# Universo do Discurso

## §1 Introdução

O presente trabalho busca modelar um aplicativo de interação entre empresas do ramo alimentício e a população no geral. Os conhecimentos construídos a partir de disciplina de Fundamentos de Bancos de Dados, portanto, estão sendo utilizados na modelagem de estruturas (entidades e relacionamentos, ER de agora em diante) que visam representar clientes, empresas e alimentos (as "grandes classes" do trabalho), bem como outras relações, descritas abaixo.

As relações estabelecidas buscam

1. permitir que clientes encontrem informações acerca de empresas de alimentação e de alimentos consumidos;
2. permitir que empresas encontrem informações sobre seus possíveis produtos

As entidades e relações descritas neste documento fazem uso de database markup language para expressar as estruturas do banco de dados de forma mais simples na representação dos diagramas ER, e fazem uso da sintaxe do PostgreSQL para demonstrar uma primeira versão, possivelmente passível de modificações, das tabelas a serem implementadas. Os modelos ER foram criados através da aplicação de database markup language sobres o web-aplicativos dbdiagram e lucid.app.

---

## §2 Para que serve este aplicativo?

Tanto para empresas no geral quanto para usuários, o aplicativo fornece informações sobre

- valores nutricionais de diversos alimentos;
- uma possível precificação de alimentos individuais ou de refeições;
- a receita de uma refeição; e
- formas de categorizar e encontrar alimentos, refeições ou restaurantes, de acordo com suas preferências.

---

## §3 Organização, entidades e relações

### §3.1 `Alimentos`

A principal entidade de todo o sistema refere-se aos objetos `Alimentos`, que marcam alimentos individuais. Objetos desta categoria são bem descritas através da tabela abaixo:

```
TABLE Alimentos(
        id integer NOT NULL,
        name varchar NOT NULL,
        price float,
        primary key (food_id)
)
```

O atributo `price` é introduzido em um segundo momento, a partir do acesso à outra tabela.

### §3.2 `AlimentoData`

A entidade `AlimentoData` relaciona-se diretamente à entidade `Alimentos` como um descritor dos **valores nutricionais** de um elemento `Alimentos`. Seus valores foram extraídos do dataset 'Food, Vitamins, Minerals, Macronutrient', encontrado em Kaggle.com. Um objeto `AlimentoData` será representado por

```
TABLE AlimentoData(
        food_id integer NOT NULL,
        description varchar NOT NULL,
        category varchar NOT NULL,

        -- diversos valores, referentes aos nutrientes

        primary key (category, description)
        foreign key (food_id) references (Alimentos)
)
```

#### Valores nutricionais associados à `AlimentoData`

Os diversos valores, indicados pelo comentário acima, são

`"Nutrient Data Bank Number", "Data.Alpha Carotene", "Data.Beta Carotene","Data.Beta Cryptoxanthin", "Data.Carbohydrate", "Data.Cholesterol", "Data.Choline", "Data.Fiber", "Data.Lutein and Zeaxanthin", "Data.Lycopene", "Data.Niacin", "Data.Protein", "Data.Retinol", "Data.Riboflavin", "Data.Selenium", "Data.Sugar Total",`

```
"Data.Thiamin", "Data.Water", "Data.Fat.Monosaturated Fat",
"Data.Fat.Polysaturated Fat", "Data.Fat.Saturated Fat", "Data.Fat.Total
Lipid", "Data.Major Minerals.Calcium", "Data.Major Minerals.Copper",
"Data.Major Minerals.Iron", "Data.Major Minerals.Magnesium", "Data.Major
Minerals.Phosphorus", "Data.Major Minerals.Potassium", "Data.Major
Minerals.Sodium", "Data.Major Minerals.Zinc", "Data.Vitamins.Vitamin A -
RAE", "Data.Vitamins.Vitamin B12", "Data.Vitamins.Vitamin B6",
"Data.Vitamins.Vitamin C", "Data.Vitamins.Vitamin E",
"Data.Vitamins.Vitamin K"
```

conforme encontrados no *dataset* descrito.

---

### §3.2 `Receitas`, `ReceitaTimings` e `Ingredientes`

Uma `Receita`, relaciona-se diretamente com as duas entidades descritas anteriormente. Para definir uma receita, é necessário que tenhamos uma lista de ingredientes; cada um destes ingredientes refere-se a um elemento `Alimento`; por sua vez, cada um desses ingredientes estará referenciado por um `AlimentoData` que contém seus valores nutricionais. Uma vez precificados cada um dos ingredientes, a receita como um todo pode ser precificada.

```
Table Ingredientes(
        food_id INTEGER NOT NULL,
        recipe_id INTEGER NOT NULL,
        category  VARCHAR NOT NULL,
        quantity FLOAT NOT NULL,
        measure VARCHAR NOT NULL,
        primary key (food_id, recipe_id),
        foreign key (food_id) references id (Alimentos),
        foreign key (recipe_id) references rt_id (Receitas),
        foreign key (category) references category (AlimentoData)
)
```

```
Table Receitas(
        title VARCHAR NOT NULL,
        category VARCHAR NOT NULL,
        rt_id INTEGER NOT NULL,
        servings INTEGER,
        yield VARCHAR,
        price FLOAT,
        primary key (title),
        foreign key (ingridients) REFERENCES id (Alimentos)
        foreign key (rt_id) REFERENCES rt_id (ReceitaTimings),
)
```

```
Table ReceitaTimings(
        recipe_title VARCHAR NOT NULL,
        rt_id INTEGER NOT NULL,
        prep_time VARCHAR NOT NULL,
        cook_time VARCHAR NOT NULL,
        total_time VARCHAR,
        primary key (rt_id),
        foreign key (recipe_title) REFERENCES (Receitas)
)
```

Podemos definir estas tabelas utilizando database markup language, o que nos garante uma organização do tipo:

```
-- Definição das entidades
Table Alimentos{
        id integer [primary key]
        name varchar
        price float
}

-- A fim de econimizar espaço, apenas quatro nutrientes são apresentados
Table AlimentoData{
        food_id integer
        description varchar [primary key]
        category varchar [primary key]
        alpha_carotene float
        beta_carotene float
        cryptoxanthin float
        carbohydrate float
}

Table Receitas{
        title varchar [primary key]
        category varchar
        rt_id integer
        servings integer
        yield varchar
        price float
}

Table Ingredientes{
        food_id integer [primary key]
        recipe_id integer [primary key]
        category varchar
        quantity float
        measure varchar
}


Table ReceitaTimings{
        recipe_title varchar
        rt_id integer [primary key]
        prep_time float
        cook_time float
        total_time float
}

-- Definição das relações
Ref: AlimentoData.food_id < Alimentos.id
Ref: Ingrediente.food_id < Alimentos.id
Ref: Ingrediente.recipe_id < Receitas.rt_id
```

```
Ref: ReceitaTimings.rt_id < Receitas.rt_id
Ref: ReceitaTimings.recipe_title < Receitas.name
```

### §3.3 Usuários, clientes, empresas e restaurantes

Uma entidade `User` é um tipo geral sobre o qual todas as demais entidades aqui descritas pertencem.

```
Table Usuarios(
        id INTEGER NOT NULL,
        type VARCHAR NOT NULL,
        primary key (id),
        check (type in ('Client', 'Restaurant'))
)
```

```
Table Clientes(
        id INTEGER NOT NULL,
        name VARCHAR NOT NULL,

        -- relações e atributos a serem implementados

        primary key (id, name),
        foreign key (id) REFERENCES id (Usuarios)
)
```

```
Table Restaurantes(
        id INTEGER NOT NULL,
        c_id INTEGER NOT NULL,
        name VARCHAR NOT NULL,
        category,

        -- relações e atributos a serem implementados

        primary key (name),
        foreign key (c_id) REFERENCES  c_id (Empresas),
        foreign key (category) REFERENCES category (Alimentos)
        foreign key (id) REFERENCES id (Usuarios)
)
```

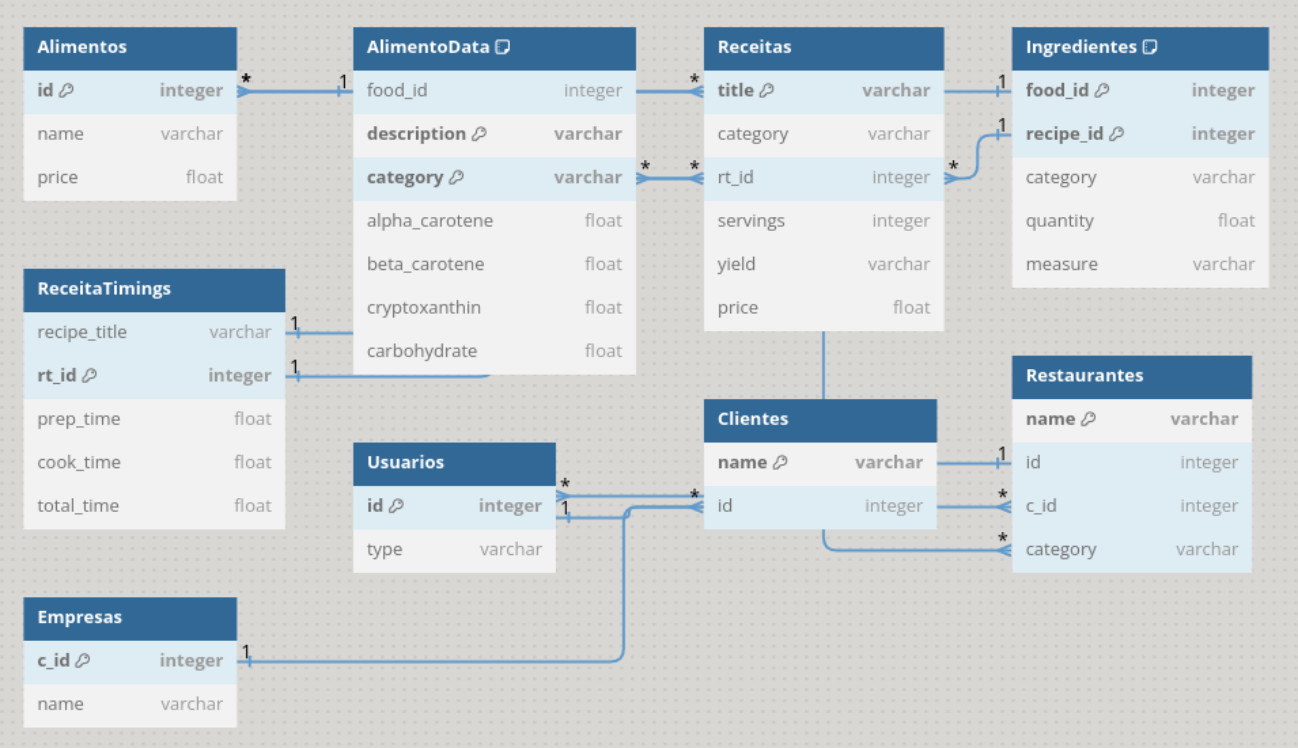
```
Table Empresas(
        c_id INTEGER NOT NULL,
        name VARCHAR NOT NULL,

        -- relações e atributos a serem implementados

        primary key (company_id)
)
```

Dessa forma, cada `Restaurante` é associado a uma `Empresa`. Uma empresa, naturalmente, está associada a nenhum, um ou mais restaurantes. Cada restaurante, por sua vez, pertencerá a uma categoria, como "culinária japonesa" ou qualquer outra das categorias que o atributo `Receita.category` pode receber. É evidente, no entanto, que é possível um restaurante pode servir refeições de mais de uma categoria, mas esta especialização será utilizada na implementação do aplicativo final.

O acesso aos dados sempre se dará mediante ao acesso de um `User`. As restrições e *features* do produto final, entretanto, discriminarão entre um usuário que representa um cliente e entre um usuário que representa uma empresa ou restaurante.

Em database markup language, estas estruturas foram implementadas até o momento sob a seguinte forma:

```
Table Usuarios{
        id integer [primary key]
        type varchar|
}

Table Clientes{
        name varchar [primary key]
        id integer
}

Table Restaurantes{
        name varchar [primary key]
        id integer
        c_id integer
        category varchar
}

Table Empresas{
        c_id integer [primary key]
        name varchar
}

Ref: Restaurantes.c_id > Empresas.c_id
Ref: Restaurantes.id < Usuarios.id
Ref: Restaurantes.category <> AlimentoData.category
Ref: Clientes.id > Usuarios.id
```


Alimentos
id integer
name varchar
price float
AlimentoData
food_id integer
description varchar
category varchar
alpha_carotene float
beta_carotene float
cryptoxanthin float
carbohydrate float
Receitas
title varchar
category varchar
rt_id integer
servings integer
yield varchar
price float
Ingredientes
food_id integer
recipe_id integer
category varchar
quantity float
measure varchar
ReceitaTimings
recipe_title varchar
rt_id integer
prep_time float
cook_time float
total_time float
Usuarios
id integer
type varchar
Clientes
name varchar
id integer
Restaurantes
name varchar
id integer
c_id integer
category varchar
Empresas
c_id integer
name varchar

### §3.4 Precificação

A precificação será feita através da coleção e manipulação de dados por parte do programa. Esses dados são inseridos ou extraídos de clientes e de restaurantes. Um cliente pode precificar objetos `Alimento`, enquanto um restaurante pode precificar objetos `Receita`.

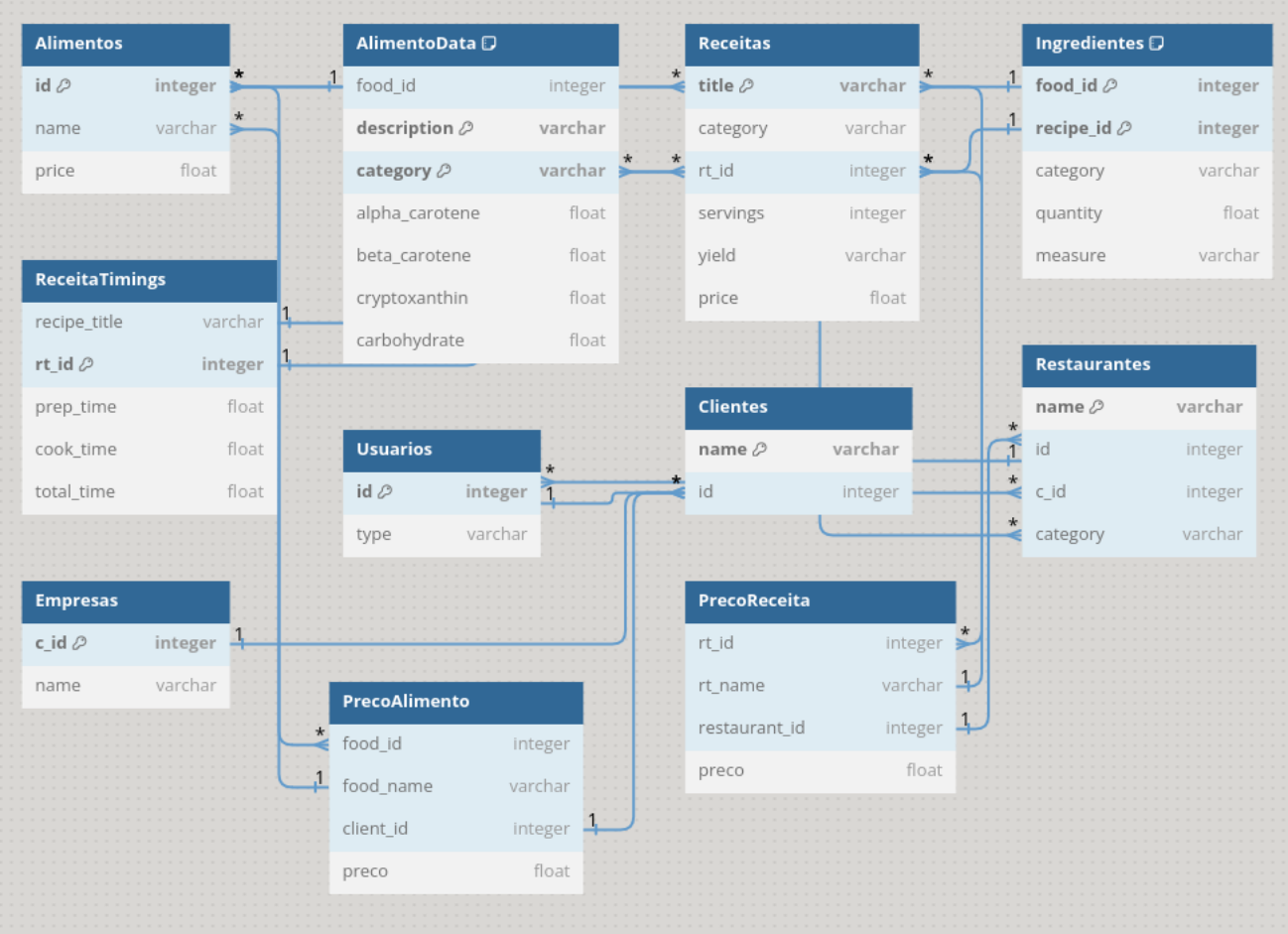
```
Table PrecoAlimento(
        food_id integer NOT NULL,
        food_name varchar NOT NULL,
        client_id integer NOT NULL,
        preco float NOT NULL,
        primary key (client_id, food_id),
        foreign key (food_id) REFERENCES id (Alimentos)
        foreign key (food_name) REFERENCES name (Alimentos)
        foreign key (client_id) REFERENCES id (Clientes)
)

Table PrecoReceita(
        rt_id integer NOT NULL,
        rt_name varchar NOT NULL,
        restaurant_id integer NOT NULL,
        preco float NOT NULL,
        primary key (restaurant_id, rt_id),
        foreign key (rt_id) REFERENCES rt_id (Receitas),
        foreign key (rt_name) REFERENCES name (Receitas)
)
```

**Construção dos preços**

A implementação da *feature* de precificação dos alimentos (objetos `Alimento`) se dará através de um cálculo sobre todas as entradas na tabela `Prices` nas quais o objeto `Alimento` é referenciado.
Se, por exemplo, a precificação for realizada através do produto de uma média de $n$ objetos `PrecoAlimento` que referenciam um determinado objeto `Alimento`, por um fator de lucro $k$, diga-se

$$\text{Price}_{\text{Food, Final}} = k \cdot \frac{1}{n} \cdot \sum_{i=0}^{n} \text{Price}_i$$

onde cada $\text{Price}_i$ será extraído da tabela `PrecoAlimento` para formar o $\text{Price}_{\text{Food, Final}}$, isto é, o atributo `Alimentos.preco` de um objeto `Alimento`.
Processo similar ocorre nos objetos `Receita`, onde a inserção dos preços é feita por `Restaurantes`.


Alimentos
id integer
name varchar
price float
AlimentoData
food_id integer
description varchar
category varchar
alpha_carotene float
beta_carotene float
cryptoxanthin float
carbohydrate float
Receitas
title varchar
category varchar
rt_id integer
servings integer
yield varchar
price float
Ingredientes
food_id integer
recipe_id integer
category varchar
quantity float
measure varchar
ReceitaTimings
recipe_title varchar
rt_id integer
prep_time float
cook_time float
total_time float
Clientes
name varchar
id integer
Restaurantes
name varchar
id integer
c_id integer
category varchar
Usuarios
id integer
type varchar
Empresas
c_id integer
name varchar
PrecoReceita
rt_id integer
rt_name varchar
restaurant_id integer
preco float
PrecoAlimento
food_id integer
food_name varchar
client_id integer
preco float

```
Table Descritores()
        name varchar NOT NULL,
        description varchar NOT NULL,
        primary key (name)
```

### Descritores de atributos para `AlimentoData`

A tabela `Descritores` reúne todos os descritores necessários para descrever os atributos nutricionais dos alimentos. Assim, uma vez implementado a aplicação final, será possível receber a descrição de um atributo, `carbohydrate`, por exemplo, através de

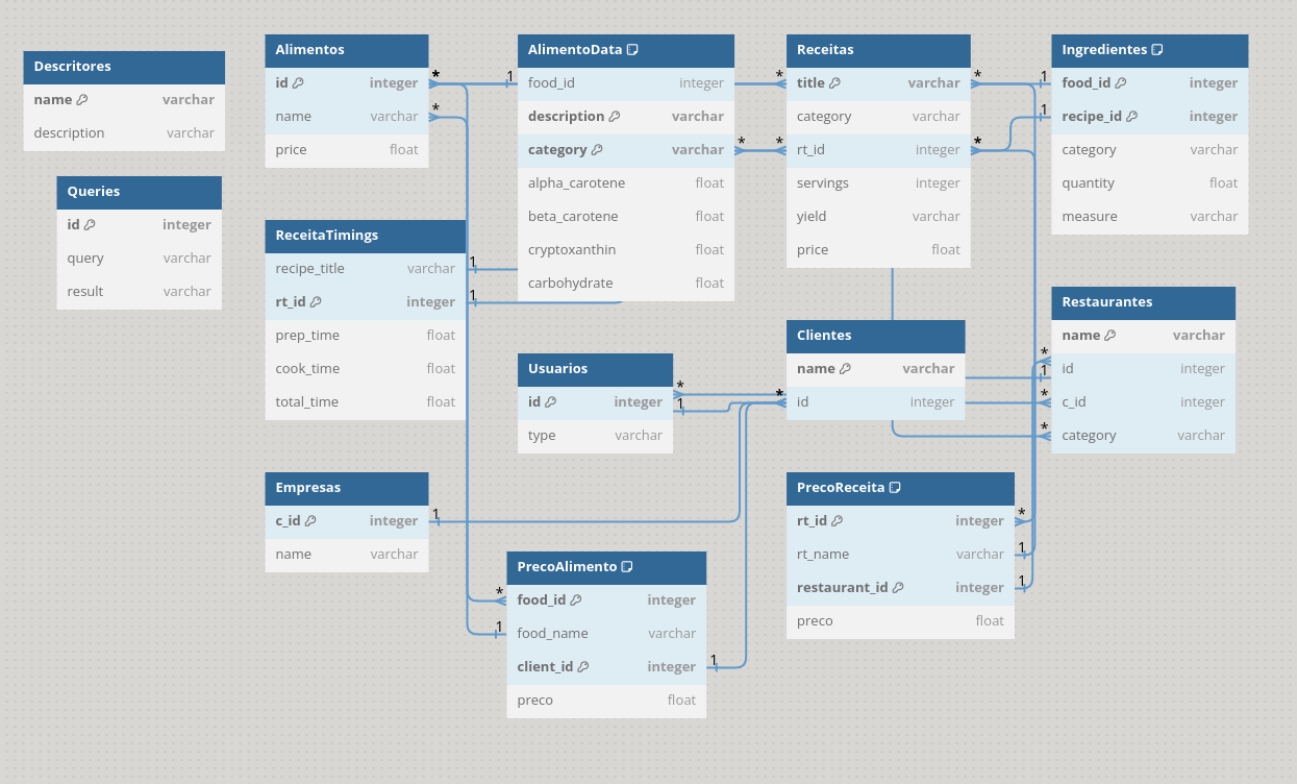
```
> describe carbohydrate
```

o que retornará a projeção do atributos `(name, description)` na tabela `Descriptors` onde `Descritores.name = 'carbohydrate'`.

```
Table Queries(
        id INTEGER NOT NULL,
        query VARCHAR NOT NULL,
        result VARCHAR
        primary key (id)
)
```

### Armazenar os resultados de pesquisas feitas por usuários

As pesquisas feitas por usuários poderão ser armazenadas em uma tabela, `Queries`, composta por um identificador único, a *query* do usuário e o resultado obtido pelo sistema. Desta forma, tanto uma organização interna voltada para melhoramentos (análise, por exemplo, dos tipos de pesquisas mais frequentemente feitas), quanto a análise para descoberta de possíveis *bugs* ou *queries* que retornam resultados indesejados ou errôneos.


Descritores
name varchar
description varchar
Queries
id integer
query varchar
result varchar
Alimentos
id integer
name varchar
price float
AlimentoData
food_id integer
description varchar
category varchar
alpha_carotene float
beta_carotene float
cryptoxanthin float
carbohydrate float
Receitas
title varchar
category varchar
rt_id integer
servings integer
yield varchar
price float
Ingredientes
food_id integer
recipe_id integer
category varchar
quantity float
measure varchar
ReceitaTimings
recipe_title varchar
rt_id integer
prep_time float
cook_time float
total_time float
Usuarios
id integer
type varchar
Clientes
name varchar
id integer
Restaurantes
name varchar
id integer
c_id integer
category varchar
Empresas
c_id integer
name varchar
PrecoAlimento
food_id integer
food_name varchar
client_id integer
preco float
PrecoReceita
rt_id integer
rt_name varchar
restaurant_id integer
preco float