Banco de Dados II
Ótica

Sávio Santos Serra

Sumário

Mini-mundo

A ótica X atualmente realiza o controle de ordens de serviço e pagamentos de clientes manualmente, aspectos esses que mais impactam em sua receita. Contudo, existe uma grande dificuldade para verificar quando uma ordem de serviço foi solicitada, ou quando um pagamento deve ser realizado. Assim, a ótica tem interesse em um sistema que possa automatizar tais interações e extrair relatórios do sistema, como pagamentos a serem recebidos em certo mês, ordens de serviço solicitadas, funcionários mais ativos, receita bruta, entre outros.

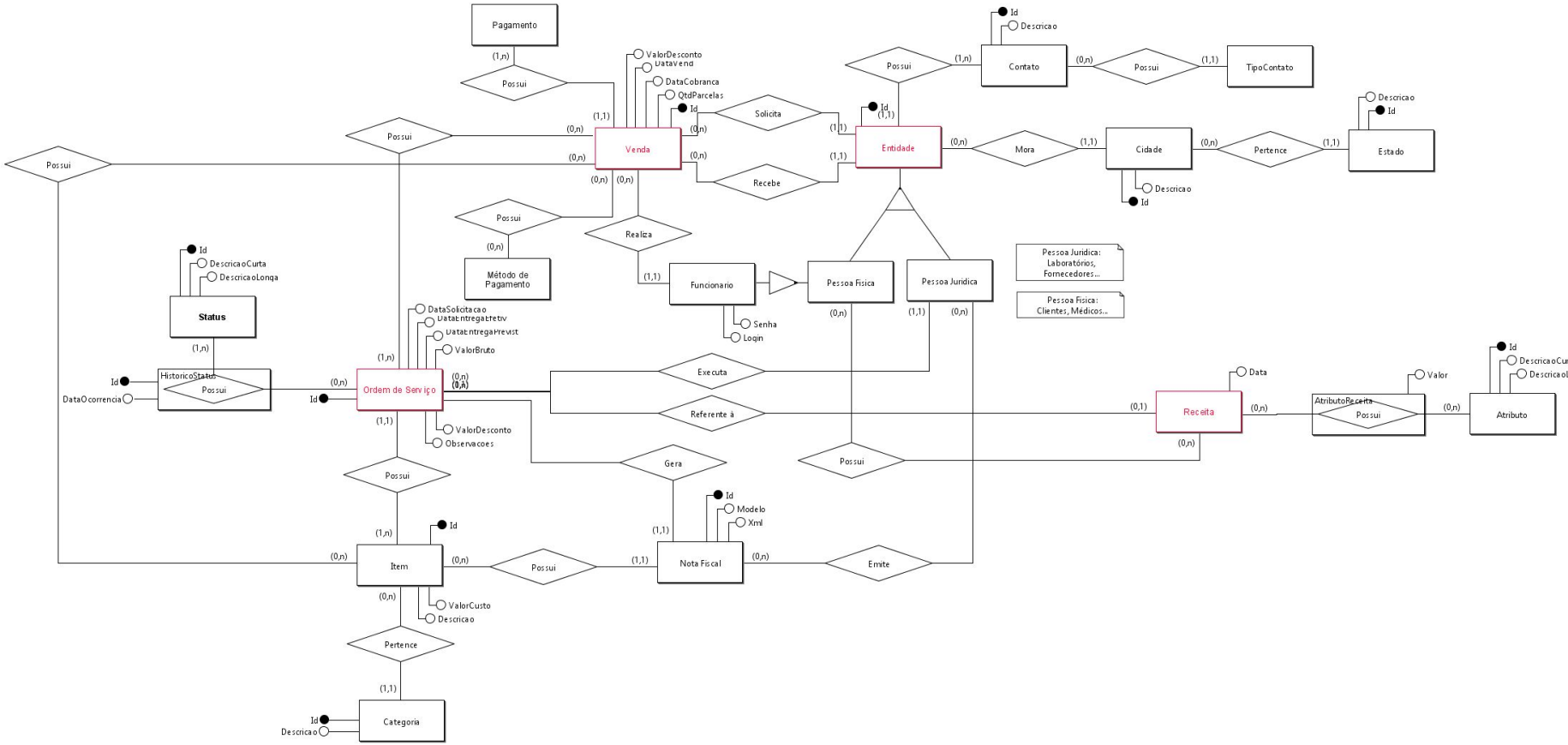
O processo atualmente ocorre da seguinte forma:

A. Um cliente solicita uma venda. Caso ele já possua ficha no estabelecimento, sua ficha é resgatada; caso contrário, é criada. A venda pode ser sobre vários itens, como armações e lentes;

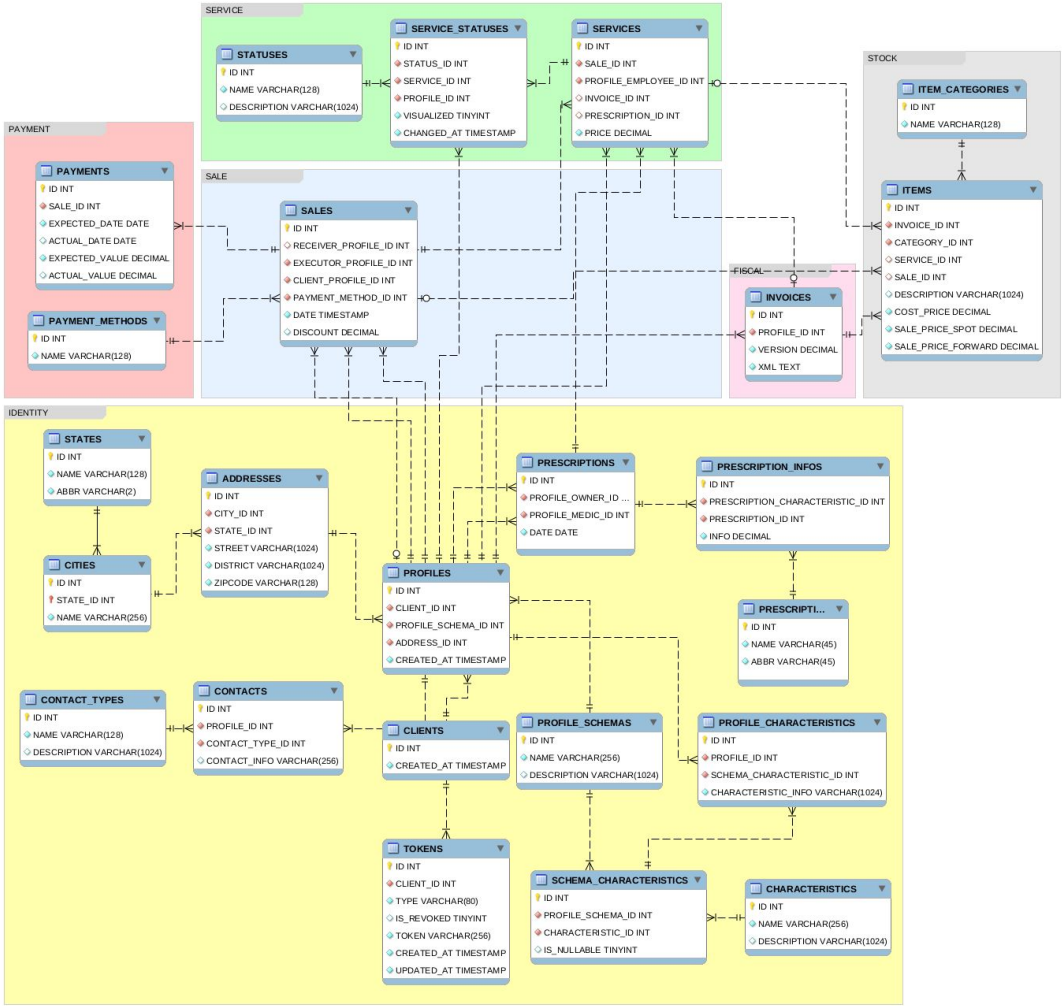
B. Caso um item seja uma lente, será necessário o envio de uma ordem de serviço para um laboratório para produzir a lente (e montar caso acompanhe a armação). A lente é feita com base em uma receita prescrita por um profissional oftalmologista;

C. Uma vez que o laboratório entrega a ordem de serviço, o cliente é notificado e pode buscar o item na loja.

Modelo Conceitual


Pagamento
Possui
Venda
ValorDesconto
DataVenda
DataCobranca
QtdParcelas
Id
Solicita
Recebe
Entidade
Possui
Contato
Id
Descricao
TipoContato
Mora
Cidade
Pertence
Estado
Método de Pagamento
Realiza
Funcionario
Senha
Login
Pessoa Fisica
Pessoa Juridica
Pessoa Juridica: Laboratórios, Fornecedores...
Pessoa Fisica: Clientes, Médicos...
Status
DescricaoCurta
DescricaoLonga
HistoricoStatus
DataOcorrencia
Ordem de Serviço
DataSolicitacao
DataEntregaRetirada
DataEntregaPrevista
ValorBruto
ValorDesconto
Observacoes
Executa
Referente à
Gera
Receita
Data
Valor
AtributoReceita
Atributo
Item
ValorCusto
Descricao
Nota Fiscal
Modelo
Xml
Emite
Categoria

Modelo Físico


SERVICE
STATUSES
ID INT
NAME VARCHAR(128)
DESCRIPTION VARCHAR(1024)
SERVICE_STATUSES
ID INT
STATUS_ID INT
SERVICE_ID INT
PROFILE_ID INT
VISUALIZED TINYINT
CHANGED_AT TIMESTAMP
SERVICES
ID INT
SALE_ID INT
PROFILE_EMPLOYEE_ID INT
INVOICE_ID INT
PRESCRIPTION_ID INT
PRICE DECIMAL
STOCK
ITEM_CATEGORIES
ID INT
NAME VARCHAR(128)
ITEMS
ID INT
INVOICE_ID INT
CATEGORY_ID INT
SERVICE_ID INT
SALE_ID INT
DESCRIPTION VARCHAR(1024)
COST_PRICE DECIMAL
SALE_PRICE_SPOT DECIMAL
SALE_PRICE_FORWARD DECIMAL
PAYMENT
PAYMENTS
ID INT
SALE_ID INT
EXPECTED_DATE DATE
ACTUAL_DATE DATE
EXPECTED_VALUE DECIMAL
ACTUAL_VALUE DECIMAL
PAYMENT_METHODS
ID INT
NAME VARCHAR(128)
SALE
SALES
ID INT
RECEIVER_PROFILE_ID INT
EXECUTOR_PROFILE_ID INT
CLIENT_PROFILE_ID INT
PAYMENT_METHOD_ID INT
DATE TIMESTAMP
DISCOUNT DECIMAL
FISCAL
INVOICES
ID INT
PROFILE_ID INT
VERSION DECIMAL
XML TEXT
IDENTITY
STATES
ID INT
NAME VARCHAR(128)
ABBR VARCHAR(2)
CITIES
ID INT
STATE_ID INT
NAME VARCHAR(256)
ADDRESSES
ID INT
CITY_ID INT
STATE_ID INT
STREET VARCHAR(1024)
DISTRICT VARCHAR(1024)
ZIPCODE VARCHAR(128)
PRESCRIPTIONS
ID INT
PROFILE_OWNER_ID ...
PROFILE_MEDIC_ID INT
DATE DATE
PRESCRIPTION_INFOS
ID INT
PRESCRIPTION_CHARACTERISTIC_ID INT
PRESCRIPTION_ID INT
INFO DECIMAL
PRESCRIPTI...
ID INT
NAME VARCHAR(45)
ABBR VARCHAR(45)
PROFILES
ID INT
CLIENT_ID INT
PROFILE_SCHEMA_ID INT
ADDRESS_ID INT
CREATED_AT TIMESTAMP
CONTACT_TYPES
ID INT
NAME VARCHAR(128)
DESCRIPTION VARCHAR(1024)
CONTACTS
ID INT
PROFILE_ID INT
CONTACT_TYPE_ID INT
CONTACT_INFO VARCHAR(256)
CLIENTS
ID INT
CREATED_AT TIMESTAMP
PROFILE_SCHEMAS
ID INT
NAME VARCHAR(256)
DESCRIPTION VARCHAR(1024)
PROFILE_CHARACTERISTICS
ID INT
PROFILE_ID INT
SCHEMA_CHARACTERISTIC_ID INT
CHARACTERISTIC_INFO VARCHAR(1024)
TOKENS
ID INT
CLIENT_ID INT
TYPE VARCHAR(80)
IS_REVOKED TINYINT
TOKEN VARCHAR(256)
CREATED_AT TIMESTAMP
UPDATED_AT TIMESTAMP
SCHEMA_CHARACTERISTICS
ID INT
PROFILE_SCHEMA_ID INT
CHARACTERISTIC_ID INT
IS_NULLABLE TINYINT
CHARACTERISTICS
ID INT
NAME VARCHAR(256)
DESCRIPTION VARCHAR(1024)

Estratégias

- IDs
- RECEITAS
- PERFIS

Database Seed

Bibliotecas

- Faker
  - https://pypi.org/project/Faker/

- SQL Alchemy
  - https://pypi.org/project/SQLAlchemy/

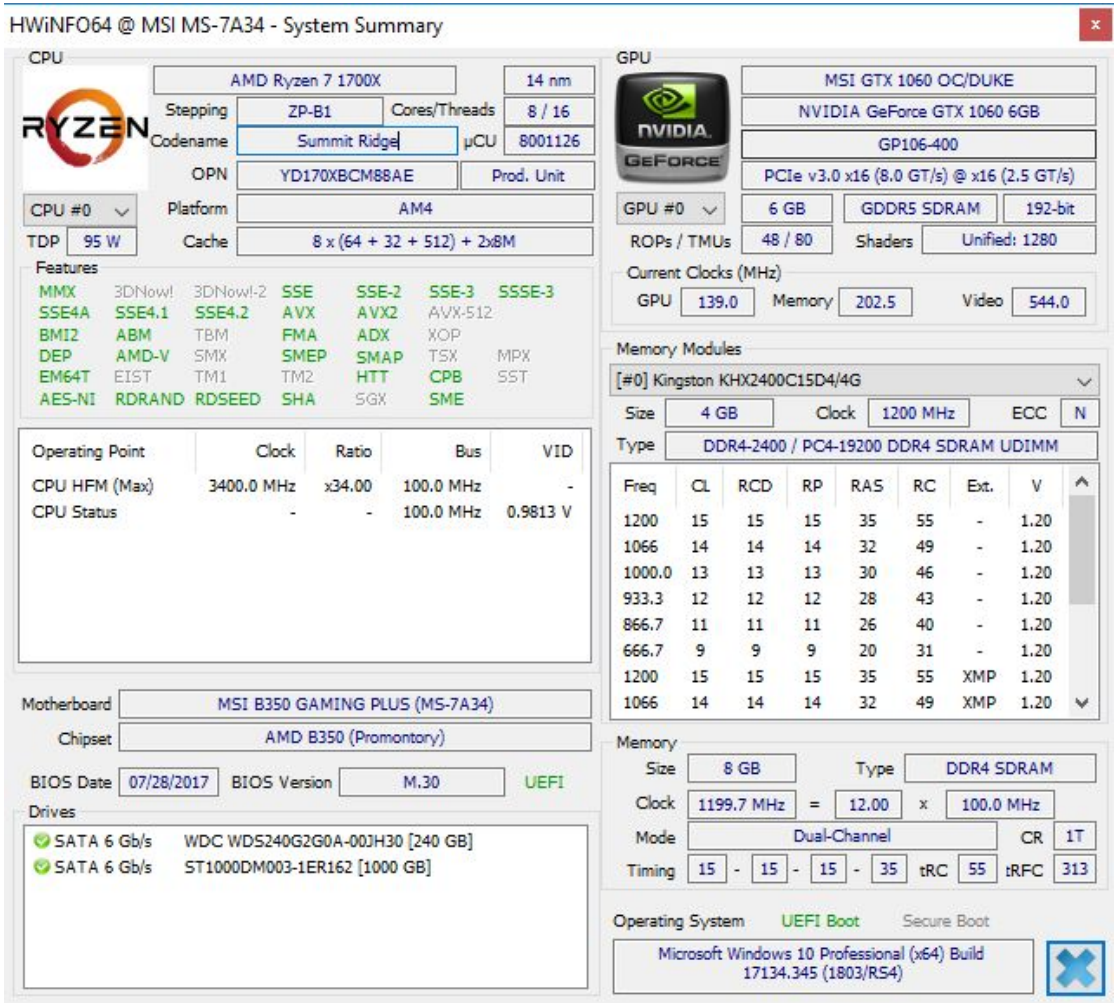
seed.py

C: ▸ Users ▸ Niddhogur ▸ Documents ▸ Projects ▸ trab-banco-dados ▸ artifacts ▸ seed ▸ seed.py ▸ ...

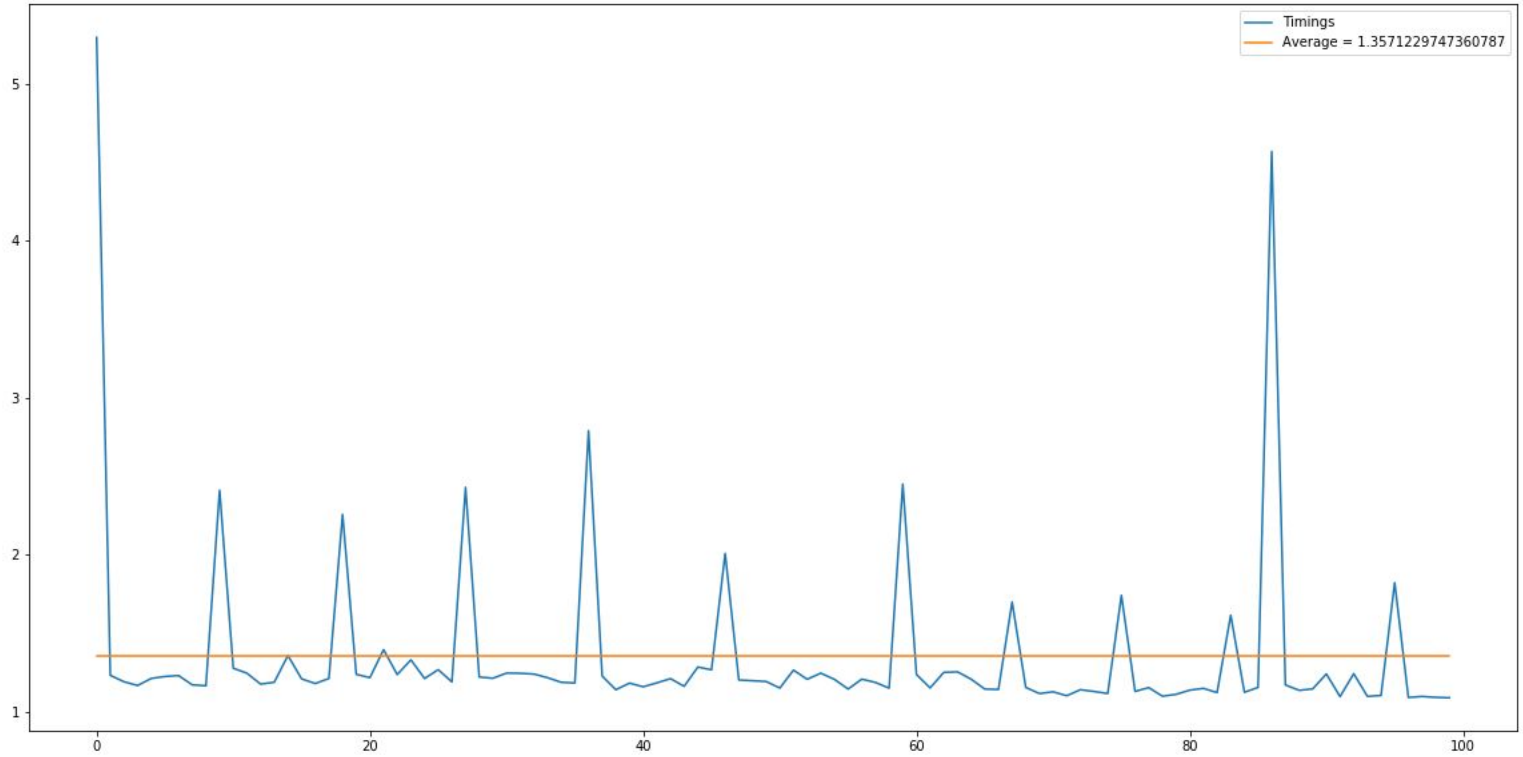
```python
from sqlalchemy.orm import Session, joinedload
from sqlalchemy.sql.expression import func, select
from src.engine import engine
from src import models
from faker import Faker
from faker.providers import address, misc, phone_number
import random

if __name__ == "__main__":
    faker = Faker()
    faker.add_provider(address)
    faker.add_provider(misc)

    sess = Session(bind=engine)

    sess.add_all([models.Characteristic(name=k) for k in ['Nome', 'Senha', 'Email']])
    sess.add_all([models.ProfileSchema(name=k) for k in ['Cliente', 'Funcionário', 'Empresa']])
    sess.add_all([models.Status(name=k) for k in ['Enviado', 'Disponível', 'Entrege']])
    sess.add_all([models.PaymentMethod(name=k) for k in ['Cartão', 'Dinheiro']])
    sess.add_all([models.ItemCategory(name=k) for k in ['Armação']])
    sess.add_all([models.ContactType(name=k) for k in ['Telefone']])
    sess.add_all([models.State(name=faker.state(), abbr=faker.state_abbr()) for k in range(10)])

    sess.commit()

    states = sess.query(models.State).all()
    sess.add_all([models.City(name=faker.city(), states=random.choice(states)) for k in range(50)])
    sess.commit()

    schemas = sess.query(models.ProfileSchema).all()
    characteristics = sess.query(models.Characteristic).all()
    cities = sess.query(models.City).options(joinedload('*')).all()
    contact_type = sess.query(models.ContactType).first()

    for schema in schemas:
        sess.add_all([
            models.SchemaCharacteristic(profile_schemas=schema, user_characteristics=k, is_nullable=False)
            for k in characteristics
        ])
```

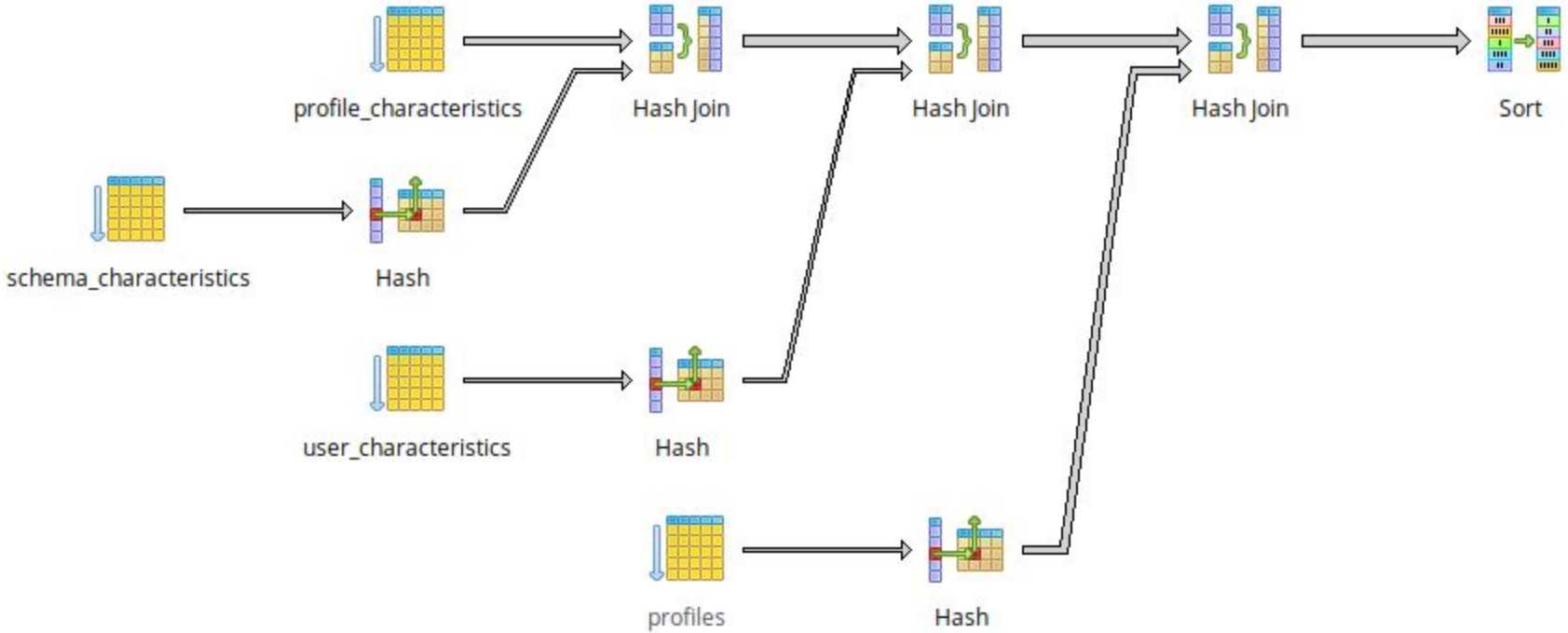
Análises

Configurações



[*]:
```
timings = []
for k in range(1000):
    t_0 = time.clock()

    session.execute("""
    SELECT * FROM CLIENTS
       JOIN PROFILES P ON CLIENTS.ID = P.CLIENT_ID
       JOIN ADDRESSES A ON P.ADDRESS_ID = A.ID
       JOIN CITIES C2 ON A.CITY_ID = C2.ID AND A.STATE_ID = C2.STATE_ID
       JOIN PROFILE_CHARACTERISTICS PC ON P.ID = PC.PROFILE_ID
       JOIN PROFILE_SCHEMAS PS ON P.PROFILE_SCHEMA_ID = PS.ID
       JOIN SCHEMA_CHARACTERISTICS SC ON PC.SCHEMA_CHARACTERISTIC_ID = SC.ID
       JOIN USER_CHARACTERISTICS C3 ON SC.CHARACTERISTIC_ID = C3.ID
       JOIN SALES S2 ON P.ID = S2.CLIENT_PROFILE_ID
       JOIN ITEMS_STOCK STOCK ON S2.ID = STOCK.SALE_ID
    WHERE CLIENT_ID = 'ED9BE65E-E705-4F46-A7B5-1BDA967E42E4';""").fetchall()

    timings.append(time.clock() - t_0)
```


Timings
Average = 1.3571229747360787
5
4
3
2
1
0
20
40
60
80
100



Rotina de Autorização

[3]:
```
ids = pd.read_sql_query("""
    SELECT *
    FROM CROSSTAB(
             'SELECT PROFILES.ID, UC.NAME, PC.CHARACTERISTIC_INFO
                        FROM PROFILES
                               JOIN PROFILE_CHARACTERISTICS PC ON PROFILES.ID = PC.PROFILE_ID
                               JOIN SCHEMA_CHARACTERISTICS SC ON PC.SCHEMA_CHARACTERISTIC_ID = SC.ID
                               JOIN USER_CHARACTERISTICS UC ON SC.CHARACTERISTIC_ID = UC.ID
                        WHERE UC.NAME = ''Email''
                          OR UC.NAME = ''Senha''
             ORDER BY PROFILES.ID, UC.NAME') AS PROFILE(ID UUID, EMAIL VARCHAR(1024), SENHA VARCHAR(1024));
""", engine)

ids.head()
```

[3]:

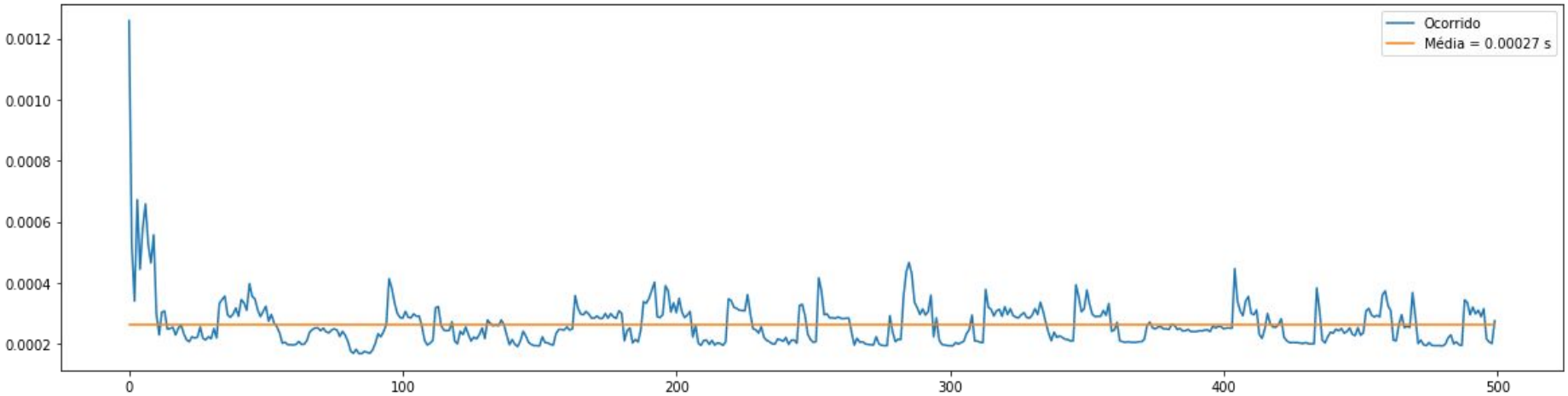
	id	email	senha
0	001ad5c1-7694-46e2-943a-aa5bf7df2920	lblake@gmail.com	^c&MdR_%0@
1	007b05cf-02c7-4d11-ae8c-c1913c81a55a	margaretlowe@fletcher.net	@m6Mo8isv^
2	00a12e7a-bc99-4ff2-a1c1-ebee5f8ddadd	fcaldwell@yahoo.com	6)+O9NWd)+
3	00b0173a-dd56-46b9-a1b9-52501ea7158e	sally52@yahoo.com	8@y4cTZdZ8
4	00f8c554-1248-4fbc-baed-dd30f360974a	brussell@buchanan-larson.net	Y#1PTm(qPI

[4]:
```
timings = []

for k in ids.index:
    pwd = ids.loc[k, 'senha']
    email = ids.loc[k, 'email']

    start = time.clock()
    engine.execute("SELECT * FROM AUTH(%s, %s);", pwd, email).fetchone()
    timings.append(time.clock() - start)

mean = pd.np.mean(timings)
```

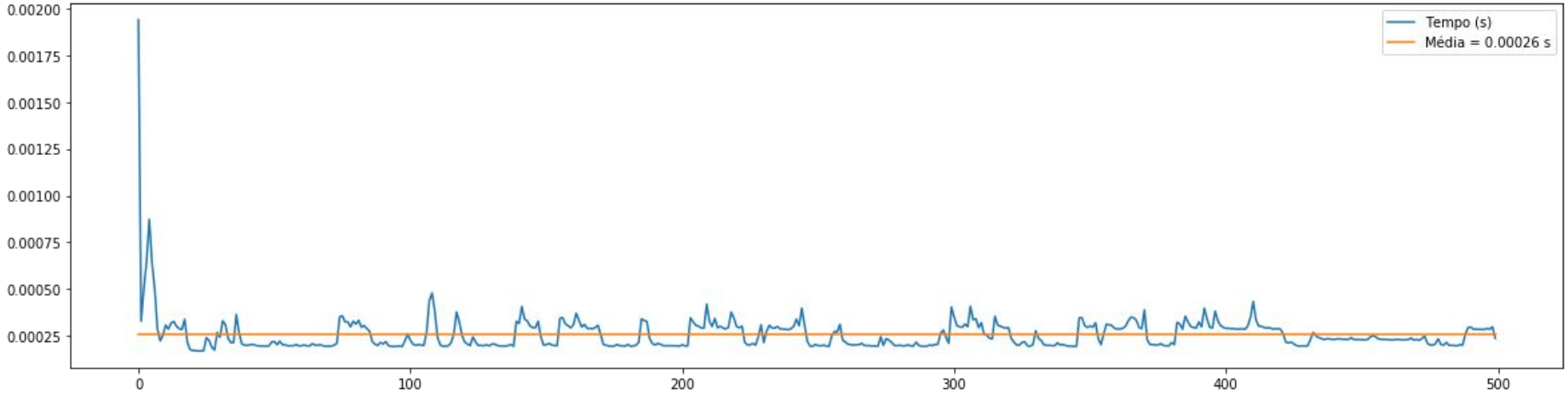
Método de medição das consultas de autorização

```
CREATE INDEX PROFILEID ON PROFILES USING BTREE (ID);
CREATE INDEX SCHEMAID ON SCHEMA_CHARACTERISTICS USING BTREE (ID);
CREATE INDEX CHARACTATERISTICID ON USER_CHARACTERISTICS USING BTREE (ID);
CREATE INDEX CHARACTATERISTICNAME ON USER_CHARACTERISTICS USING BTREE (NAME);
```

Índices



Consulta vs. Tempo (Sem Índice)



Consulta vs. Tempo (Com Índice)