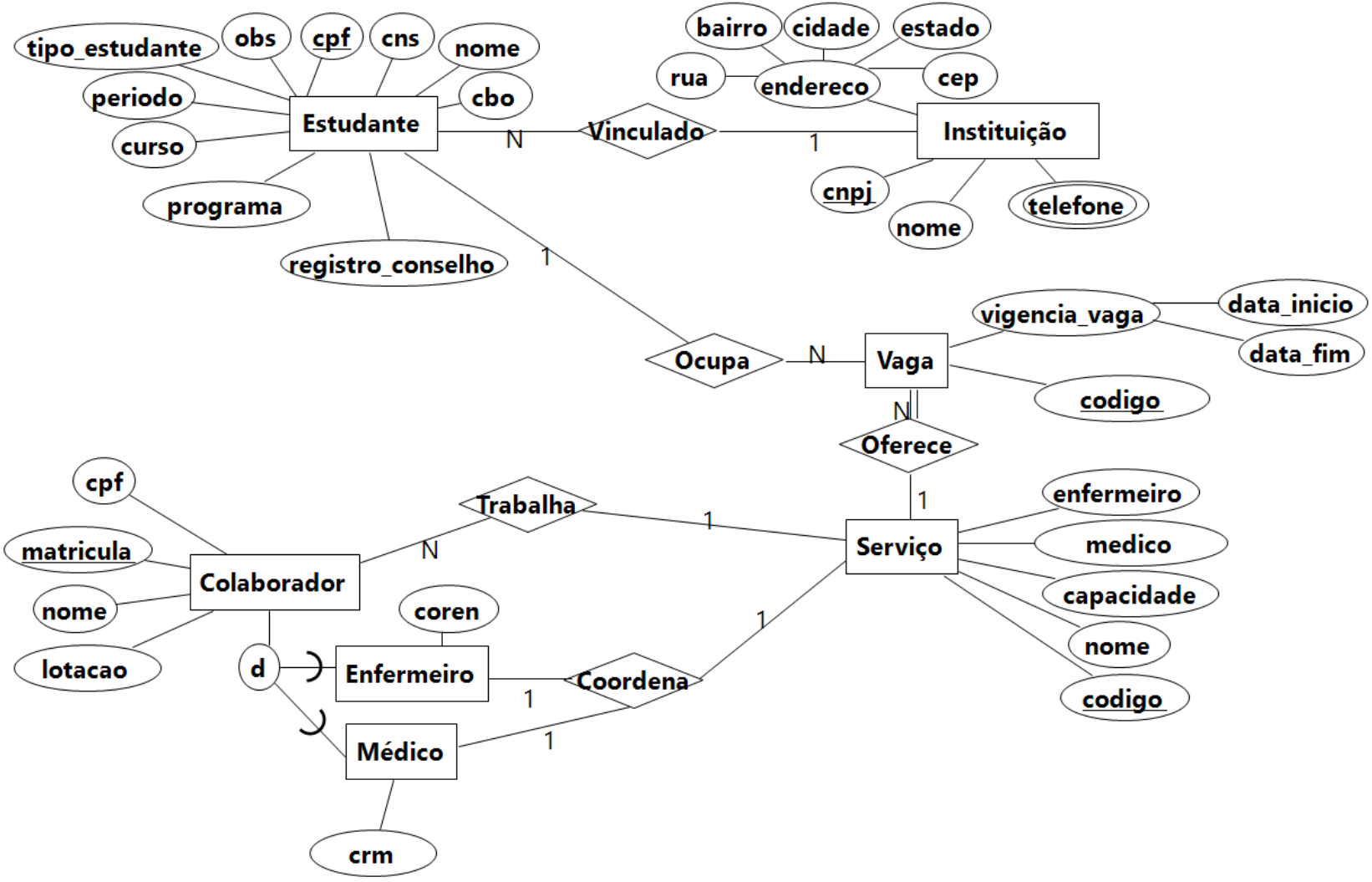
Coordena

É um relacionamento ternário entre Médico, Enfermeiro e Serviço. Indica que um médico e um enfermeiro coordenam um serviço e um serviço é coordenado por um médico e um enfermeiro.

Cardinalidade: 1:1:1

ESQUEMA CONCEITUAL

Diagrama de Entidade-Relacionamento (DER)



CRIAÇÃO DOS TIPOS

```
CREATE OR REPLACE TYPE tp_endereco AS OBJECT (
   rua VARCHAR2(100),
   bairro VARCHAR2(50),
   cidade VARCHAR2(50),
   uf VARCHAR2(2),
   cep VARCHAR2(9)
);

CREATE OR REPLACE TYPE tp_telefone AS OBJECT (
   cod_area VARCHAR2(2),
   numero VARCHAR2(9)
);

CREATE OR REPLACE TYPE tp_telefones AS VARRAY(3) OF tp_telefone;

CREATE OR REPLACE TYPE  tp_instituicao AS OBJECT (
   endereco tp_endereco,
   telefone tp_telefones,
   nome VARCHAR2(255),
   cnpj VARCHAR2(14)
);

CREATE OR REPLACE TYPE tp_colaborador AS OBJECT (
   matricula VARCHAR2(11),
   nome VARCHAR2(255),
   cpf VARCHAR2(11),
   lotacao VARCHAR2(50)
) NOT FINAL;

CREATE TYPE tp_medico UNDER tp_colaborador (
   crm VARCHAR2(8)
);

CREATE TYPE tp_enfermeiro UNDER tp_colaborador (
   coren VARCHAR2(9)
);

CREATE OR REPLACE TYPE tp_servico AS OBJECT (
```

```
    medico ref tp_medico,
    enfermeiro ref tp_enfermeiro,
    colaborador ref tp_colaborador,
    codigo VARCHAR2(5),
    nome VARCHAR2(50),
    capacidade NUMBER(3)
);

ALTER TYPE tp_colaborador add attribute (servico ref tp_servico) CASCADE;

CREATE OR REPLACE TYPE tp_estudante AS OBJECT (
    tipo_estudante VARCHAR2(20),
    instituicao REF tp_instituicao,
    nome VARCHAR2(255),
    cpf VARCHAR2(11),
    observacao VARCHAR2(200),
    cns VARCHAR2(255),
    periodo NUMBER(2),
    curso VARCHAR2(100),
    cbo VARCHAR2(6),
    programa VARCHAR2(100),
    registro_conselho VARCHAR2(9),
 )

CREATE OR REPLACE TYPE tp_vigencia_vaga AS OBJECT (
    data_inicio DATE,
    data_fim DATE
);

CREATE OR REPLACE TYPE tp_vaga AS OBJECT (
    estudante ref tp_estudante,
    servico ref tp_servico,
    vigencia tp_vigencia_vaga,
    codigo NUMBER(5,0),
    indica_ativa NUMBER(1)
);
```

CRIAÇÃO DAS TABELAS

```
CREATE TABLE tb_instituicao OF tp_instituicao (
    CONSTRAINT tb_instituicao_nome_not_null CHECK (nome IS NOT NULL),
    CONSTRAINT tb_instituicao_cnpj_check CHECK (LENGTH(cnpj) = 14),
    CONSTRAINT tb_instituicao_pkey PRIMARY KEY (cnpj)
);
```

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ENDERECO	TP_ENDERECO	Yes	(null)	1	(null)
2	TELEFONE	TP_TELEFONES	Yes	(null)	2	(null)
3	NOME	VARCHAR2(255 BYTE)	Yes	(null)	3	(null)
4	CNPJ	VARCHAR2(14 BYTE)	No	(null)	4	(null)

```
 CREATE TABLE tb_medico OF tp_medico (
    CONSTRAINT tb_medico_nome_not_null CHECK (nome IS NOT NULL),
    CONSTRAINT tb_medico_cpf_not_null CHECK (cpf IS NOT NULL),
    CONSTRAINT tb_medico_crm_not_null CHECK (crm IS NOT NULL),
    CONSTRAINT tb_medico_crm_unique UNIQUE (crm),
    CONSTRAINT tb_medico_cpf_unique UNIQUE (cpf),
    CONSTRAINT tb_medico_pkey PRIMARY KEY (matricula)
);
```

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	MATRICULA	VARCHAR2(11 BYTE)	No	(null)	1	(null)
2	NOME	VARCHAR2(255 BYTE)	Yes	(null)	2	(null)
3	CPF	VARCHAR2(11 BYTE)	Yes	(null)	3	(null)
4	LOTACAO	VARCHAR2(50 BYTE)	Yes	(null)	4	(null)
5	SERVICO	TP_SERVICO	Yes	(null)	5	(null)
6	CRM	VARCHAR2(8 BYTE)	Yes	(null)	6	(null)

```
 CREATE TABLE tb_enfermeiro OF tp_enfermeiro (
    CONSTRAINT tb_enfermeiro_nome_not_null CHECK (nome IS NOT NULL),
    CONSTRAINT tb_enfermeiro_cpf_not_null CHECK (cpf IS NOT NULL),
    CONSTRAINT tb_enfermeiro_coren_not_null CHECK (coren IS NOT NULL),
    CONSTRAINT tb_enfermeiro_coren_unique UNIQUE (coren),
    CONSTRAINT tb_enfermeiro_cpf_unique UNIQUE (cpf),
    CONSTRAINT tb_enfermeiro_pkey PRIMARY KEY (matricula)
);
```

1	MATRICULA	VARCHAR2(11 BYTE)	No	(null)	1	(null)
2	NOME	VARCHAR2(255 BYTE)	Yes	(null)	2	(null)
3	CPF	VARCHAR2(11 BYTE)	Yes	(null)	3	(null)
4	LOTACAO	VARCHAR2(50 BYTE)	Yes	(null)	4	(null)
5	SERVICO	TP_SERVICO	Yes	(null)	5	(null)
6	COREN	VARCHAR2(9 BYTE)	Yes	(null)	6	(null)

```
 CREATE TABLE tb_colaborador OF tp_colaborador (
    CONSTRAINT tb_colaborador_nome_not_null CHECK (nome IS NOT NULL),
    CONSTRAINT tb_colaborador_cpf_not_null CHECK (cpf IS NOT NULL),
    CONSTRAINT tb_colaborador_cpf_unique UNIQUE (cpf),
    CONSTRAINT tb_colaborador_pkey PRIMARY KEY (matricula)
);
```

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID
1	MATRICULA	VARCHAR2(11 BYTE)	No	(null)	1
2	NOME	VARCHAR2(255 BYTE)	Yes	(null)	2
3	CPF	VARCHAR2(11 BYTE)	Yes	(null)	3
4	LOTACAO	VARCHAR2(50 BYTE)	Yes	(null)	4
5	SERVICO	TP_SERVICO	Yes	(null)	5

```
CREATE TABLE tb_servico OF tp_servico (
   CONSTRAINT tb_servico_medico_responsavel_not_null CHECK (medico IS NOT
NULL),
   CONSTRAINT tb_servico_enfermeiro_responsavel_not_null CHECK (enfermeiro IS
NOT NULL),
   CONSTRAINT tb_servico_nome_not_null CHECK (nome IS NOT NULL),
   CONSTRAINT tb_servico_pkey PRIMARY KEY (codigo),
   medico WITH ROWID REFERENCES tb_medico,
   enfermeiro WITH ROWID REFERENCES tb_enfermeiro,
   colaborador WITH ROWID REFERENCES tb_colaborador
);
```

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	MEDICO	TP_MEDICO	Yes	(null)	1	(null)
2	ENFERMEIRO	TP_ENFERMEIRO	Yes	(null)	2	(null)
3	CODIGO	VARCHAR2(5 BYTE)	No	(null)	3	(null)
4	NOME	VARCHAR2(50 BYTE)	Yes	(null)	4	(null)
5	CAPACIDADE	NUMBER(3,0)	Yes	(null)	5	(null)

```
CREATE TABLE tb_estudante OF tp_estudante (
    CONSTRAINT tb_estudante2_cpf_check CHECK (LENGTH(cpf) = 11),
```

```
    CONSTRAINT tb_estudante2_cns_check CHECK (LENGTH(cns) = 0 OR
LENGTH(cns) = 15),
    CONSTRAINT tb_estudante2_periodo_check CHECK (periodo > 0 AND periodo <
100),
    CONSTRAINT tb_estudante2_cns_unique UNIQUE (cns),
    CONSTRAINT tb_estudante2_pkey PRIMARY KEY (cpf),
    CONSTRAINT tb_estudante2_registro_conselho_check CHECK (registro_conselho
IS NULL OR LENGTH(registro_conselho) IN (0, 8, 9)),
    CONSTRAINT tb_estudante2_registro_conselho_unique UNIQUE
(registro_conselho),
    CONSTRAINT tb_estudante2_cbo_unique UNIQUE (cbo),
    instituicao WITH ROWID REFERENCES tb_instituicao
);
```

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	TIPO_ESTUDANTE	VARCHAR2(20 BYTE)	Yes	(null)	1	(null)
2	INSTITUICAO	TP_INSTITUICAO	Yes	(null)	2	(null)
3	NOME	VARCHAR2(255 BYTE)	Yes	(null)	3	(null)
4	CPF	VARCHAR2(11 BYTE)	No	(null)	4	(null)
5	OBSERVACAO	VARCHAR2(200 BYTE)	Yes	(null)	5	(null)
6	CNS	VARCHAR2(255 BYTE)	Yes	(null)	6	(null)
7	PERIODO	NUMBER(2,0)	Yes	(null)	7	(null)
8	CURSO	VARCHAR2(100 BYTE)	Yes	(null)	8	(null)
9	CBO	VARCHAR2(6 BYTE)	Yes	(null)	9	(null)
10	PROGRAMA	VARCHAR2(100 BYTE)	Yes	(null)	10	(null)
11	REGISTRO_CONSELHO	VARCHAR2(9 BYTE)	Yes	(null)	11	(null)

```
CREATE SEQUENCE codigo_vaga
    START WITH 1
    INCREMENT BY 1
    NOCACHE;

CREATE TABLE tb_vaga OF tp_vaga (
    codigo DEFAULT codigo_vaga.NEXTVAL PRIMARY KEY,
    CONSTRAINT tb_vaga_indica_ativa_check_not_null CHECK (indica_ativa IS NOT
NULL),
    estudante WITH ROWID REFERENCES tb_estudante,
    servico WITH ROWID REFERENCES tb_servico
);
```

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ESTUDANTE	TP_ESTUDANTE	Yes	(null)	1	(null)
2	SERVICO	TP_SERVICO	Yes	(null)	2	(null)
3	VIGENCIA	TP_VIGENCIA_VAGA	Yes	(null)	3	(null)
4	CODIGO	NUMBER	No	"P241IN0940_EQ01"."CODIGO_VAGA"."NEXTVAL"	4	(null)
5	INDICA_ATIVA	NUMBER(1,0)	Yes	(null)	5	(null)

EXEMPLOS DE POVOAMENTO:

```
INSERT INTO tb_enfermeiro (
        nome,
        matricula,
        cpf,
        lotacao,
        servico,
        coren
    )
    VALUES (
        'Otávio Gonçalves',
        '22138193686',
        '6572938160',
        'Cardiologia',
        NULL,
        '94918/PE'
    );


    INSERT INTO tb_medico (
        nome,
        matricula,
        cpf,
        lotacao,
        servico,
        crm
    )
    VALUES (
        'Lucca da Paz',
        '15640859267',
        '46418705917',
        'Dermatologia',
        NULL,
        '84141/PE'
    );


INSERT INTO tb_estudante (
        tipo_estudante,
        instituicao,
        nome,
        cpf,
        observacao,
        cns,
        periodo,
        curso,
```

```
        cbo,
        programa,
        registro_conselho
    )
    VALUES (
        'Graduando',
      (SELECT REF(i) FROM tb_instituicao i WHERE i.cnpj = '12345678912345'),
        'Olivia das Neves',
        '45793654149',
        'Observação: 95999034818452809612393180833 4',
        '553312846990506',
        2,
        'Medicina',
        NULL,
        NULL,
        NULL
    );


INSERT INTO tb_instituicao (

     nome,
     cnpj,
     endereco,
     telefone
)
VALUES (
    'Universidade Federal de São Paulo',
    '15975335795159',
tp_endereco
('Avenida dos Médicos', 'Aclimação', 'São Paulo', 'SP', '98765-432'),
tp_telefones(
tp_telefone('11', '987654321'),
tp_telefone('11', '123456789'),
tp_telefone('11', '987654321'))
);

INSERT INTO tb_servico
(medico, enfermeiro, codigo, nome, capacidade)
VALUES (
       (SELECT REF(m) FROM tb_medico m WHERE m.matricula =
'15640859267'),
       (SELECT REF(e) FROM tb_enfermeiro e WHERE e.matricula =
'22138193686'),
  'CAR',
  'Cardiologia',
  65
);
```

```
INSERT INTO tb_vaga (estudante, servico, vigencia, indica_ativa)
   VALUES (
      (SELECT REF(e) FROM tb_estudante e WHERE e.cpf = '45793654149'),
      (SELECT REF(s) FROM tb_servico s WHERE s.codigo = 'ONCO'),
      tp_vigencia_vaga('16/03/2023','12/09/2023'),
      1);
```

ANEXO

A. CONSULTAS SOLICITADAS PELO CLIENTE PARA GERAR RELATÓRIOS

Quantidade de estudantes do curso de Medicina estão fazendo estágio no mês de julho.

```
SELECT COUNT(*) AS Quantidade

FROM tb_vaga v

WHERE v.vigencia.data_inicio BETWEEN to_date('01/07/23','DD/MM/RR')
AND to_date('31/07/23','DD/MM/RR')

AND

v.estudante.curso LIKE 'Medicina';
```

Lista dos estudantes que começaram o estágio no mês de julho, do curso de Medicina.

```
SELECT v.estudante.nome AS Estudante, v.estudante.curso AS Curso

FROM tb_vaga v

WHERE v.vigencia.data_inicio BETWEEN to_date('01/07/23','DD/MM/RR')
AND to_date('31/07/23','DD/MM/RR')

AND

v.estudante.curso LIKE 'Medicina';
```

Relação de estudantes por curso, estagiando em determinado setor.

```
SELECT v.estudante.nome AS Estudante, v.estudante.curso AS Curso,
v.servico.nome AS Serviço
FROM tb_vaga v
WHERE v.servico.codigo ='CAR'
```

Quantidade de estudantes por setor.

```
SELECT s.nome AS Serviço, COUNT(v.estudante.cpf) AS Estudantes
FROM tb_vaga v
INNER JOIN tb_servico s ON v.servico.codigo = s.codigo
GROUP BY v.servico.codigo, s.nome;
```

Quantidade de estagiários que estão rodando por setor.

```
SELECT s.nome AS Serviço, COUNT(v.estudante.cpf) AS Estudantes
FROM tb_vaga v
INNER JOIN tb_servico s ON v.servico.codigo = s.codigo
WHERE v.estudante.tipo_estudante = 'Graduando'
GROUP BY v.servico.codigo, s.nome;
```

B. COMANDOS E CONSULTAS TESTADOS BASEADOS NO CHECKLIST

3. CREATE SEQUENCE:

```
CREATE SEQUENCE id_vaga
START WITH 1
 INCREMENT BY 1
NOCACHE;
```

6. UPDATE

```
UPDATE tb_medico m
SET servico = (SELECT REF(s) FROM tb_servico s WHERE s.codigo =
'RADIO')
WHERE m.matricula = '38468769864'
```

7. DELETE

```
DELETE FROM tb_vaga v
WHERE v.codigo = 200;
```

9. BETWEEN

```
SELECT nome, cpf, periodo
FROM tb_estudante
WHERE tipo_estudante = 'Graduando'
AND curso = 'Medicina'
AND periodo BETWEEN 3 AND 5;
```

10. IN

```
select v.estudante.nome
from tb_vaga v
where v.estudante.curso NOT IN 'Medicina';
```

11. LIKE e RIGHT JOIN (Bonus)

```
SELECT
    est.nome,
    est.cpf,
    est.curso,
    est.instituicao.endereco.rua AS rua,
    est.instituicao.endereco.bairro AS bairro,
    est.instituicao.endereco.cidade AS cidade
FROM
tb_estudante est
RIGHT join
tb_vaga vag
ON
```

```
est.curso LIKE 'M%'
WHERE
DEREF(est.instituicao).endereco.cidade LIKE 'Recife' AND vag.indica_ativa = 1
```

13. INNER JOIN

```
SELECT v.codigo AS "Código da Vaga", v.vigencia.data_inicio AS "Início da Vigência", v.vigencia.data_fim AS "Fim da Vigência",
    s.nome AS "Nome do Serviço", s.capacidade AS "Capacidade do Serviço"
FROM tb_vaga v
INNER JOIN tb_servico s ON v.servico.codigo = s.codigo;
```

13. INNER JOIN + 17.COUNT + 23.GROUP BY

```
SELECT
    s.nome AS nome_servico,
    COUNT(v.codigo) AS total_vagas,
    SUM(CASE WHEN v.indica_ativa = 1 THEN 1 ELSE 0 END) AS vagas_ativas,
    COUNT(v.codigo) - SUM(CASE WHEN v.indica_ativa = 1 THEN 1 ELSE 0 END) AS vagas_desocupadas
FROM
    tb_servico s
INNER JOIN
    tb_vaga v ON s.codigo = v.servico.codigo
WHERE
    s.nome = 'Cardiologia'
GROUP BY
    s.nome;
```

	NOME_SERVICO	TOTAL_VAGAS	VAGAS_ATIVAS	VAGAS_DESOCUPADAS
1	Cardiologia	70	32	38

14. MAX

```
SELECT s.nome, s.capacidade AS nome_servico
FROM tb_servico s
WHERE s.capacidade = (
    SELECT MAX(capacidade)
    FROM tb_servico
);
```

15. MIN

```
SELECT MIN(capacidade) FROM tb_servico;
```

16. AVG:

```
SELECT AVG(capacidade) AS media_capacidade
FROM tb_servico;
```

18. SUBCONSULTA COM OPERADOR RELACIONAL

```
SELECT m.nome
FROM tb_estudante m
WHERE periodo < ALL (SELECT periodo FROM tb_estudante WHERE
m.instituicao.cnpj <> '12345678912345');
```

19. SUBCONSULTA COM IN

```
SELECT nome, curso
FROM tb_estudante
WHERE cpf IN (
    SELECT v.estudante.cpf
    FROM tb_vaga v
    WHERE DEREF(v.servico).codigo = 'ONCO');
```

20. SUBCONSULTA COM ANY

```
SELECT nome, capacidade FROM tb_servico
WHERE capacidade < ANY (SELECT capacidade FROM tb_servico WHERE
nome = 'RADIOLOGIA');
```

21. SUBCONSULTA COM ALL

```
SELECT nome, cpf, periodo FROM tb_estudante
WHERE periodo <> all (SELECT periodo FROM tb_estudante WHERE periodo
= 5)
order by periodo;
```

22. ORDER BY

```
SELECT e.nome, e.tipo_estudante, e.curso, e.programa, v.vigencia.data_inicio,
v.vigencia.data_fim
FROM tb_estudante e
INNER JOIN tb_vaga v ON REF(e) = v.estudante
INNER JOIN tb_servico s ON v.servico = REF(s)
WHERE s.nome = 'Cardiologia'
ORDER BY e.curso, e.programa;
```

23. GROUP BY e 43. SELECT DEREF

```
SELECT
    tipo_estudante, deref (e.instituicao).nome as "Nome da instituição",
count(nome) as "Total"
FROM
tb_estudante e
GROUP BY
    tipo_estudante, deref (e.instituicao).nome
```

```
ORDER BY "Total"
```

24. HAVING

```
SELECT codigo, nome, SUM(capacidade) AS capacidade_total
FROM tb_servico
GROUP BY codigo, nome
HAVING SUM(capacidade) > 15;
```

25. UNION ou INTERSECT ou MINUS:

```
SELECT v.estudante.nome as nome, v.servico.nome as Area_de_interesse FROM tb_vaga v  WHERE v.servico.nome = 'Cardiologia'
UNION
SELECT nome  as nome, lotacao as area_de_interessa  FROM tb_medico WHERE lotacao = 'Cardiologia'
```

26. CREAT VIEW

```
CREATE VIEW vw_estudantes_serviço AS
SELECT s.nome AS Serviço, COUNT(v.estudante.cpf) AS Estudantes
FROM tb_vaga v
INNER JOIN tb_servico s ON v.servico.codigo = s.codigo
GROUP BY v.servico.codigo, s.nome;
```

27. CREATE OR REPLACE TYPE

```
CREATE OR REPLACE TYPE tp_telefone AS OBJECT (
   cod_area VARCHAR2(2),
   numero VARCHAR2(9)
);
```

28. CREATE OR REPLACE TYPE BODY e 29. MEMBER PROCEDURE :

```
create or replace TYPE BODY tp_instituicao AS
```

```
    MEMBER PROCEDURE imprimir_endereco (self tp_instituicao) IS
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Detalhes da instituição');
        DBMS_OUTPUT.PUT_LINE('Nome: ' || nome);
        DBMS_OUTPUT.PUT_LINE('cnpj: ' || cnpj);
    END;
END;
```

36. NOT INSTANTIABLE TYPE/MEMBER:

```
create or replace TYPE tp_endereco AS OBJECT (
    rua VARCHAR2(100),
    bairro VARCHAR2(50),
    cidade VARCHAR2(50),
    uf VARCHAR2(2),
    cep VARCHAR2(9)
)
NOT INSTANTIABLE;
```

47. NESTED TABLE

```
CREATE TYPE tp_nt_telefones AS TABLE OF tp_telefone;
CREATE TABLE tb_listafone_instituicao(
cod_inst NUMBER(5),
listafone tp_nt_telefones
NESTED TABLE listafone STORE AS tb_listafone_instituicao;
```

D. OUTROS TESTES REALIZADOS:

Avaliação da restrição de integridade ao tentar deletar um médico que é coordenador de um serviço

```
DELETE
FROM tb_medico
WHERE nome = 'Lucca da Paz'
```

Exiba o nome, o cpf e as observações dos 20 primeiros estudantes cadastrados

```
SELECT nome, cpf, observacao
FROM tb_estudante
FETCH FIRST 20 ROWS ONLY;
```

Indique os estudantes que, no momento, ocupam vagas nos serviços?

```
SELECT e.nome AS estudante, v.codigo AS codigo_da_vaga, s.nome AS nome_serviço
FROM tb_vaga v
INNER JOIN tb_estudante e ON v.estudante = REF(e)
INNER JOIN tb_servico s ON v.servico = REF(s)
```

```
WHERE v.indica_ativa = 1;
```

Listar os nomes dos graduandos do 5º período do curso de medicina que estão nos serviços de Oncologia

```
SELECT e.nome, e.tipo_estudante
FROM tb_estudante e JOIN tb_vaga v
ON REF(e) = v.estudante
INNER JOIN tb_servico s
ON v.servico = REF(s)
WHERE e.tipo_estudante = 'Graduando'
AND e.periodo = 5
AND e.curso = 'Medicina'
AND s.codigo = 'ONCO';
```

Selecionar os nomes e os números dos conselhos do médico e do enfermeiro do serviço de Cardiologia.

```
SELECT s.nome, s.medico.nome,s.medico.crm, s.enfermeiro.nome,
s.enfermeiro.coren
FROM tb_servico s
WHERE s.nome = 'Cardiologia'
```

Verifique qual serviço o medico “Davi Mendes” é coordenador e quem é o enfermeiro desse mesmo serviço.

```
SELECT  s.nome, s.medico.nome, s.enfermeiro.nome
FROM tb_servico s
WHERE  s.medico.nome like 'Davi Mendes'
```

Qual é o CPF do estudante chamado "Erick Viana"?

```
SELECT cpf, nome
FROM tb_estudante
WHERE nome like 'Erick Viana';
```

Quantos serviços estão cadastrados no sistema?

```
SELECT COUNT(*) AS total_servicos
FROM tb_servico;
```

Qual é o endereço da Universidade de São Paulo?

```
SELECT
i.endereco.rua as rua, i.endereco.bairro as bairro, i.endereco.cidade as cidade
FROM tb_instituicao i
WHERE i.nome like 'Universidade de São Paulo'
```

Quantidade de estudantes do curso de Medicina estão fazendo estágio no mês de julho.

```
SELECT COUNT(*)
FROM tb_vaga v
WHERE v.vigencia.data_inicio BETWEEN to_date('01/07/23','DD/MM/RR')
AND to_date('31/07/23','DD/MM/RR')
AND
v.estudante.curso LIKE 'Medicina';
```

Lista dos estudantes(e seus períodos) que começaram o estágio no mês de julho, do curso de Medicina.

```
SELECT v.estudante.nome AS Estudante, v.estudante.periodo AS Período
FROM tb_vaga v
```

```
WHERE v.vigencia.data_inicio BETWEEN to_date('01/07/23','DD/MM/RR')
AND to_date('31/07/23','DD/MM/RR')

AND

v.estudante.curso LIKE 'Medicina';
```