UFRJ
UNIVERSIDADE FEDERAL
DO RIO DE JANEIRO

Relatório de Desenvolvimento

1 - Scripts ETL para carga no Data Mart

MAE016 - Programação, IA e Banco de Dados - 2025.2

Professor: Milton R Ramirez

Alunos:

Erick de Oliveira Pessoa *(Eng. Eletrônica e Computação)* **DRE:** 119180127

Gabriel Santa Bárbara Rodrigues Souza *(Eng. Nuclear)* **DRE:** 123356007

Iagor Libianno G. Crivellari *(Eng. Eletrônica e Computação)* **DRE:** 122061374

Juan Daniel Teixeira da Fonseca *(Eng. Controle e Automação)* **DRE:** 119030542

Márcio Melchiades Nascimento *(Eng. Eletrônica e Computação)* **DRE:** 122157036

Rafael Lúcio Martins *(Eng. Computação e Informação)* **DRE:** 120089122

Resumo (Abstract):

Este relatório documenta a etapa de desenvolvimento inicial do Data Mart da Semana de Integração Acadêmica (SIAc) da UFRJ. O foco desta fase consistiu na implementação da arquitetura de dados (Modelagem Dimensional), configuração do ambiente de desenvolvimento versionado e construção dos pipelines de Engenharia de Dados (ETL) utilizando a linguagem Python. O trabalho abrange a modelagem do esquema estrela em SQL, a estruturação do repositório Git e o desenvolvimento dos scripts de Extração e Transformação de dados a partir de documentos PDF não estruturados. Adicionalmente, discute-se a estratégia de infraestrutura, justificando a opção por um ambiente de banco de dados local em detrimento da nuvem nesta etapa inicial, visando a segurança de custos e a agilidade no desenvolvimento.

Palavras-chave: Engenharia de Dados, ETL, Python, Modelagem Dimensional, PostgreSQL, SIAc UFRJ.

1. Introdução

O projeto do Repositório de Dados PBD-SIAc tem como objetivo centralizar, estruturar e democratizar o acesso aos dados da produção acadêmica apresentada na Semana de Integração Acadêmica da UFRJ. Partindo de dados brutos e não estruturados (arquivos PDF dos Cadernos de Resumos) e dados semi-estruturados (HTML), a equipe de desenvolvimento iniciou a construção de um Data Mart analítico.

Este relatório detalha a execução das tarefas técnicas vinculadas às User Stories **U-PBI-01 (Infraestrutura e Modelagem)** e **U-PBI-02 (Scripts ETL),** descritas no PDF **Product Backlog (Projeto SIAc Data Mart)**. Descrevemos as escolhas arquiteturais para o banco de dados, as bibliotecas utilizadas para a mineração de texto e as soluções encontradas para normalizar os dados acadêmicos.

2. Desenvolvimento

O desenvolvimento foi guiado pela metodologia ágil, dividindo o escopo em tarefas técnicas menores. Abaixo, detalhamos a implementação de cada componente.

2.1. Modelagem de Dados e Scripts SQL (Tarefa 1.1)

2.1.1. Adoção do Esquema Estrela para o Data Mart

A decisão fundamental na Tarefa 1.1 foi adotar o paradigma de Modelagem Dimensional e, especificamente, o Esquema Estrela. Este modelo foi escolhido por ser o padrão da indústria para sistemas de Data Warehousing e Business Intelligence (BI), priorizando a performance de consulta em detrimento da redundância de dados.

O objetivo é otimizar o tempo de resposta para relatórios e análises complexas, garantindo que os requisitos de saída do projeto, como os relatórios (R-01 a R-03) e os Dashboards (D-01 e D-02), possam ser processados com eficiência. O Esquema Estrela alcança essa eficiência ao desnormalizar a estrutura, permitindo que a maioria das consultas exija apenas uma única operação JOIN entre a Tabela Fato central e as Tabelas Dimensão periféricas, em vez de múltiplas junções complexas características de um modelo relacional transacional.

2.1.2. Estrutura e Justificativa da Tabela Fato e Dimensões

A Tabela Fato_Resumo foi definida como o centro do Data Mart, pois representa o evento mensurável principal do projeto: a submissão e o registro de um trabalho. A Tabela Fato contém as chaves estrangeiras (Foreign Keys) que a

conectam a todas as dimensões, além da métrica essencial contagem_resumo (um valor não-nulo de 1), que é totalmente aditiva e permite agregações simples para responder à pergunta "Quantos trabalhos existem?".

As dimensões foram selecionadas em alinhamento direto com os requisitos de análise do domínio SIAC:

- **Dim_Tempo:** É uma dimensão essencial em BI, responsável por isolar todos os atributos baseados em tempo (ano, mês, dia da semana, etc.). Isso permite análises de tendências, sazonalidade e comparação de períodos sem a necessidade de cálculos complexos.
- **Dim_Pessoa:** Agrupa os atributos dos autores e coautores (nome completo, e-mail, ORCID), facilitando a segmentação por contribuições individuais ou grupos de pesquisa.
- **Dim_Trabalho:** Descreve o próprio objeto do evento (título, tipo de trabalho, área de conhecimento), permitindo aos usuários analíticos filtrar e agrupar resultados por características do conteúdo.
- **Dim_Local:** Captura afiliações e procedência geográfica (instituição, cidade, país), crucial para análises de impacto e distribuição regional ou institucional dos trabalhos.

2.1.3. Escolhas Técnicas e Portabilidade do DDL

Os scripts SQL DDL foram escritos utilizando sintaxe padrão e recursos amplamente suportados, garantindo a máxima portabilidade do código. Essa portabilidade é uma escolha técnica que permite que o esquema seja implantado em praticamente qualquer SGBDR, como PostgreSQL, MySQL ou até mesmo em serviços gerenciados na nuvem (AWS RDS ou Google Cloud SQL), conforme previsto no planejamento inicial do projeto. Uma escolha técnica chave implementada no DDL é o uso de Chaves Surrogadas (ex: id_tempo, id_pessoa, id_fato), definidas como SERIAL PRIMARY KEY. As chaves surrogadas são inteiros sequenciais gerados pelo sistema que substituem as chaves de negócio no Data Mart. Essa prática é crítica para a estabilidade do modelo, pois isola a Tabela Fato de quaisquer alterações futuras nas chaves originais dos dados de origem, garantindo a integridade referencial ao longo do ciclo de vida do Data Mart.

2.2. Estrutura do Repositório e Versionamento (Tarefa 1.2)

A organização do código fonte foi estabelecida utilizando o **Git** e hospedada no **GitHub**, seguindo boas práticas de desenvolvimento Python.

- **Estrutura de Pastas:** Separamos os scripts de ETL (*src/etl*), os arquivos de definição de banco (*src/sql*) e os dados brutos/processados *(data/raw, data/processed*), garantindo que dados sensíveis ou pesados não fossem versionados (uso de *.gitignore*), de acordo com a seguinte estrutura:

```
├── data/              # Dados brutos ou intermediários
├── docs/              # Documentação e relatórios
├── etl_scripts/       # Scripts de ETL (Python)
├── sql_schemas/       # Scripts SQL (DDL e DML)
├── dashboards/        # Dashboards e relatórios visuais
├── .gitignore         # Arquivos ignorados pelo Git
└── README.md          # Descrição geral do projeto
```

2.3. Infraestrutura de Banco de Dados (Tarefa 1.3)

Originalmente, a tarefa 1.3 previa o provisionamento de um banco de dados na nuvem (AWS RDS). No entanto, durante a fase de planejamento da Sprint, a equipe optou por **revisar esta estratégia**, dada a inexperiência de todos os membros em conhecimentos de *Cloud Computing,* e manter o banco de dados em ambiente local (Localhost/Docker) nesta fase.

Motivação:

1. **Curva de Aprendizado:** A equipe identificou a necessidade de maior maturidade nos conceitos de *Cloud Computing* e configuração de VPC/Security Groups antes de expor dados.
2. **Mitigação de Custos:** O risco de cobranças acidentais ("Bill Shock") por configurações incorretas na AWS (ex: provisionamento fora do Free Tier) foi considerado alto.
3. **Agilidade:** O uso de um container Docker local com PostgreSQL eliminou a latência de rede durante os testes intensivos dos scripts de carga.

Esta decisão não compromete a arquitetura, pois os scripts de conexão utilizam a biblioteca *SQLAlchemy*, permitindo que a "string de conexão" seja alterada para um endpoint na nuvem no futuro sem reescrita de código.

2.4. Pipeline de ETL e limpeza (Tarefas 2.1, 2.2, 2.3 e 2.4)

Esta etapa envolveu a transferência dos "Cadernos de Resumos" extraídos do site da SIAc em formato PDF, a padronização e transformação dos dados neles e o salvamento do resultado em arquivos *json*. Foram criados três arquivos Python para a execução da tarefa, sendo um para a extração dos dados, um para a transformação e um para guardar código utilitário.

2.4.1. Extração (Extract)

O processo de extração dos cadernos de resumos do SIAC/UFRJ foi implementado no arquivo *extract.py* com base em um conjunto de decisões que garantem robustez, paralelismo e adaptação às diferentes estruturas presentes no site da instituição. A estratégia adotada consistiu em tratar separadamente duas fontes distintas de informação: os cadernos de edições antigas (2009-2016) e os cadernos de edições mais recentes (2017-2025) da SIAc, uma vez que cada uma utiliza um layout HTML diferente a partir da página online do evento.

Para realizar as requisições HTTP de forma eficiente e paralela, optou-se por utilizar a biblioteca *httpx* em conjunto com *asyncio*, permitindo que várias operações de rede ocorram simultaneamente. Esse padrão foi aplicado tanto à coleta dos links quanto ao download dos arquivos PDF propriamente ditos.

A primeira parte da extração é conduzida por uma função nomeada como *extrair_links_cadernos_antigos*. Nela, o código solicita a página principal do SIAC e processa seu conteúdo com o BeautifulSoup. Como as edições mais antigas são apresentadas por meio de botões HTML, a função localiza todos os elementos <*button*> associados à classe CSS utilizada pelo site. Esses botões contêm, no atributo *onclick*, uma chamada JavaScript que indica o link direto para o PDF. Não havendo uma API formal, a solução encontrada foi empregar expressões regulares para capturar a URL embutida nessa chamada. Cada PDF recebe um nome padronizado — “Cadernos Resumos {ano}.pdf” — e é armazenado em um dicionário para processamento posterior.

Para as edições mais recentes, a estrutura da página é diferente, exigindo uma segunda função dedicada: *extrair_links_cadernos_novos*. Nesse caso, os anos estão organizados em blocos com um atributo *data-year*, e os links dos cadernos aparecem em elementos internos do tipo cartão (“*siac-card*”). A lógica consiste em navegar pelos cartões, localizar as entradas individuais de cada caderno, extrair novamente o link via expressão regular no atributo *onclick* e gerar um nome de arquivo que combine o título do caderno com o respectivo ano. Essa separação entre funções garante clareza e reduz acoplamento entre o código e possíveis mudanças futuras no site.

Após reunir os links provenientes dos dois formatos distintos, a função main consolida todos os nomes e URLs dos PDFs em um único dicionário, que são enviados a uma função para baixar os arquivos. O download em si ocorre na função *baixar_pdf*. Optou-se por realizar chamadas HTTP assíncronas, verificando o status da resposta e descartando arquivos suspeitos — por exemplo, PDFs muito pequenos, que podem indicar links quebrados ou resultados incorretamente extraídos. Cada arquivo válido é salvo no diretório *data/*, definido explicitamente no

início da execução. O uso da função *asyncio.gather* permite que todos os downloads ocorram em paralelo, reduzindo consideravelmente o tempo total de execução.

2.4.2. Transformação (Transform):

O processo de transformação dos dados extraídos dos cadernos do SIAC/UFRJ foi desenvolvido com o objetivo de converter documentos PDF heterogêneos, historicamente inconsistentes e sem um padrão oficial de formatação, em dados estruturados fáceis de manipular. O script responsável por essa etapa foi posto no arquivo *transform.py* e dedica-se a limpar o texto, identificar padrões recorrentes e reconstruir de forma coerente informações como título, autores, unidade acadêmica, centro, resumo e formas de contato.

Para dar início ao processamento, o script percorre todos os PDFs baixados previamente para o diretório *data/*. Cada arquivo é lido com auxílio da biblioteca PyPDF2, e o texto extraído página a página é concatenado em uma única string. A extração de texto de PDFs costuma gerar ruído, como caracteres quebrados, acentuação irregular ou quebras de linha inesperadas. Para lidar com isso, todo o conteúdo passa por uma função de limpeza — *limpar_texto* — definida no módulo *utils_text.py*. Essa função aplica uma série de normalizações: correções automáticas de codificação via ftfy, padronização Unicode, redução de múltiplos espaços e uniformização dos identificadores dos trabalhos, garantindo que elementos como “T-123” ou “T - 123” sejam convertidos para um formato único e fácil de detectar por expressões regulares.

Um aspecto fundamental do script é a necessidade de lidar com duas estruturas distintas de conteúdo: a do ano de 2016, que segue um padrão baseado em seções de “ARTIGO”, e a dos demais anos, que utilizam uma formatação mais recente. Por isso, a decisão foi separar o processamento em duas funções independentes: *parse_artigos* para o caso atípico de 2016 e *parse_trabalhos* para todos os outros anos.

A função parse_artigos opera identificando blocos delimitados pela palavra-chave “ARTIGO:”. Dentro de cada bloco, são extraídos campos como título, resumo e participantes, usando expressões regulares guiadas por um mecanismo auxiliar chamado extrair. Essa função foi projetada para isolar o texto entre marcadores conhecidos, independentemente da quantidade de linhas ou espaçamento interno, facilitando um parsing robusto mesmo quando os PDFs apresentam pequenas diferenças internas.

Por outro lado, a função *parse_trabalhos* lida com uma estrutura consideravelmente mais irregular. Os trabalhos são identificados por padrões como “T-123”, e o texto correspondente é dividido de acordo com esses marcadores. Dentro de cada bloco, o script reconstrói manualmente o título, que normalmente aparece nas linhas imediatamente após o identificador, acumulando todas as linhas até o trecho onde aparecem as seções da unidade acadêmica ou do centro. A identificação da unidade, do centro e da linha de contato é realizada por buscas simples por prefixos padronizados, como “Unidade:” ou “Centro:”. Essa abordagem reflete a decisão de priorizar legibilidade e previsibilidade, em vez de depender de heurísticas mais frágeis.

A extração de autores foi um dos pontos mais delicados do processamento, pois o formato varia significativamente entre anos. Algumas linhas contêm múltiplos autores concatenados sem separadores adequados, enquanto outras trazem marcações como “- Docente”, “- Técnico” ou outros papéis institucionais. Para resolver isso, o script utiliza a função *separa_nomes_autores*, que aplica uma estratégia baseada na detecção de transições entre nomes (como passagens de letras minúsculas para maiúsculas), remoção de sufixos desnecessários e padronização dos nomes via a função *padronizar_autor*. Após essa limpeza, os nomes são capitalizados adequadamente, garantindo uma representação homogênea e adequada para posterior análise ou tabulação.

O resumo de cada trabalho é reconstruído como o texto subsequente às seções de autores e antes da linha de contato — trecho que costuma apresentar maior variação de posição e formato. Para garantir que apenas conteúdo textual relevante seja incluído no resumo, regras condicionais controlam a fronteira entre nomes de autores e início do resumo, baseando-se na extensão das linhas e na observação dos padrões típicos do documento.

Ao fim do processamento de cada PDF, o script produz um arquivo JSON específico para o ano do caderno, contendo todos os trabalhos devidamente estruturados. Esses arquivos são salvos na mesma pasta, com nomes como *caderno_parsed_2022.json*, possibilitando fácil reuso em etapas posteriores de análise, integração com bancos de dados ou construção de dashboards.

2.4.2. Carregamento (Load):

A tarefa de pipeline de script *Load* dos dados foi criada no Github e está em desenvolvimento, mas o membro do grupo responsável se encontrou enfermo nos últimos dias e não teve tempo hábil de finalizá-la. Mas a mesma continua em desenvolvimento, e logo estará na branch principal do projeto.

4. Conclusão

A equipe concluiu com êxito as etapas fundamentais de arquitetura e extração de dados. A modelagem em esquema estrela provê uma base sólida para os dashboards futuros, e os scripts de extração desenvolvidos conseguiram converter documentos PDF não estruturados em dados manipuláveis. Embora a infraestrutura em nuvem tenha sido postergada, a solução local via Docker/PostgreSQL mostrou-se funcional e adequada ao estágio atual de desenvolvimento, garantindo a entrega dos requisitos funcionais sem custos adicionais ou riscos de infraestrutura.