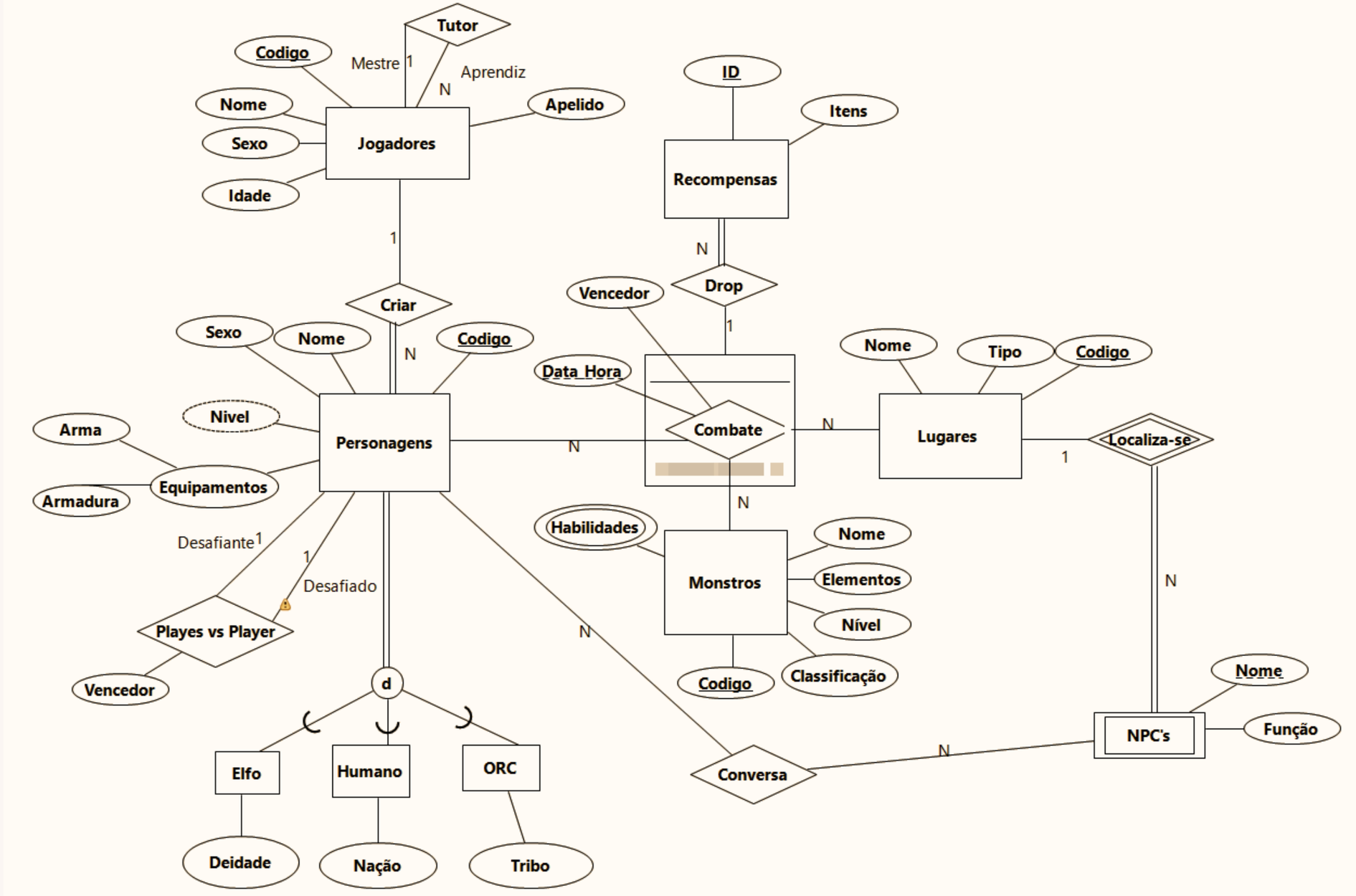
RPG

gjls, jlvs, jrsm, lfgp, pvoa, rjal

TÓPICOS DE ABORDAGEM

- Modelo Conceitual
- Modelo Logico
- Modelo Físico
- Consultas

MODELOCONCEITUAL



MODELO LÓGICO

1 -Jogadores (Cod_J, [Apelido]!, Nome!, Sexo, Idade!)

2 - Personagens (Cod_P, [Nome]!, Sexo!, Nivel!, cod_J)
codJ -> Jogadores(Cod_J)

3 -Equipamento (Cod_P, Arma, Armadura)
Cod_P -> Personagens(Cod_P)

4 -Elfo (Cod_E, Deidade!)
Cod_E -> Personagens(Cod_P)

MODELO LÓGICO

5 -Humano (Cod_H, Nação!)
Cod_H -> Personagens(Cod_P)

6 -Orc (Cod_A ,Tribo!)
Cod_A -> Personagens(Cod_P)

7 -Monstros (Cod_M, Nome!, Elemento!, Nível!, Classificação!)

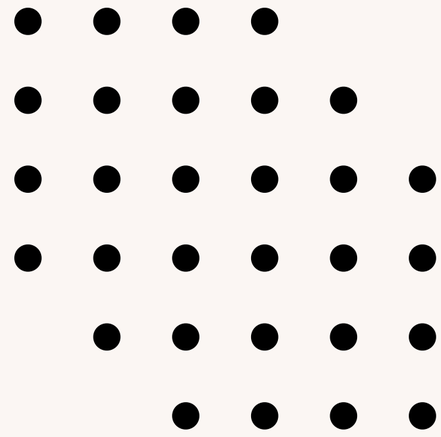
8 -Lugares (Cod_L, Nome!, Tipo!)

MODELO LÓGICO

9 -Habilidades (Cod_M, Habilidade)
Cod_M -> Monstros(Cod_M)

10 -NPC’s (Cod_L, Nome, Função!)
Cod-L -> Lugares(Cod_L)

11- Conversa (Cod_P, Cod_L, Nome)
Cod_P-> Personagem(Cod_P)
Cod_L, Nome -> NPC’s(Cod_L, Nome)

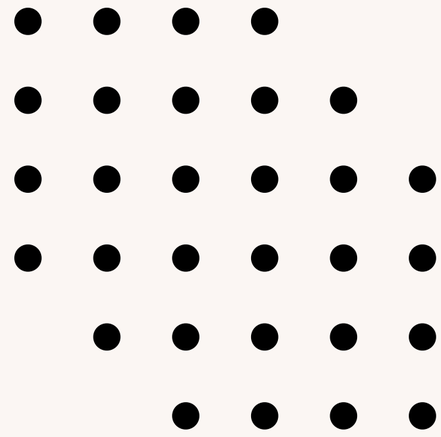


MODELO LÓGICO

13 - Player vs Player (Desafiante, Desafiado, Vencedor!)
Desafiante -> Personagens(Cod_P)
Desafiado -> Personagens(Cod_P)

14 - Combate (Cod_P, Cod_M, Cod_L, Data_hora!, Vencedor!)
Cod_P -> Personagens(Cod_P)
Cod_M -> Monstro(Cod_M)
Cod_L -> Lugares(Cod_L)

15 - Recompensas (ID, item, Cod_P, Cod_M, Cod_L, Data_hora)
Cod_P, Cod_M, Cod_L, Data_hora--> Combate(Cod_P, Cod_M, Cod_L, Data_hora)



MODELO FÍSICO

```
CREATE TABLE jogadores (
cod_j VARCHAR2 (4),
apelido VARCHAR2 (20) NOT NULL,
nome VARCHAR2 (255) NOT NULL,
Sexo CHAR,
idade NUMBER NOT NULL,

CONSTRAINT jogadores_pkey PRIMARY KEY (cod_j),
CONSTRAINT jogadores_akey UNIQUE (apelido),
CONSTRAINT jogadores_checkGenero CHECK (sexo = 'M'or sexo = 'F' or sexo = 'O')
);
```

MODELO FÍSICO

```
CREATE TABLE personagens (
cod_p VARCHAR2 (4),
nome VARCHAR2 (20) NOT NULL,
sexo CHAR NOT NULL,
nivel NUMBER NOT NULL,
cod_j VARCHAR2 (20),

CONSTRAINT personagens_pkey PRIMARY KEY (cod_p),
CONSTRAINT personagens_akey UNIQUE (nome),
CONSTRAINT personagens_fkey FOREIGN KEY (cod_j) REFERENCES jogadores (cod_j)
);
```

MODELO FÍSICO

```
CREATE TABLE equipamento (
cod_p VARCHAR2 (4),
arma VARCHAR2 (30),
armadura VARCHAR2 (20),

CONSTRAINT equipamento_pkey PRIMARY KEY (cod_p),
CONSTRAINT equipamento_fkey FOREIGN KEY (cod_p) REFERENCES
personagens(cod_p)
);

CREATE TABLE elfo (
cod_e VARCHAR2 (4),
deidade VARCHAR2 (20) NOT NULL,

CONSTRAINT elfo_pkey PRIMARY KEY (cod_e),
CONSTRAINT elfo_fkey FOREIGN KEY (cod_e) REFERENCES personagens(cod_p)
);
```

MODELO FÍSICO

```
CREATE TABLE humano (
cod_h VARCHAR2 (4),
nacao VARCHAR2 (20) NOT NULL,

CONSTRAINT humano_pkey PRIMARY KEY (cod_h),
CONSTRAINT humano_fkey FOREIGN KEY (cod_h) REFERENCES personagens(cod_p)
);

CREATE TABLE orc (
cod_o VARCHAR2 (4),
tribo VARCHAR2 (20) NOT NULL,

CONSTRAINT anao_pkey PRIMARY KEY (cod_o),
CONSTRAINT anao_fkey FOREIGN KEY (cod_o) REFERENCES personagens (cod_p)
);
```

MODELO FÍSICO

```
CREATE TABLE monstros (
cod_m VARCHAR2 (4),
nome VARCHAR2 (20) NOT NULL,
elemento VARCHAR2 (10) NOT NULL,
nivel NUMBER NOT NULL,
classificacao VARCHAR2 (20) NOT NULL,

CONSTRAINT monstros_pkey PRIMARY KEY (cod_m)
);

CREATE TABLE lugares (
cod_l VARCHAR2 (4),
nome VARCHAR2 (20) NOT NULL,
tipo VARCHAR2 (15) NOT NULL,

CONSTRAINT lugares_pkey PRIMARY KEY (cod_l)
);
```

MODELO FÍSICO

```
CREATE TABLE habilidades (
cod_m VARCHAR2 (4),
habilidade VARCHAR2 (30),

CONSTRAINT habilidades_pkey PRIMARY KEY (cod_m, habilidade),
CONSTRAINT habilidades_fkey FOREIGN KEY (cod_m) REFERENCES
monstros(cod_m)
);

CREATE TABLE npc (
cod_l VARCHAR2 (4),
nome VARCHAR2 (20),
funcao VARCHAR2 (10) NOT NULL,

CONSTRAINT npc_pkey PRIMARY KEY (cod_l, nome),
CONSTRAINT npc_fkey FOREIGN KEY (cod_l) REFERENCES lugares(cod_l)
);
```

MODELO FÍSICO

```
CREATE TABLE conversa (
cod_p VARCHAR2 (4),
cod_l VARCHAR2 (4),
nome VARCHAR2 (20),

CONSTRAINT conversa_pkey PRIMARY KEY (cod_p, cod_l, nome),
CONSTRAINT conversa_fkey1 FOREIGN KEY (cod_p) REFERENCES personagens(cod_p),
CONSTRAINT conversa_fkey3 FOREIGN KEY (cod_l, nome) REFERENCES npc(cod_l, nome)
);
```

MODELO FÍSICO

```
CREATE TABLE tutor (
aprendiz VARCHAR2 (4),
mestre VARCHAR2 (4),

CONSTRAINT tutor_pkey PRIMARY KEY (mestre, aprendiz),
CONSTRAINT tutor_fkey1 FOREIGN KEY (aprendiz) REFERENCES jogadores(cod_j),
CONSTRAINT tutor_fkey2 FOREIGN key (mestre) REFERENCES jogadores(cod_j)
);
```

MODELO FÍSICO

```
CREATE TABLE combate (
cod_p VARCHAR2 (4),
cod_m VARCHAR2 (4),
cod_l VARCHAR2 (4) ,
data_hora DATE NOT NULL,
vencedor VARCHAR2 (1) NOT NULL,

CONSTRAINT combate_pkey PRIMARY KEY (cod_p, cod_m, cod_l, data_hora),
CONSTRAINT combate_fkey1 FOREIGN KEY (cod_p) REFERENCES personagens(cod_p),
CONSTRAINT combate_fkey2 FOREIGN KEY (cod_m) REFERENCES monstros(cod_m),
CONSTRAINT combate_fkey3 FOREIGN KEY (cod_l) REFERENCES lugares(cod_l)
);
```

MODELO FÍSICO

```
CREATE TABLE recompensas (
id VARCHAR2 (4),
item VARCHAR2 (20),
cod_p VARCHAR2 (4),
cod_m VARCHAR2 (4),
cod_l VARCHAR2 (4),
data_hora DATE,

CONSTRAINT recompensas_pkey PRIMARY KEY (id),
CONSTRAINT recompensas_fkey1 FOREIGN KEY (cod_p, cod_m, cod_l, data_hora) REFERENCES
combate (cod_p, cod_m, cod_l, data_hora)
);
```

CONSULTAS

GROUP BY/HAVING

Retorna o nivel e a qtd de jogadores em cada nivel, que possuem nivel acima da media

```
SELECT nivel, COUNT(*) as Qnt_personagens
FROM personagens
WHERE nivel > (SELECT AVG(P.nivel)
                      FROM personagens P)
GROUP BY nivel
ORDER BY nivel ASC;
```

NIVEL	QNT_PERSONAGENS
45	1
50	2
60	2
62	1
70	1
99	1

CONSULTAS

JUNÇÃO EXTERNA

Retorna jogadores sem personagens

```
SELECT J.nome
FROM jogadores J LEFT JOIN personagens P ON
(J.cod_j = P.cod_j)
WHERE P.cod_j IS NULL
ORDER BY J.nome;
```

NOME
Luan
Rafael

CONSULTAS

JUNÇÃO INTERNA

Jogadores e seus personagens que desafiaram e sairam vitoriosos no PVP ordenado por nivel do menor para o maior

```
SELECT J.nome as PLAYER, P.NOME as PERSONAGEM, P.nivel as NIVEL
FROM jogadores J INNER JOIN personagens P ON (J.cod_j = P.cod_j)
INNER JOIN playervsplayer PVP ON (P.cod_p = PVP.desafiante)
WHERE vencedor = 0
ORDER BY nivel ASC;
```

PLAYER	PERSONAGEM	NIVEL
Ana	Almondegas	45
Gabriel	HeiFengxi	50
Gabriel	BaiFengxi	60

CONSULTAS

ANTI-JUNÇÃO

Retorna o nome dos personagens que não tem armadura

```
SELECT nome as vulneravel
FROM personagens
WHERE cod_p NOT IN (SELECT cod_p
                    FROM equipamento
WHERE equipamento.armadura IS NOT NULL);
```

VULNERAVEL
KarlDoctor
DarkToto
Weevil
Pixie

CONSULTAS

SEMI JOIN

- Mostrar as colunas dos lugares onde ocorreram boss fight (player e o jogador venceu)

```
SELECT *
FROM lugares l
WHERE EXISTS (SELECT c.COD_L
FROM combate c
WHERE l.COD_L = c.COD_L
AND c.vencedor = 1
AND c.COD_M IN (SELECT m.COD_M
FROM monstros m
WHERE m.classificacao = 'Boss')
);
```

COD_L	NOME	TIPO
3002	Arredores De Serdin	Campo

CONSULTAS

SUBCONSULTA ESCALAR

Retorna a data do drop do combate de um determinado local

```
SELECT data_hora
FROM combate
WHERE cod_l = (SELECT cod_l
               FROM recompensas
               WHERE id= '7006');
```

DATA_HORA
30-SEP-23

CONSULTAS

SUBCONSULTA DO TIPO LINHA

Retorna o nome e apelido do jogador que
possui o mesmo sexo e idade do jogador 1002

```
SELECT nome, apelido
FROM jogadores
WHERE (sexo, idade) = (SELECT  sexo, idade
                       FROM jogadores
                       WHERE cod_j = '1002');
```

NOME	APELIDO
Gabriel	BielzinGL
Wesley	Wes

CONSULTAS

SUBCONSULTA TABELA

retorna o código do local em que o vencedor do combate foi o jogador

```
SELECT cod_l
FROM lugares
WHERE cod_l in (SELECT cod_l
                FROM combate
                WHERE vencedor = 1);
```

COD_L
3002
3003
3005

CONSULTAS

OPERACAO DE CONJUNTO

Retorna todos os possiveis participantes em combate (Monstros e personagens)

```
SELECT nome AS
PARTICIPANTES_DE_COMBATE
FROM personagens
      UNION
SELECT nome
FROM monstros;
```

PARTICIPANTES_DE_COMBATE
20pegar70correr
Almondegas
BaiFengxi
BoltFreez
Christensen
DarkToto
Devorador de Mundos
Elvis

TRIGGER

Retorna mensagem de erro quando se tenta inserir um nível de personagem negativo

```
FOR EACH ROW
        DECLARE
                nivel_personagm_negativo EXCEPTION;
        BEGIN
                IF : NEW.nivel < 0 THEN
                        DBMS_OUTPUT.PUT_LINE('NIVEL DE PERSONAGEM NEGATIVO');
                        RAISE nivel_personagm_negativo;
                END IF;
                EXCEPTION
                        WHEN nivel_personagm_negativo THEN
                        Raise_application_error(
                                -20202, 'Nivel do personagem negativo.' || ' Não é possível inserir nível negativo!'
                        );
         END;
```

PROCEDIMENTO

Procedimento que mostro todos os personagens do jogador BielzinGL

```
CREATE PROCEDURE personagens_BielzinGL IS
        player VARCHAR(10) := '1002';
        CURSOR cur_character IS
        SELECT nome, nivel
        FROM personagens
        WHERE cod_j = player;

         nm personagens.nome%TYPE;
         nv personagens.nivel%TYPE;
         BEGIN
                OPEN cur_character;
                 LOOP
                        FETCH cur_character INTO nm, nv;
                        EXIT WHEN cur_character%NOTFOUND;
                        DBMS_OUTPUT.PUT_LINE (nm || ' ' || nv);
                  END LOOP;

          CLOSE cur_character;
          END;
```

PROCEDIMENTO

Procedimento para mostrar todos os personagens que possuem nomes iguais a monstros

```
CREATE PROCEDURE personagens_nome_monstro IS
BEGIN
        FOR entidade IN (SELECT nome
                                FROM personagens
                                WHERE nome IN (SELECT nome
                                                        FROM monstros))
        LOOP
                dbms_output.put_line(entidade.nome);
        END LOOP;
END;
```

Obrigado!