Documentação Caso de Uso do UberEats

Sumário

Introdução ao Caso de Uso

O caso de uso UberEats foi desenvolvido para a Engenharia de Dados Academy como um cenário realista e abrangente para aprendizagem de engenharia de dados, ciência de dados e tecnologias de data lakehouse.

O modelo simula uma plataforma de entrega de comida que conecta três principais stakeholders: **clientes**, **restaurantes** e **entregadores**. Os dados são gerados usando o ShadowTraffic, uma ferramenta sofisticada de geração de dados sintéticos que cria aproximadamente 1TB de dados interconectados.

O ecossistema de dados abrange:

1. **Múltiplas fontes de dados**:
    - MSSQL (perfis de clientes)
    - MySQL (restaurantes e produtos)
    - PostgreSQL (entregadores e inventário)
    - MongoDB (perfis estendidos e itens de pedidos)

    - Kafka (eventos em tempo real)
2. **Fluxos de processos de negócio**:
    - Ciclo de vida completo do pedido, desde a colocação até a entrega
    - Processamento de pagamentos com diversos estados
    - Rastreamento de entregadores e rotas
    - Feedback e avaliações de clientes
3. **Casos de uso analíticos e de ML**:
    - Previsão de tempo de entrega
    - Recomendação de restaurantes
    - Detecção de fraudes
    - Precificação dinâmica
    - Extração automatizada de informações de faturas com GenAI

Este cenário serve como base para implementação de arquitetura moderna de dados (Bronze, Silver, Gold), modelagem Data Vault, e desenvolvimento de pipelines batch e streaming, proporcionando uma experiência prática completa que reflete desafios reais do mercado.

ShadowTraffic: Gerador de Dados

O ShadowTraffic é uma poderosa ferramenta de geração de dados sintéticos projetada para criar datasets realistas para desenvolvimento, testes e treinamento em engenharia de dados. No caso de uso do UberEats, o ShadowTraffic produz aproximadamente 1TB de dados que simulam com precisão as operações de uma plataforma de delivery de comida.

A ferramenta utiliza uma abordagem declarativa baseada em JSON para definir a estrutura e características dos dados. Cada entidade é configurada com geradores específicos que determinam como os valores serão criados - sejam eles sequenciais, aleatórios, baseados em distribuições estatísticas ou derivados de outras entidades.

Principais características:

- **Geradores variados**: Sequenciais (IDs), UUID, strings baseadas em templates, distribuições estatísticas, datas/timestamps
- **Integridade referencial**: Mecanismo de lookup que referencia valores de outras entidades
- **Máquinas de estado**: Para simular progressões temporais em processos como pedidos e pagamentos
- **Distribuições ponderadas**: Para criar cenários realistas com probabilidades específicas
- **Localização**: Suporte para gerar dados específicos para mercados como o brasileiro (CPF, CNPJ, endereços)
- **Formatação temporal**: Controle preciso sobre timestamps e sequências temporais

- **Destinos flexíveis**: Armazenamento em Azure Blob Storage em formato JSONL

O ShadowTraffic resolve o desafio fundamental de obter dados realistas para desenvolvimento e treinamento sem expor informações sensíveis de produção, permitindo que equipes de dados trabalhem com volumes e complexidades representativos do mundo real.

Arquitetura de Dados

A arquitetura de dados do UberEats é estruturada para suportar um processamento eficiente e escalável, seguindo o padrão Medallion (Bronze, Silver, Gold):

Fontes de Dados

- **MSSQL**: Dados de usuários
- **MySQL**: Dados de restaurantes, produtos e avaliações
- **PostgreSQL**: Dados de entregadores e inventário
- **MongoDB**: Dados de perfis estendidos e itens de pedidos
- **Kafka**: Eventos em tempo real (pedidos, status, pagamentos)

Camada Bronze (Raw)

- Ingestão de dados brutos sem transformações
- Preservação do formato original
- Adição de metadados de ingestão
- Histórico completo de dados de origem

Camada Silver (Cleansed)

- Dados validados e padronizados
- Resolução de identidades e estabelecimento de relacionamentos
- Implementação de regras de qualidade de dados
- Enriquecimento e transformação

Camada Gold (Business)

- Agregações orientadas ao negócio
- Modelos dimensionais para análises
- Métricas pré-calculadas
- Otimização para casos de uso específicos

Esta arquitetura permite um fluxo de dados que evolui de dados brutos para insights acionáveis, suportando tanto processamento em lote quanto streaming.

Domínios e Entidades

Domínio do Cliente

1. **Users (MSSQL)** Entidade fundamental de cliente armazenada no MSSQL contendo identificação pessoal e informações básicas, incluindo user_id sequencial como chave primária, UUID único, identificadores pessoais (CPF), componentes de nome (first_name, last_name), dados demográficos (birthday), informações de emprego (company_name, job), detalhes de contato (phone_number) e contexto geográfico (country).

2. **Users (MongoDB)** Informações estendidas de perfil de cliente armazenadas no MongoDB que complementam os dados básicos do MSSQL, contendo os mesmos user_id e UUID como identificadores, mas focando em atributos específicos para entrega como email para comunicações, phone_number alternativo, delivery_address detalhado para entrega precisa, e city e country para segmentação geográfica.

3. **Support Tickets** Entidade de rastreamento de casos de atendimento ao cliente armazenada no MongoDB que documenta problemas reportados, identificada por um ticket_id único, vinculada a um user_id e order_id específicos, categorizada por tipo de problema (Late Delivery, Wrong Item, Missing Item, Payment Issue, Other), contendo descrição detalhada, status atual (Open, In Progress, Resolved, Closed) e timestamps para monitoramento.

4. **Search History** Entidade de rastreamento de atividade de busca do usuário armazenada no Kafka que captura comportamentos de exploração na plataforma, incluindo search_id único, user_id que realizou a busca, query_text inserido, filtros específicos aplicados, número de resultados retornados, product_id que foi clicado e timestamp preciso da busca.

5. **Recommendations** Entidade de interação usuário-produto armazenada no MongoDB que captura atividades do motor de recomendação e respostas dos usuários, contendo event_id único, user_id que recebe recomendações, event_type específico (view, click, add_to_cart, recommendation_served), product_id recomendado e informações detalhadas de timestamp.

Domínio do Restaurante

6. **Restaurants** Entidade central de restaurante armazenada no MySQL que representa estabelecimentos de serviço de alimentação na plataforma, contendo restaurant_id sequencial como chave primária, UUID único, nome comercial, endereço completo com múltiplos componentes, city e country para organização geográfica, phone_number de contato, categorização de cuisine_type, horários de funcionamento, indicadores de

qualidade (average_rating, num_reviews) e identificador brasileiro de negócio (CNPJ).

7. **Products** Entidade de item de menu armazenada no MySQL representando pratos e itens oferecidos pelos restaurantes, identificada por um product_id único, vinculada a um restaurant_id específico, contendo nome do item, categorização (product_type, cuisine_type), atributos dietéticos (is_vegetarian, is_gluten_free), informações financeiras (unit_cost, price), características de sabor, tags promocionais e detalhes operacionais (prep_time_min).

8. **Menu Sections** Entidade de organização de menu de restaurante armazenada no MySQL que estrutura produtos em agrupamentos lógicos, identificada por um menu_section_id único, vinculada a um restaurant_id específico, contendo nome de seção (como "Starters" ou "Main Dishes"), texto descritivo explicando a seção e flag active para controlar visibilidade.

9. **Inventory** Entidade de rastreamento de disponibilidade de produto armazenada no PostgreSQL que monitora níveis de estoque para itens de restaurante, identificada por um stock_id único, vinculada a restaurant_id e product_id, contendo contagem quantity_available e timestamp last_updated para rastrear quando o inventário foi atualizado.

10. **Ratings** Entidade de feedback do cliente armazenada no MySQL que captura avaliações de qualidade de restaurantes, rastreada por rating_id sequencial, vinculada a um restaurante específico via seu CNPJ (restaurant_identifier), contendo uma pontuação de avaliação numérica entre 1-5 e timestamp de quando a avaliação foi enviada.

Domínio do Entregador

11. **Drivers** Entidade central de pessoal de entrega armazenada no PostgreSQL contendo identificação essencial e informações de veículo, incluindo driver_id sequencial como chave primária, UUID único, detalhes pessoais (first_name, last_name, date_birth), informações de localização (city, country), detalhes de contato (phone_number), informações de licenciamento (license_number) e especificações de veículo (vehicle_type, vehicle_make, vehicle_model, vehicle_year).

12. **Shifts** Entidade de período de trabalho do motorista armazenada no Kafka que rastreia quando os motoristas estão ativos na plataforma, identificada por shift_id único, vinculada a um driver_id específico, contendo informações temporais (start_time, end_time, shift_duration_min), status de disponibilidade, contexto de localização (city, region), categorização (shift_type), detalhes técnicos (device_os, login_method) e métricas de desempenho (num_orders, distance_covered_km, earnings_brl, shift_rating).

13. **Routes** Entidade de caminho de entrega armazenada no Kafka que registra a jornada do restaurante ao cliente, identificada por um route_id único, vinculada a um order_id e driver_id, contendo coordenadas geográficas (start_lat, start_lon, end_lat, end_lon), medição de distância (distance_km) e informações de tempo (estimated_duration_min, start_time, end_time).

14. **GPS** Entidade de localização de motorista em tempo real armazenada no Kafka que captura dados precisos de rastreamento durante entregas, identificada por um gps_id único, vinculada a um order_id específico, contendo coordenadas geográficas (lat, lon), dados de movimento (speed_kph, direction_deg, duration_ms), métricas de qualidade (accuracy_m), informações de altitude e dados detalhados de timestamp.

Domínio de Transações

15. **Orders** Entidade central de transação armazenada no Kafka representando compras de clientes, identificada por um order_id único, vinculada a cliente (user_key), restaurante (restaurant_key), avaliação (rating_key) e motorista (driver_key) através de chaves estrangeiras respectivas, contendo informações temporais (order_date), dados financeiros (total_amount) e referência de pagamento (payment_key).

16. **Order Items** Entidade de item de linha armazenada no MongoDB que detalha os produtos específicos dentro de um pedido, identificada por um order_item_id único, vinculada a um order_id e restaurant_id específicos, contendo informações de produto (product_id, product_name, product_type, cuisine_type, atributos dietéticos), detalhes de personalização (modifiers), informações de quantidade e dados de preço (unit_price, discount_applied, subtotal).

17. **Payments** Entidade de transação financeira armazenada no Kafka que gerencia trocas monetárias, identificada por um payment_id único, vinculada a um pedido específico (order_key), contendo informações financeiras (amount, currency, platform_fee, provider_fee, tax_amount, net_amount), detalhes de pagamento (method, provider, informações de cartão), rastreamento de status de transação, informações de reembolso e timestamps.

18. **Status** Entidade de rastreamento de progresso do pedido armazenada no Kafka que implementa uma máquina de estado para gerenciamento do ciclo de vida do pedido, identificada por um status_id sequencial, vinculada a um pedido específico (order_identifier), contendo um objeto de status estruturado com campos de nome e timestamp que progride através de uma sequência definida (Order Placed → In Analysis → Accepted → Preparing → Ready for Pickup → Picked Up → Out for Delivery → Delivered → Completed).

19. **Receipts** Entidade de documentação de transação armazenada no Kafka que representa o registro final de pedidos completados, identificada por um receipt_id único, vinculada

a order_id e payment_id, contendo informações de resumo financeiro (total_amount), resumo da composição do pedido (item_count) e timestamp de geração do recibo.

20. **Events** Entidade de ciclo de vida de pagamento armazenada no Kafka que rastreia a progressão de estado do processamento de pagamento, identificada por um event_id único, vinculada a um payment_id específico, contendo um objeto de evento estruturado que captura tanto o event_name quanto o timestamp preciso enquanto os pagamentos fluem através de uma máquina de estado definida (created → authorized → captured → succeeded/refunded → settled → closed).

Relacionamentos entre Entidades

A arquitetura de dados do UberEats é baseada em complexos relacionamentos entre entidades que permitem o funcionamento integrado da plataforma. Estes relacionamentos podem ser visualizados através de seus domínios:

Relações entre Domínios

1. **Cliente → Transações**
    - Clientes colocam pedidos, criando a entidade central Orders
    - Clientes interagem com Support Tickets para resolver problemas
    - Perfis de clientes são usados para personalizar recommendations
2. **Restaurante → Transações**
    - Restaurantes fornecem Products que se tornam Order Items
    - Restaurantes processam pedidos, gerando atualizações de Status
    - Ratings são vinculados a Restaurants com base na experiência de pedido
3. **Transações → Entregador**
    - Orders são atribuídos a Drivers para entrega
    - Status de entrega é rastreado via GPS e atualizações de Status
    - Routes são criadas para otimizar a entrega
4. **Fluxo Circular**
    - O Domínio de Transações atua como hub central, conectando todos os outros domínios
    - A entidade Orders funciona como ponto de integração entre Cliente, Restaurante e Entregador

Mecanismos de Relacionamento

O ShadowTraffic mantém a integridade referencial entre entidades através de lookups:

```
"user_key": {
   "_gen": "lookup",
   "container": "owshq-shadow-traffic-uber-eats",
   "keyPrefix": "mssql/users/",
   "path": ["data", "cpf"]
}
```

Este mecanismo garante que as chaves estrangeiras referenciem chaves primárias válidas em todo o conjunto de dados, criando um ecossistema de dados coeso.

Fluxos de Processos

Ciclo de Vida do Pedido

O ciclo de vida do pedido no UberEats segue uma progressão linear através de vários estados, cada um gerenciado por diferentes stakeholders:

1. **Order Placed** (Cliente)
    - Cliente navega pelo menu
    - Seleciona itens
    - Fornece endereço de entrega
    - Finaliza pagamento
2. **In Analysis** (Sistema)
    - Sistema valida o pedido
    - Verifica disponibilidade
    - Processa pagamento
3. **Accepted** (Restaurante)
    - Restaurante recebe o pedido
    - Revisa os detalhes
    - Confirma disponibilidade
    - Aceita o pedido
4. **Preparing** (Restaurante)
    - Restaurante inicia o preparo
    - Prepara os itens
    - Embala os alimentos
    - Atualiza status
5. **Ready for Pickup** (Restaurante)
    - Restaurante completa o pedido
    - Finaliza embalagem

    - Coloca para retirada
    - Marca como pronto
6. **Picked Up** (Entregador)
    - Entregador chega ao restaurante
    - Verifica o pedido
    - Coleta os alimentos
7. **Out for Delivery** (Entregador)
    - Entregador segue a rota
    - Compartilha localização GPS
    - Dirige até o endereço
8. **Delivered/Completed** (Cliente)
    - Cliente recebe o pedido
    - Confirma a entrega
    - Fornece avaliação
    - Transação completa

Cada transição de estado é registrada com timestamp na entidade Status, permitindo análise detalhada do desempenho e identificação de gargalos.

Fluxo de Processamento de Pagamentos

O processamento de pagamentos segue uma máquina de estado própria, implementada na entidade Events:

1. **Created**
    - Novo registro de pagamento
    - Vinculado ao pedido
    - Método de pagamento registrado
    - Timestamp inicial
2. **Authorized**
    - Cartão validado
    - Fundos disponíveis verificados
    - Código de aprovação gerado
    - Atraso de 1-3s
3. **Captured**
    - Fundos reservados
    - Pagamento bloqueado
    - Pronto para processamento
    - Atraso de 1-2s
4. **Bifurcação**:
    - **Succeeded** (90% dos casos)
        - Pagamento bem-sucedido
        - Progride para liquidação
    - **Refunded** (10% dos casos)
        - Pagamento reembolsado

        - Progride para fechamento
5. **Settled** (após Succeeded)
    - Fundos transferidos
    - Atraso de 2-5s
6. **Closed**
    - Transação finalizada
    - Estado terminal do pagamento
    - Atraso de 1-2s

O fluxo cria um registro completo e auditável de cada transação financeira, com timestamps precisos para cada transição de estado.

Casos de Uso Analíticos e ML

O conjunto de dados do UberEats suporta diversos casos de uso analíticos e de machine learning:

1. Previsão de Tempo de Entrega

Descrição: Estimar com precisão o tempo desde o pedido até a entrega.

Dados Utilizados:

- Pedidos históricos e suas durações
- Distância entre restaurante e cliente
- Tempos de preparo dos restaurantes
- Tipo de veículo do entregador
- Condições de tráfego (derivadas de GPS)

Abordagem ML:

- Algoritmos como XGBoost ou Random Forest
- Features de distância, hora do dia, tempo de preparo
- Treinamento com dados históricos

Benefício: Melhora a experiência do cliente com estimativas precisas.

2. Recomendação de Restaurantes

Descrição: Sugerir restaurantes relevantes baseados nas preferências do usuário.

Dados Utilizados:

- Histórico de pedidos
- Ratings dados

- Preferências de culinária
- Comportamento de usuários similares

Abordagem ML:

- Filtragem colaborativa
- Filtragem baseada em conteúdo
- Sistemas híbridos

Benefício: Aumenta conversão e satisfação com descobertas personalizadas.

3. Detecção de Fraudes

Descrição: Identificar transações potencialmente fraudulentas.

Dados Utilizados:

- Padrões de pagamento
- Histórico da conta
- Anomalias de localização
- Características de pedidos

Abordagem ML:

- Algoritmos de detecção de anomalias
- Modelos de classificação supervisionada
- Sistemas baseados em regras combinados com ML

Benefício: Protege tanto clientes quanto a plataforma de atividades fraudulentas.

4. Precificação Dinâmica

Descrição: Ajustar taxas de entrega com base em demanda e oferta.

Dados Utilizados:

- Níveis atuais de demanda
- Disponibilidade de entregadores
- Condições climáticas
- Eventos especiais

Abordagem ML:

- Aprendizado por reforço
- Modelagem de elasticidade
- Algoritmos de ajuste em tempo real

Benefício: Otimiza a eficiência do marketplace durante períodos de pico.

5. Extração de Dados de Faturas com GenAI

Descrição: Automatizar a extração de informações de faturas.

Dados Utilizados:

- Imagens e PDFs de faturas
- Dados históricos para validação
- Informações de pedidos

Abordagem:

- Visão computacional para análise de documentos
- Grandes modelos de linguagem para extração de informações
- Validação de dados estruturados

Benefício: Reduz processamento manual e melhora precisão.

Modelagem Data Vault

A arquitetura do UberEats é adequada para modelagem Data Vault, uma abordagem que oferece flexibilidade, escalabilidade e auditabilidade:

Hubs (Chaves de Negócio)

- **Hub_Customer** (cpf)
- **Hub_Restaurant** (cnpj)
- **Hub_Driver** (license_number)
- **Hub_Order** (order_id)
- **Hub_Product** (product_id)

Links (Relacionamentos)

- **Link_Order_Customer** (conecta pedidos a clientes)
- **Link_Order_Restaurant** (conecta pedidos a restaurantes)
- **Link_Order_Driver** (conecta pedidos a entregadores)
- **Link_Order_Item** (conecta pedidos a produtos)
- **Link_Restaurant_Product** (conecta restaurantes a seus produtos)

Satellites (Contexto)

- **Sat_Customer_Profile** (detalhes e histórico do cliente)
- **Sat_Restaurant_Details** (informações do restaurante)

- **Sat_Order_Status** (histórico de status com timestamps)
- **Sat_Payment_Details** (informações e histórico de pagamento)
- **Sat_Driver_Performance** (métricas e avaliações do entregador)

Esta abordagem permite:

- Rastreamento de mudanças ao longo do tempo
- Flexibilidade para evoluir o modelo
- Manutenção de histórico completo
- Auditabilidade de ponta a ponta

Conclusão

O conjunto de dados do UberEats gerado pelo ShadowTraffic proporciona um ambiente de aprendizado abrangente e realista para engenharia de dados, ciência de dados e tecnologias modernas de lakehouse.

A riqueza e complexidade das entidades, distribuídas em múltiplos sistemas e interconectadas através de relações bem definidas, simula os desafios reais enfrentados por plataformas de delivery. Os fluxos de processos, como o ciclo de vida do pedido e o processamento de pagamentos, refletem operações do mundo real com padrões temporais autênticos.

Esta arquitetura de dados suporta diversos casos de uso analíticos e de ML, permitindo aos estudantes aplicar técnicas modernas para resolver problemas reais de negócio. A adequação à modelagem Data Vault e à arquitetura Medallion (Bronze, Silver, Gold) oferece oportunidades para implementar as melhores práticas de design de data warehouse moderno.

Em suma, este modelo de dados representa um recurso educacional valioso que prepara os profissionais para os desafios do mercado atual de dados, combinando volumetria, complexidade e casos de uso relevantes em um único ambiente integrado.