Modelagem de uma Plataforma Genômica Personalizada

José Lucas Silva Mayer (N. USP: 11819208)
Luã Nowacki Scavacini Santilli (N. USP: 11795492)
Maximilian Cabrajac Göritz (N. USP: 11795418)

Apresentação do Projeto

- O projeto consiste em uma plataforma genômica personalizada, cujo intuito é armazenar dados genéticos e fornecer informações a respeito de relações familiares de usuários.
- O sistema fornecerá informações sobre predisposições genéticas a doenças e analisará os dados coletados de exames para estabelecer relações familiares entre os usuários cadastrados.

Requisitos Principais

Requisitos de Dados

- Os bancos de dados devem armazenar **dados genéticos**, **informações do usuário** e **relações familiares**.
- Os usuários devem ser classificados em quatro categorias diferentes: **clientes**, **laboratoristas**, **médicos** e **administradores**.
- Os usuários são avisados a respeito do andamento de suas solicitações e de outras informações por meio de **notificações**.
- As solicitações podem ser direcionadas a um **painel genético**, que investiga uma dada predisposição pontual, de acordo com alguma recomendação prévia do usuário (investigar predisposição à leucemia, por exemplo).
- As solicitações também podem ser direcionadas a um **mapeamento completo**, que é mais custoso mas tem o intuito de oferecer informações mais ricas aos usuários, como uma lista de predisposições genéticas.

Requisitos do Sistema

- Os clientes podem **solicitar a coleta de material genético**, **visualizar sua árvore genealógica** e **receber notificações de novos membros da família**.
- Os laboratoristas podem **registrar o material genético** e **dar andamento às solicitações dos usuários**.
- Os médicos são os responsáveis por **verificarem as solicitações** e **definirem um parecer dos dados inferidos pelo sistema** (como a lista de predisposições, por exemplo).
- O sistema deve decidir a **lista de predisposições e probabilidades de tê-las** com base em mapeamentos já predefinidos.
- O sistema deve **notificar usuários** a respeito de **novos parentescos**.

Os Componentes do Sistema e os Diferentes Tipos de Bancos de Dados

Separação dos Sistemas

Relacional

- Informações de usuários e de tipos de usuário;
- Notificações;
- Coletas;
- Exames;
- Painéis;
- Condições associadas a usuários

Chave-valor

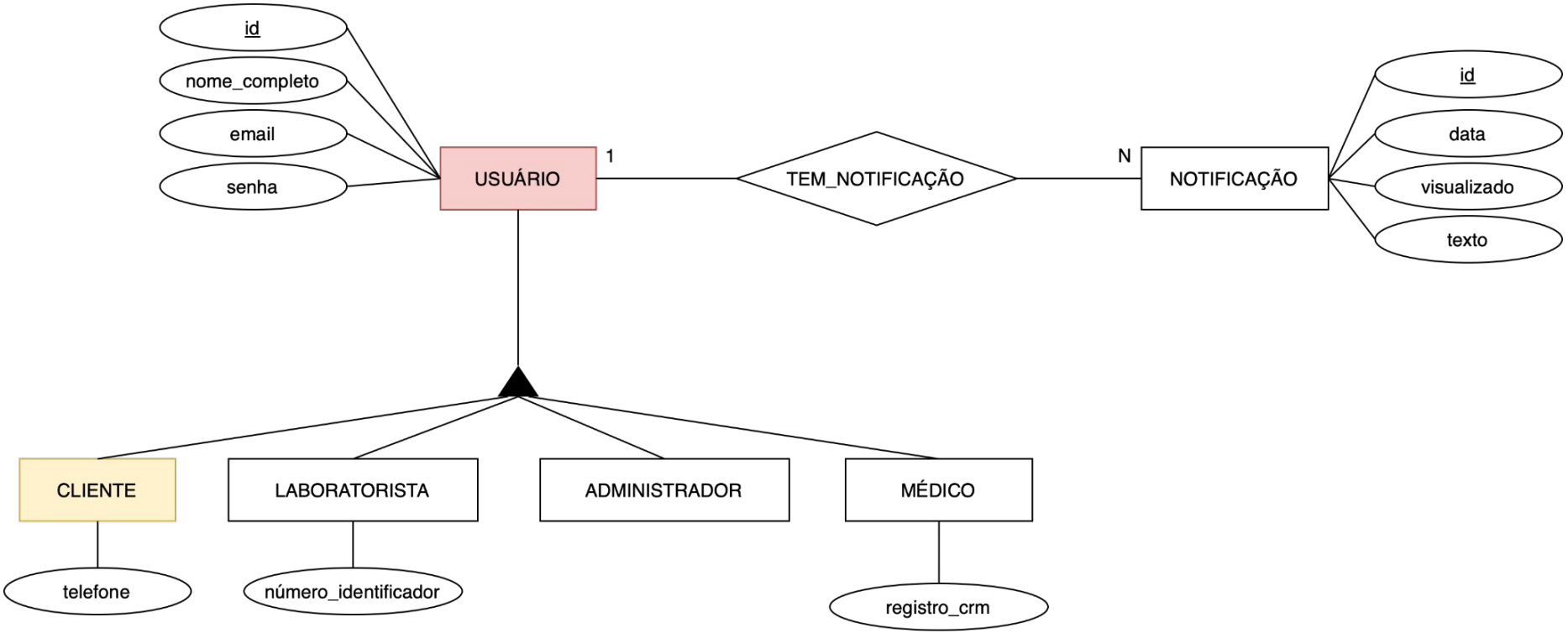
- Identificação de parentesco
- Descoberta de condições genéticas

Grafos

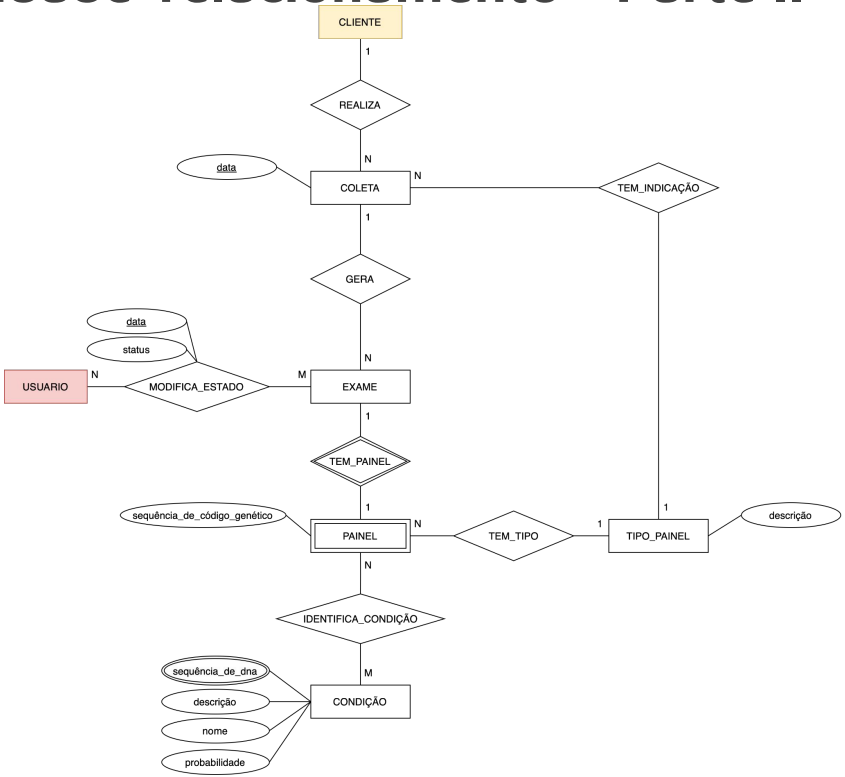
- Relacionamentos familiares entre usuários

Modelagem dos Bancos de Dados

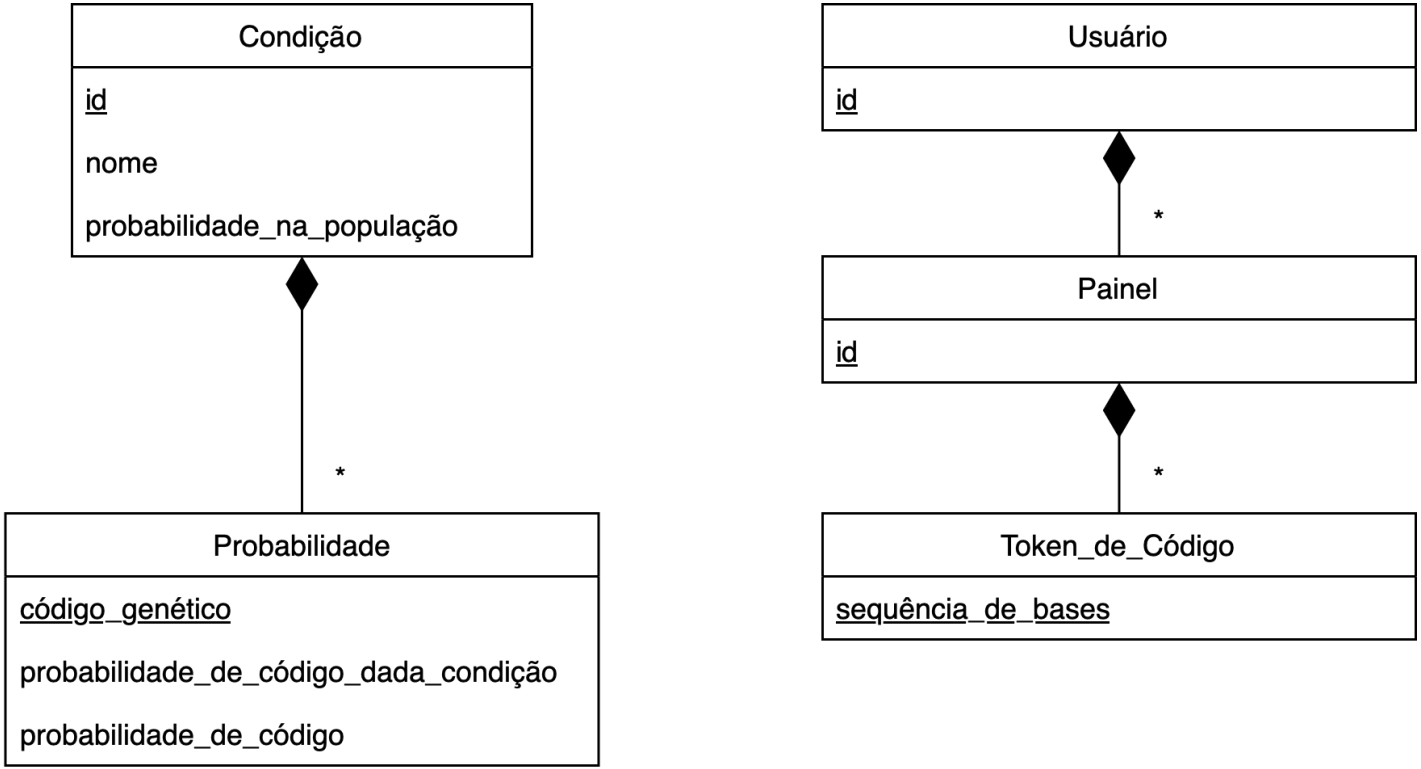
Banco de Dados Relacional
Modelo Entidade-relacionamento - Parte I



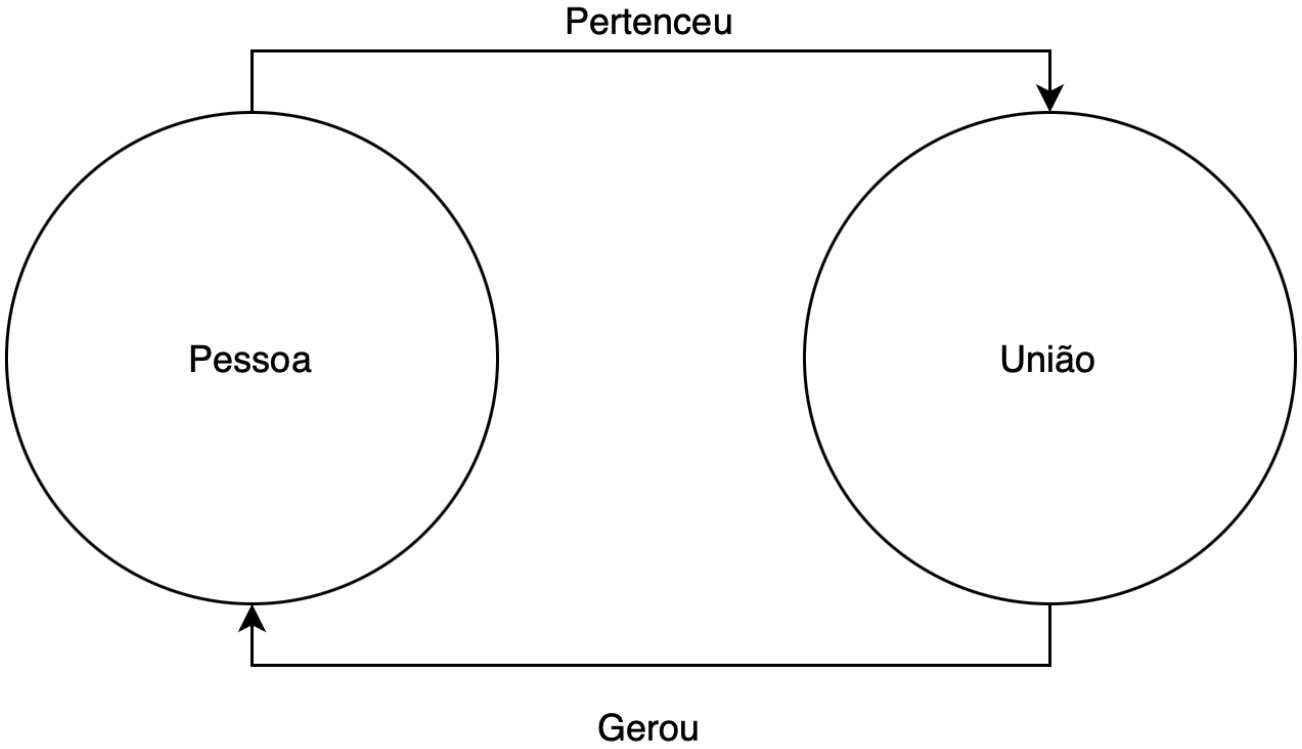
Banco de Dados Relacional
Modelo Entidade-relacionamento - Parte II



Banco de Dados Orientado a Chave-valor



Banco de Dados Orientado a Grafo



Estabelecendo a Probabilidade de uma Condição

Probabilidade bayesiana

$$P(A|B) = P(A) * P(B|A) / P(B)$$

A := Pessoa possui condição
B := Determinada sequência genética está presente

P(A) = Prob. da pessoa possuir a condição A
P(B) = Prob. da seq. gen. estar presente
P(B|A) = Prob. da cond. dado que a seq. gen. está presente

Exemplo: Probabilidade bayesiana

A := Pessoa possui osteoporose
B := Pessoa contém a seq. gen. **aagtc**

P(A) = 0.06 (prob. na população em geral)
P(B) = 0.01 (prob. da pessoa possuir a seq. gen.)
P(B|A) = 0.3 (prob. da pessoa possuir a cond. dado B)

P(A|B) = 0.06 *0.01 / 0.3

P(A|B) = P(A) * P(B|A) / P(B)

Pseudo-código
Chave-Valor para o Redis

Adicionando usuários no Redis

```
def add_user_to_redis(user):
     sadd f”user:{user.id}:codigos {user.codigos}”

codigo: {
     seq: <a|t|c|g>[]
}

user: {
     id: <number>
     codigos: <codigo>[]
}

Exemplo: sadd user:3:codigos gatc gaaatc tagc taaaagc
```

Adicionando condições no Redis

```
def add_condition(condition):
     sadd f"condicao:{condition.id}:codigos condition.codigos”
     for seq, prob in condition.codigos:
          hset f“condicao:{condition.id}:probs {seq} {prob}”

seq_w_probs: {
     seq: <a|t|c|g>[]
     prob_disso: <number>
     prob_da_cond_dado_isso: <number>
}

condition: {
     name: <string>
     prob_da_cond: <number>
     seqs: <seq_w_probs>[]
}
```

Descobrindo condições no Redis

```
def find_user_conditions(user):
     conditions_with_probs = []
     for cond in conditions:
          condition_prob = cond.prob_na_pop
          user_key = f”users:{user.id}:codigos”
          cond_key = f”condicao:{cond.id}:codigos”
          intersec = sinter user_key cond_key
          probs = hmget intersec
          for prob in probs:
               condition_prob = bayesian_update(condition_prob, prob)
          condition_with_probs.append((cond, condition_prob))
```

Indice de semelhanca entre usuários

```
def find_related(user):
     user_key = f”users:{user.id}:codigos”
     possibly_related = []
     for other in users:
          if user == other: return
          other_key = f"users:{other.cod}:codigos"
          intersec = sinter user_key other_key
          indice_de_similaridade = f(intersec)
          if indice_de_similaridade > CONSTANTE_SIMILARIDADE:
               possibly_related.append(other)
     return possibly_related
```