Paulo Baran

FileDB - Leitura e Manipulação de Arquivos XLSX através de Linguagem SQL

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: Ciência da Computação.

Orientador: André Pires Guedes.

Curitiba PR
2019

Lista de Figuras

2.1. Processo de Atuação Planning Service 6
2.2. Recepção e tratamento dos arquivos Planning Service 7
3.1. Fluxo de Execução FileDB 20
3.2. ETL de leitura FileDB 22
3.3. ETL de escrita FileDB 23
3.4 Arquivos de Produção 24
3.5. Arquivo de Produção (layout) 24
3.6. Prompt de comando da Aplicação 26

Lista de Acrônimos

API	Application Program Interface
CSV	Comma-Separated Values
DCL	Data Control Language
DDL	Data Definition Language
DML	Data Manipulation Language
DQL	Data Query Language
FDW	Foreign Data Wrapper
JDBC	Java Database Connectivity
ODBC	Open Database Connectivity
SGBD	Sistema Gerenciador de Banco de Dados
SGBDR	Sistema Gerenciador de Banco de Dados Relacional
SQL	Structured Query Language
VBA	Visual Basic for Applications
WAL	Write Ahead Log
XML	Extensible Markup Language

Sumário

1. Introdução **4**

2. Tratamento manual de dados em editores de planilhas XLSX **6**

2.1. A manipulação dos arquivos e como a Planning Service trabalha com as informações fornecidas pelos clientes 6

2.2. Busca de uma ferramenta para atendimento da demanda 8

3. FileDB **10**

3.1 Conceitos relacionados 10

3.1.1. Microsoft Excel 10

3.1.2 ETL 11

3.1.3. Sistemas Gerenciadores de Bancos de Dados 11

3.1.3.1. Banco de Dados Relacionais 12

3.1.3.1.1. SQL 13

3.1.3.2. ACID 13

3.1.4. Cache 14

3.1.5. Arquivos WAL 16

3.2. Tecnologias Aplicadas 16

3.2.1 SQLite 17

3.2.2 Python 17

3.3. Especificações da Aplicação 18

3.3.1 Fluxo de Execução da Aplicação **20**

3.3.1 Esquemas ETL de leitura e escrita de arquivos XLSX 22

3.3.3. Caso de uso 24

3.3.3.1. Utilizando o FileDB 26

4. Conclusão **28**

1. Introdução

O presente trabalho teve como objeto de estudo a empresa Planning Service, especializada em Transfer Pricing e inserida há mais de 12 anos no cenário contábil do país, onde se propõe a elaboração de um aplicativo específico para a recepção e manipulação dos dados fornecidos por seus clientes, já que o diferencial da empresa é em tecnologia e flexibilidade na prestação dos seus serviços, [1].

As empresas atendidas tem suas informações hospedadas em ERPs. ERP por sua vez, é a definição dada a um grande sistema que visa atender a necessidade de informação de todos os setores de uma determinada empresa em um único banco de dados, [2]. Porém, nem sempre isso é possível, ainda mais quando se trata de atendimento ao fisco brasileiro, considerando a complexidade por ele imposta, levando as empresas a contratar sistemas satélites especialistas para sanar as deficiências de seu ERP. A aderência a tais sistemas gera então a necessidade de migrar as informações do seu banco de dados para um banco de dados de terceiros.

Transfer Pricing é uma das tantas obrigações fiscais estabelecidas pelo fisco, sendo este para toda empresa multinacional que possui transações com empresas do mesmo grupo empresarial ou empresas situadas em paraísos fiscais, ou até mesmo empresas nacionais que possuam transações com empresas situadas em paraísos fiscais, [1].

Existem ERP's que conseguem atender a esta demanda de controle do Transfer Pricing, mas nem sempre com satisfação plena, pois o nível de complexidade é muito alto, e planilhas de dados paralelas acabam sendo necessárias.

A Planning Service é uma empresa especializada em tecnologia da informação e em Transfer Pricing, e realiza este trabalho de forma terceirizada para empresas que não possuam esta funcionalidade em seu ERP ou que não queira utilizar da forma em que se encontra disposta.

Para realização deste trabalho é necessária a obtenção de todas as operações comerciais e fabris da empresa contratante, como também informações provindas de outros países nos quais tal empresa possua subsidiárias instaladas.

A obtenção de 90% destas informações é feita através de informações padronizadas, através de arquivos magnéticos estabelecidos pelo fisco brasileiro como o arquivo SPED Fiscal de ICMS e IPI, além de arquivos XML da nota fiscal eletrônica.

Os 10% restantes das informações necessárias para geração do Transfer Pricing são obtidos através de planilhas de dados XLSX nos mais variados tamanhos e layouts disponíveis na contratante, sendo necessária a manipulação destas informações em editores de planilhas eletrônicas para se adequarem ao layout de importação no banco de dados da Planning Service.

O tratamento destas informações de forma manual é suscetível a erros e, quando há grande volumes de dados, estas planilhas se tornam lentas e improdutivas através do editores de planilhas convencionais, por maior que seja a capacidade de processamento e de memória dos computadores disponíveis na atualidade.

A situação acima elencada trata-se de um problema relacionado a um processo conhecido como ETL, que será detalhado na seção 3.1.2. A Planning Service identificou um gargalo para o desenvolvimento de seu trabalho relacionado a esse problema, levando a buscar uma solução ágil de forma a não interferir no custo do serviço prestado a seu cliente, como também não o obrigando a produzir um layout específico para que as informações pudessem ser migradas.

2. Tratamento manual de dados em editores de planilhas XLSX

O capítulo a seguir aborda a contextualização e a especificação do problema, de forma a identificar quais são as necessidades em relação ao que se está sendo proposto.

Da mesma forma, será abordada a busca realizada pela empresa em encontrar uma solução pronta para o atendimento da sua demanda.

2.1. A manipulação dos arquivos e como a Planning Service trabalha com as informações fornecidas pelos clientes

A Planning Service se propõe a receber a informações de seus clientes nos mais variados formatos e possui um processo estabelecido para realizar o cálculo de Transfer Pricing. A figura abaixo foi retirada do slide de apresentação da empresa e evidencia o processo de atuação relacionado à obtenção de dados dos clientes nas diferentes disposições.

Figura 2.1: Processo de Atuação Planning Service

A figura a seguir também foi retirada da apresentação da empresa e ilustra a recepção dos arquivos da contratante, onde no primeiro caso, há a obtenção dos dados de forma padronizada, extraindo-se dados cadastrais, operações de compra e venda e outras dos arquivos eletrônicos em formato padrão estabelecido pelo fisco brasileiro. O segundo e terceiro caso dizem respeito a informações a serem dispostas em planilhas já existentes na empresa contratante, que contenham as informações complementares necessárias para realização do trabalho.

Figura 2.2: Recepção e tratamento dos arquivos Planning Service.

Abaixo estão as principais necessidades de tratamento das informações identificadas de acordo com a forma que os clientes dispunham as informações:

1) Leitura de planilhas XLSX e conversão para um determinado layout para importação no sistema da Planning Service.
2) Relatórios emitidos pelos clientes são mensais, pois os sistemas dos clientes não permitem geração para um período superior a 1 mês. Assim, se torna

necessário consolidar as planilhas mensais em uma única, contendo todas as informações no layout de importação para o sistema.

3) Arquivos contendo informações referente aos processos fabris. Cliente disponibiliza um relatório com os apontamentos de produção e outro relatório com os insumos consumidos na ordem em períodos mensais. Consolidar planilhas para período anual e produzir um layout onde, na mesma linha deverá conter o produto produzido e o item consumido, ambos com quantidade de produção e de consumo respectivamente e, se houver mais de uma matéria prima para produção, deverá se repetir o código do produto produzido e a sua quantidade para todas as matérias primas consumidas na ordem. Neste segundo caso evidencia-se a necessidade de se fazer uma junção de dados das duas tabelas (insumos e produtos) para obtenção do relatório resultante.

2.2. Busca de uma ferramenta para atendimento da demanda

A partir das necessidades levantadas anteriormente, a Planning Service passou a procurar uma ferramenta que pudesse efetuar leitura de planilhas XLSX, com a possibilidade de manipulação destas a através da linguagem SQL.

Vislumbrava-se encontrar uma ferramenta que simulasse um gerenciador de banco de dados efetuando leitura diretamente nestas planilhas, para que então fosse possível a manipulação destas planilhas e a criação de planilhas num determinado layout de importação para o seu banco de dados.

A relação de ferramentas abaixo foi estudada:

1) Ferramentas que permitem carregar dados CSV ou XLSX para o banco de dados:
    - PgLoader, [3];
    - Rows, [4].

2) Ferramentas que permitem executar SQL diretamente sobre CSV:
    - Rows, [4];
    - QueryCSV, [5];
    - Q, [6];
    - TextQL, [7].
3) FDW (Foreign Data Wrapper) que permite fazer o PostgreSQL abrir arquivos de texto:
    - File_fdw, [8].

Limitações das ferramentas acima:

- Somente o Rows suporta XLSX e atende o quesito número um elencado acima. As demais ferramentas suportam apenas arquivos CSV;
- Os carregadores (1) e o FDW (3) precisam de um SGBD como o Postgres para carregar os dados.
- Todas elas (inclusive Rows) permitem uma sintaxe SQL do tipo “SELECT FROM WHERE” em um arquivo de cada vez.

Como o objetivo da Empresa é a simulação de um SGDB sobre planilhas XLSX, com possibilidades de manipulações das informações, assim como a geração destas informações resultantes em novas planilhas XLSX, demonstrou-se então vencida a busca por uma ferramenta que atendesse esta necessidade.

3. FileDB

Neste capítulo serão elencados os conceitos relacionados e a tecnologia utilizada para o desenvolvimento da aplicação. Será também apresentada a sua especificação e principais características, e por final será apresentado um caso de uso da aplicação.

3.1 Conceitos relacionados

Nesta seção serão descritos os conceitos envolvidos na elaboração da aplicação final, entre eles: Microsoft Excel, ETL, sistemas gerenciadores de bancos de dados e cache.

3.1.1. Microsoft Excel

Microsoft Excel é um programa para gerenciamento de planilhas eletrônicas desenvolvido pela Microsoft para os sistemas operacionais Windows, macOS, Android e iOS. Tem sido um programa de planilhas vastamente usado para essas plataformas, especialmente desde a versão 5 em 1993, e o Excel substituiu o Lotus como o padrão da indústria de sistemas gerenciadores de planilhas. Excel é parte do Microsoft Office, [9].

Microsoft Excel contém as funcionalidades básicas de todos os programas para gerenciamento de planilhas, usando uma tabela de células arranjadas em linhas numeradas e colunas com nomes de letras para organizar operações aritméticas e de manipulações de dados. Excel tem uma série de funções para atender às necessidades estatísticas e financeiras. E mais ainda, pode mostrar dados na forma de

gráficos e também permite o seccionamento de dados para visualizar dependências em vários fatores para diferentes perspectivas através das tabelas dinâmicas. Também oferece uma API para programação chamada VBA (Visual Basic for Applications).

3.1.2 ETL

ETL (Extract, Transform and Load) é um processo de integração de dados que consiste em recuperar dados de uma determinada origem e armazená-los em um determinado destino. É composto por 3 etapas, conforme abaixo, [10]:

- **Extraction:** a primeira etapa consiste em extrair os dados da origem, que geralmente encontram-se em arquivos ou bancos de dados;
- **Transformation:** é a segunda etapa do processo, consiste em transformar os dados que foram extraídos, de forma que eles se tornem compatíveis com o destino no qual serão armazenados, por exemplo, tornar eles compatíveis com o esquema do banco de dados de destino. Envolve uma série de regras de negócio para que haja compatibilidade de informação entre origem e destino;
- **Loading:** terceira e última etapa, consiste em carregar no banco de dados de destino os dados que foram manipulados na etapa anterior.

3.1.3. Sistemas Gerenciadores de Bancos de Dados

Um sistema gerenciador de banco de dados (SGBD) é um software que interage com usuários, outras aplicações e o banco de dados em si para capturar e analisar dados. Um SGBD de propósito geral permite a definição, criação, consulta, atualização e administração de bancos de dados, [11].

Alguns exemplos de sistemas gerenciadores de bancos de dados muito utilizados atualmente são: PostgreSQL [12], Oracle [13], MySQL [14], MariaDB [15], Microsoft SQL Server [16], Firebird [17], SQLite [18], IBM DB2 [19], entre outros.

Um banco de dados geralmente não é portável entre diferentes SGBDs, mas diferentes SGBDs podem interoperar ao usar padrões como SQL, ODBC e JDBC para permitir uma única aplicação funcionar com mais de um SGBD.

3.1.3.1. Banco de Dados Relacionais

Um banco de dados é uma coleção organizada de dados. Um banco de dados relacional, mais restritivamente, é uma coleção de schemas, tabelas, consultas, relatórios, views, e outros elementos. Projetistas de bancos de dados tipicamente organizam os dados para modelar aspectos da realidade de tal forma que suporte processos que requerem informação, tais como modelagem da disponibilidade de quartos em hotéis de forma que suporte encontrar um hotel com quartos vagos, [11].

Em um banco de dados relacional, os dados estão organizados em tabelas. Cada tabela consiste de colunas verticais (identificável pelo nome) e linhas horizontais. A célula é a unidade onde coluna e linha se interceptam. Uma tabela tem um número específico de colunas, mas pode ter qualquer número de linhas. Cada linha é identificada por um ou mais valores aparecendo em um subconjunto particular de colunas. Uma escolha específica de colunas que unicamente identifica linhas é chamado de chave primária. Sob o ponto de vista conceitual, uma tabela é muito semelhante a uma planilha do Excel.

3.1.3.1.1. SQL

SQL ou Structured Query Language é uma linguagem usada em programação e projetada para gerenciar dados contidos em um sistema gerenciador de bancos de dados relacionais. [22]

Originalmente baseado em Álgebra Relacional, SQL consiste de muitos tipos de comandos, os quais podem ser informalmente classificados como sublinguagens:

- **DQL**: Data Query Language (SELECT);
- **DDL**: Data Definition Language (CREATE TABLE, ALTER TABLE, DROP TABLE, etc);
- **DCL**: Data Control Language (GRANT, REVOKE);
- **DML**: Data Manipulation Language (INSERT, UPDATE, DELETE).

3.1.3.2. ACID

ACID é um conjunto de propriedades de bancos de dados transacionais cujo objetivo é garantir a validade mesmo em caso de erros, queda de energia, etc. Uma sequência de operações que satisfazem às propriedades ACID, e por isso podem ser percebidas como uma única operação lógica sobre os dados, é chamada de transação. Por exemplo, a transferência de fundos de uma conta bancária para outra, mesmo que envolva múltiplas operações como débito em uma conta e crédito em outra, é uma única transação. As propriedades ACID são, [23]:

- **Atomicidade**: Transações são frequentemente compostas de múltiplas operações. Atomicidade garante que cada transação é composta de uma única unidade, que ou é executada completamente com sucesso, ou falha completamente: se qualquer uma das operações que constitui uma transação

falha, a transação inteira falha e o banco de dados é intocado. Um sistema atômico precisa garantir atomicidade em toda e qualquer situação, incluindo quedas na energia e erros na aplicação.

- **Consistência**: Garante que uma transação pode trazer o banco de dados de um estado válido para outro, mantendo as invariantes do banco de dados: qualquer dado escrito no banco de dados deve ser válido de acordo com todas as regras definidas, constraints, triggers, etc. Isso previne corrupção de banco de dados que podem ser causadas por transações ilegais, mas não garante que uma transação é correta.
- **Isolamento**: Transações são frequentemente executadas concorrentemente (ler e escrever de múltiplas tabelas ao mesmo tempo). Isolamento garante que a execução concorrente de transações deixa o banco no mesmo estado que seria obtido se as transações fossem executadas sequencialmente. Isolamento é o principal objetivo do controle de concorrência; dependendo do método usado, os efeitos de uma transação incompleta podem não ser visíveis para outras transações.
- **Durabilidade**: Garante que uma vez que uma transação é finalizada, ela permanece finalizada mesmo no evento de uma falha no sistema (por exemplo queda na energia). Isso significa que transações completadas são armazenadas em memória não volátil.

3.1.4. Cache

Cache é um componente de hardware ou software que armazena dados para que requisições futuras para esses dados possam ser servidos mais rápido. Os dados armazenados em cache podem ser o resultado de um processamento anterior ou a

duplicata de dados armazenados em outro lugar. Um “cache hit” ocorre quando o dado requisitado pode ser encontrado na cache, enquanto que um “cache miss” ocorre quando não pode ser encontrado na cache. “Cache hits” são servidos ao ler dados da cache, o que é mais rápido do que recalcular o resultado de uma função muito lenta, por exemplo, ou buscar os dados em um armazenamento lento. Então, quanto mais requisições podem ser servidas à aplicação vindas da cache, mais rápido o sistema desempenha.

Para ser eficiente e permitir uso eficiente dos dados, caches precisam ser relativamente pequenas. Caches são usadas em muitas áreas da computação porque o padrão de acesso em aplicações típicas exibem localidade temporal (dado é requisitado novamente logo em seguida) e/ou localidade espacial (dados requisitados em sequência normalmente estão armazenados próximos um do outro).

Em um SGBD, a cache é uma região alocada pelo sistema em memória, também chamada de “memória compartilhada”. Quando um processo do banco de dados busca dados do disco, esses dados são também jogados na cache, para que o acesso a esses dados que futuramente possa ser feito por outros processos, seja mais rápido. Se o processo do banco de dados faz alguma alteração nos dados, essa alteração é feita na cache (dados alterados são marcados como “sujos”, para serem posteriormente armazenados no disco). Futuros processos enxergam os dados já alterados na cache, mesmo antes de essas alterações serem escritas no disco. [11]

Um procedimento chamado CHECKPOINT escreve os dados da cache para o disco, e marca todos os dados que antes estavam como “sujos”, como “não-sujos”, ou que não há diferença entre o disco e a cache. O CHECKPOINT pode ocorrer em dois momentos, [11]:

- Quando o tamanho da cache for excedido, ou seja, a cache está cheia;
- Quando um limite de tempo para CHECKPOINT é atingido. Nesse caso a cache pode não estar cheia, mas o CHECKPOINT é feito mesmo assim.

É importante observar que, caso o sistema gerenciador de banco de dados sofra uma falha (uma queda na energia, por exemplo), se houver dados "sujos" na cache, ou

seja, dados que foram alterados desde o último CHECKPOINT, todos esses dados são perdidos porque ainda não foram escritos no disco. Uma forma de resolver isso seria o uso de arquivos WAL, como veremos adiante.

3.1.5. Arquivos WAL

O registro prévio da escrita (WAL = Write Ahead Logging) é uma abordagem padrão para registrar transações. Em poucas palavras, o conceito central do WAL é que as alterações nos arquivos de dados (onde as tabelas e os índices residem) devem ser escritas somente após estas alterações terem sido registradas, ou seja, quando os registros que descrevem as alterações tiverem sido descarregados em um meio de armazenamento permanente. Se este procedimento for seguido, não será necessário descarregar as páginas de dados da cache para o disco a cada efetivação de transação (também chamado de COMMIT), porque se sabe que no evento de uma queda será possível recuperar o banco de dados utilizando os arquivos WAL: todas as alterações que não foram aplicadas às páginas de dados no disco são refeitas a partir dos arquivos WAL, [20].

3.2. Tecnologias Aplicadas

Nessa seção serão levantadas e descritas as tecnologias utilizadas na implementação da ferramenta FileDB.

3.2.1 SQLite

SQLite é um SGBD embarcado contido em uma única biblioteca escrita na linguagem de programação C, que é carregada pela aplicação. Cada banco de dados está contido em um arquivo local ou em memória, e mesmo assim consegue ser um SGBD compatível com o conceito ACID. SQLite é o SGBD mais usado do mundo. [18]

SQLite suporta a maior parte do ANSI SQL e é multi-plataforma. Apesar de não ser um SGBD de alto desempenho, se mostrou ideal para servir como um formato interno de uma aplicação ou como base para um sistema de cache, por ser embarcado, com utilização implícita e independente de configuração do usuário.

Apesar da ênfase em planilhas com grande volume de dados, foram feitos testes com arquivos de mais de 4 milhões de linhas, com um tempo de retorno de 15 minutos. Comparando-se com planilhas que travavam (o programa gerenciador de planilhas eletrônicas deixava de responder durante muito mais tempo), o tempo de resposta foi tomado como satisfatório levando em consideração as facilidades de utilização do SQLite.

3.2.2 Python

Python é uma linguagem de programação interpretada de propósito geral. Python enfatiza legibilidade de código em pequenos e grandes projetos. Provê um sistema de tipagem dinâmica e gerenciamento automático de memória. Suporta múltiplos paradigmas de programação, incluindo programação procedural, orientada a

objetos, imperativa e funcional. Python também tem uma ampla biblioteca interna padrão. [21]

Interpretadores Python estão disponíveis para a maioria dos sistemas operacionais. CPython, a implementação de referência do interpretador Python, é software livre e segue um modelo de desenvolvimento baseado na comunidade.

A linguagem Python foi utilizada por ser uma linguagem atual, de fácil entendimento e bem compreendida pelo programador da ferramenta, além de possuir bibliotecas prontas para manipular arquivos XLSX, um dos pontos principais da aplicação.

3.3. Especificações da Aplicação

FileDB é um sistema que permite o pré-carregamento de arquivos XLSX para análise utilizando o poder da linguagem SQL, sem requerer que esses arquivos sejam carregados para um sistema de bancos de dados de forma explícita ou manualmente.

Oferece uma interface de linha de comandos completa, confortável e cheia de recursos para que o usuário possa manipular os arquivos e escrever comandos SQL com facilidade.

Mantém os dados em um banco de dados SQLite utilizando-o com uma cache que poderá ser em memória ou disco, e com os comandos **cache**, **uncache** e **checkpoint**, o usuário pode carregar e descarregar dados entre arquivos e a cache, e também escrever em disco as alterações feitas somente na cache.

FileDB é uma ferramenta muito útil para gerenciar arquivos XLSX muito grandes, que muitas vezes nem podem ser abertos por editores de planilhas por ter muitas linhas. O usuário é convidado a enxergar cada arquivo XLSX como se fosse uma tabela. Então o usuário pode executar consultas **select** completas (incluindo **JOIN** ou

UNION entre múltiplos arquivos), **insert**, **update** e **delete** sobre os dados. O usuário também pode criar e remover arquivos XLSX usando os comandos **create** e **drop**.

Como os dados são carregados em uma cache construída em SQLite, todas a propriedades que o SQLite provê são garantidas na ferramenta.

Além das garantias oferecidas pelo SQLite, também foi implementado um conceito de banco de dados de recuperação de falhas, funciona de tal forma que entre um checkpoint e outro, são armazenados todos os comandos insert, update e delete em arquivos WAL, algo que sugere o funcionamento de um SGBD tradicional. Com o comando **crash**, o usuário pode simular um encerramento incorreto ou falha do sistema. Em seguida, com o comando **recover**, o sistema utiliza a informação armazenada nos arquivos WAL para recuperar todas as alterações realizadas desde o último checkpoint até o momento do **crash**.

Comandos como **cache**, **uncache**, **checkpoint**, **crash** e **recover** são comandos internos da aplicação, executados implicitamente em determinadas situações, mas que são disponibilizados ao usuário, caso deseje executá-los explicitamente pelo prompt de comandos.

3.3.1 Fluxo de Execução da Aplicação

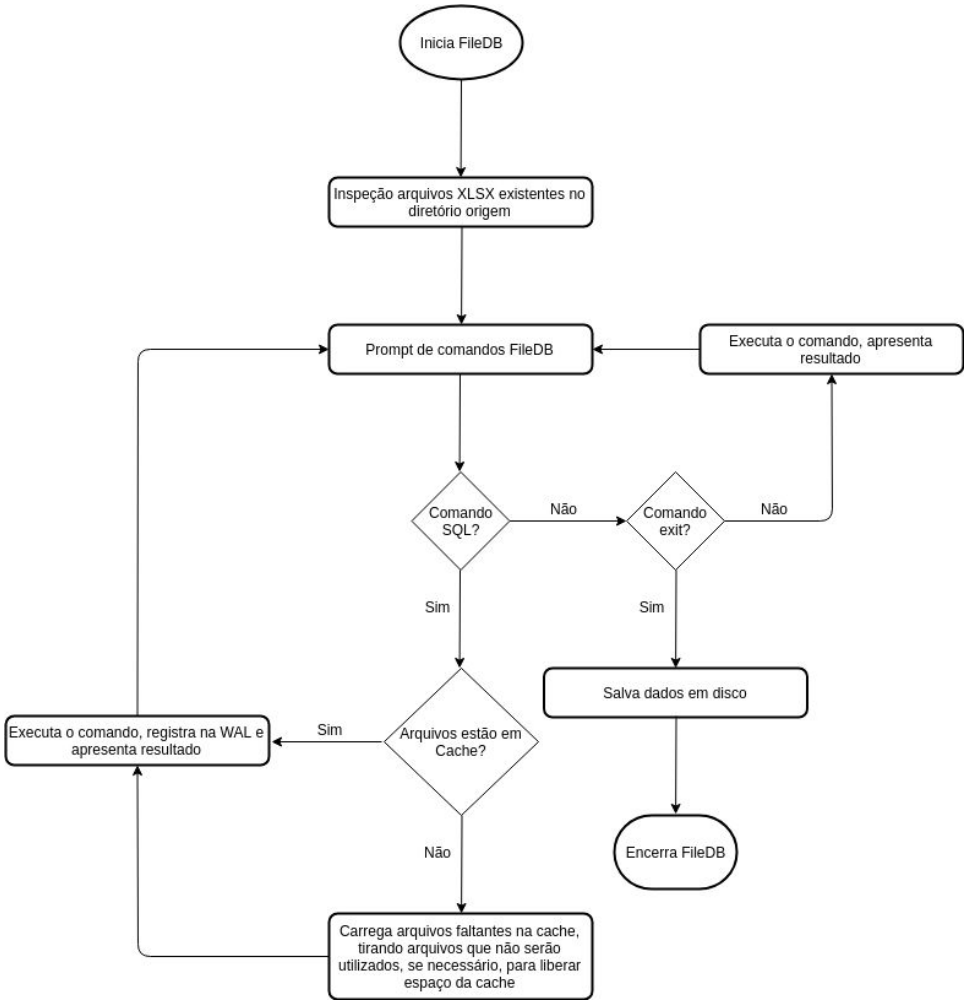

Figura 3.1: Fluxo de Execução FileDB.

No início da execução a aplicação varrerá o diretório de entrada à procura de todos os arquivos XLSX presentes. A cada arquivo encontrado, ele cria uma entrada no seu banco interno de controle, que dita, entre outros dados, se o arquivo está carregado em cache, o tamanho do arquivo, dados de último acesso, última escrita, última leitura, se ele possui modificações ainda não salvas em disco, etc.

Após essa listagem, o usuário recebe o prompt de comando d, no qual pode executar comandos Python, comandos Linux Shell, comandos SQL (para manipular os arquivos XLSX) ou comandos do próprio FileDB.

Quando o usuário executa algum comando SQL, a aplicação entende que ele gostaria de fazer algum tipo de leitura ou manipulação dos arquivos XSLX, e faz um "parsing" (interpretação) desses comandos para identificar sua categoria (DML, DCL, DQL, DDL), identificando portanto se são comandos de leitura, de escrita, de leitura e escrita, etc, além de identificar os arquivos envolvidos na operação.

Encontrados os arquivos necessários, ele verifica se estes arquivos estão em cache (um banco SQLite em memória ou em disco, dependendo da configuração do programa). Se os arquivos estiverem em cache, eles podem ser manipulados pelo comando SQL. Caso um ou mais arquivos não estejam em cache, a aplicação faz o carregamento desses dados para a cache. Esse processo consiste em abrir o arquivo XLSX através de uma biblioteca do Python, ler o arquivo em blocos, tratar os dados de cada bloco e inserir na cache, para que eles possam ser manipulados através de SQL. Cada arquivo encontrado será uma tabela no banco SQLite da cache.

Depois de carregados na cache, cada comando será aplicado sobre a cache para manipular os dados e então será armazenado nos arquivos de controle WAL. O resultado é então apresentado ao usuário, e o prompt é devolvido a ele.

Caso seja executado um comando Python, Linux Shell ou do FileDB, o aplicativo executa, devolve o resultado e retorna ao prompt.

Quando o usuário decidir usar o comando exit no prompt, o aplicativo irá salvar todas as alterações que ainda estiverem em cache no disco, e encerrar a execução.

O fluxograma acima descreve a interação explícita entre a aplicação e o usuário. Também há outros processamentos que ocorrem de forma implícita. Por exemplo, de tempos em tempos é disparado um checkpoint, que salva as alterações que estão em cache para o disco, da mesma forma que é feito ao sair da aplicação. Em outras palavras, um checkpoint realiza uma limpeza da cache e salva os dados em armazenamento persistente, que no caso são os arquivos XLSX. Essa escrita também

é feita em blocos. São lidos blocos de dados da cache, tratados e escritos nos arquivos XLSX correspondentes. O objetivo do checkpoint é persistir os dados e ir liberando a cache para novos comandos SQL, que usem outros arquivos.

Outro ponto citado no fluxograma é a escrita dos comandos SQL em arquivos WAL. Esses arquivos são utilizados em caso de recuperação de catástrofes. Eles logam todas as transações SQL executadas desde o último checkpoint, ou seja, tem todos os comandos que ainda não foram refletidos da cache para os arquivos XLSX. Caso o programa seja interrompido por alguma exceção ou catástrofe, ele consegue recuperar o estado anterior, mesmo não estando salvo nos arquivos XLSX ainda.

3.3.1 Esquemas ETL de leitura e escrita de arquivos XLSX

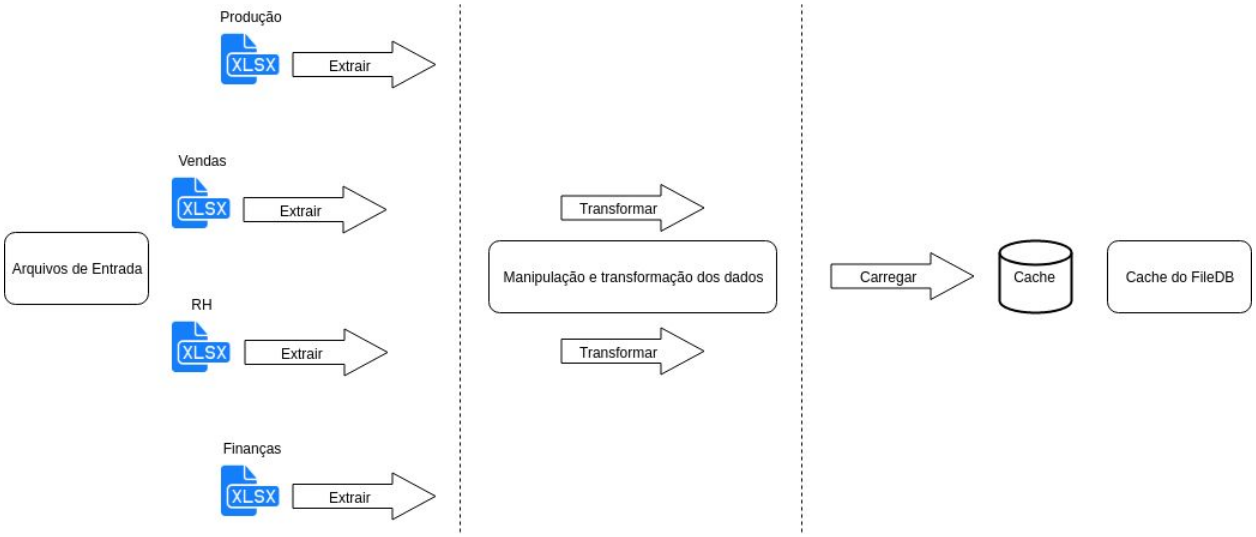

Figura 3.2: ETL de leitura FileDB.

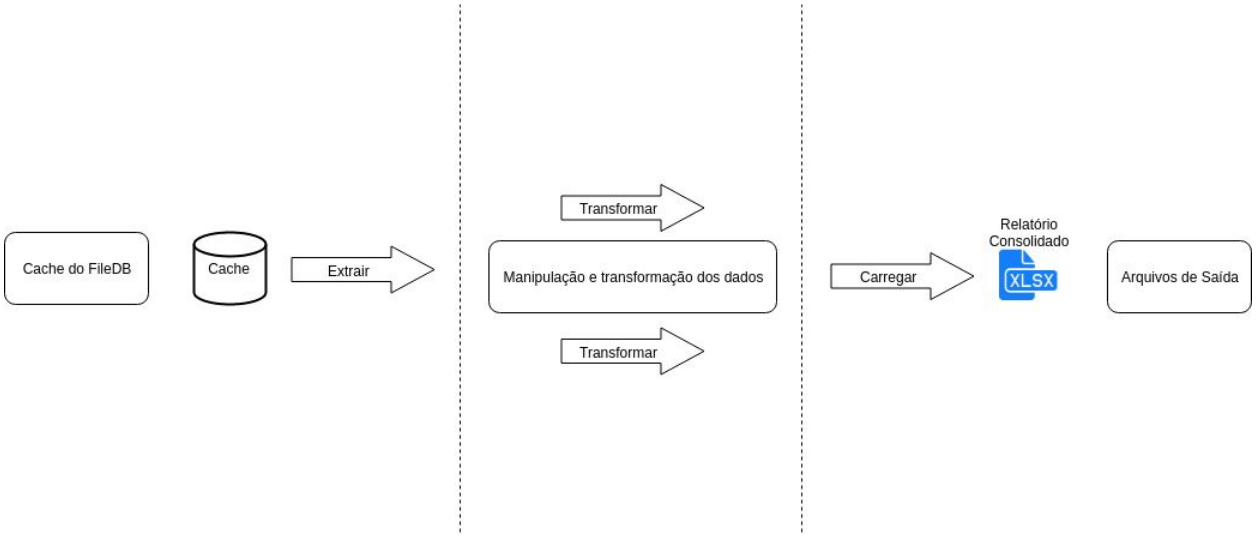

Figura .32: ETL de escrita FileDB.

Como já mencionado, a aplicação utiliza uma biblioteca Python que contém uma API para fazer leitura de arquivos XLSX.

Conforme figura 1 esta biblioteca é utilizada na etapa de extração efetuando a leitura dos dados em bloco para otimizar o uso de memória.

Este dados são tratados no Python e conforme o tipo dos dados na tabela correspondente criada na cache para recepcionar estes blocos de dados. Correspondendo ao processo de transformação e carregamento dos dados das planilhas XLSX para a cache (SQLite) .

A figura 2 mostra o processo análogo para extração dos dados da cache (SQLite) para escrita nos arquivos XLSX.

As figuras acima demonstram a relação de funcionamento da ferramenta com o conceito de ETL. Foi utilizado um cenário de exemplo no qual haviam 4 arquivos no diretório fonte, sendo estes trabalhados para gerar o relatório final apresentado na figura 2 e estes foram removidos após a utilização.

3.3.3. Caso de uso

Para o período de setembro a dezembro de 2016, cliente enviou para a Planning Service a movimentação de produção através dos arquivos ilustrados na figura 3.4:

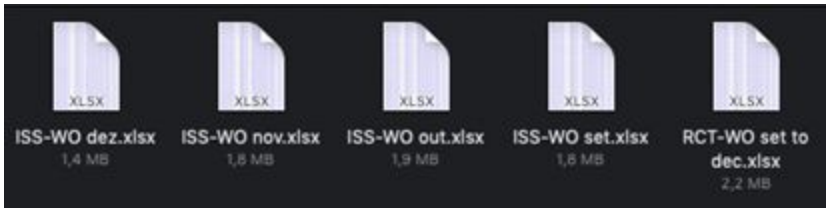

Figura 3.4: arquivos de produção.

Cada arquivo tem uma planilha com dados parecidos com o que está na figura 3.5 abaixo:

	A	B	C	D	E	F	G	H	I	J
1	Trans	Local	Numero de Item	Descricao	Descricao	Data	Data Efetiva	hora	TT	Descricao
2	3991142	2000	GR#E782-180	Graxa NKG814	NKG814 Grease	30/12/2016	30/12/2016	10:34:06	ISS-WO	Consumo Produção ou Troc
3	3991141	2000	ETJ79K002*R806	CONJUNTO TRIPECA	TRIPOD KIT	30/12/2016	30/12/2016	10:34:06	ISS-WO	Consumo Produção ou Troc
4	3991140	2000	ETJ79K002*R806	CONJUNTO TRIPECA	TRIPOD KIT	30/12/2016	30/12/2016	10:34:06	ISS-WO	Consumo Produção ou Troc
5	3991139	2000	ETJ79K002*R806	CONJUNTO TRIPECA	TRIPOD KIT	30/12/2016	30/12/2016	10:34:06	ISS-WO	Consumo Produção ou Troc
6	3991138	2000	ETJ79K002*R806	CONJUNTO TRIPECA	TRIPOD KIT	30/12/2016	30/12/2016	10:34:06	ISS-WO	Consumo Produção ou Troc
7	3991137	2000	EBJ82MK40-45*R803	GAIOLA & PISTA INTERNA	CASSETE	30/12/2016	30/12/2016	10:34:06	ISS-WO	Consumo Produção ou Troc
8	3991136	2000	EBJ82MK40-45*R803	GAIOLA & PISTA INTERNA	CASSETE	30/12/2016	30/12/2016	10:34:06	ISS-WO	Consumo Produção ou Troc
9	3991135	2000	EBJ82MK40-45*R803	GAIOLA & PISTA INTERNA	CASSETE	30/12/2016	30/12/2016	10:34:06	ISS-WO	Consumo Produção ou Troc
10	3991134	2000	8-482#ETJ79-B608	Tulipa Lavada	Whashed tulip	30/12/2016	30/12/2016	10:34:06	ISS-WO	Consumo Produção ou Troc
11	3991133	2000	8-482#ETJ79-B608	Tulipa Lavada	Whashed tulip	30/12/2016	30/12/2016	10:34:06	ISS-WO	Consumo Produção ou Troc
12	3991132	2000	8-482#ETJ79-B608	Tulipa Lavada	Whashed tulip	30/12/2016	30/12/2016	10:34:06	ISS-WO	Consumo Produção ou Troc
13	3991131	2000	3-734#EBJ82M-B608	Bell Lavado	Washed Bell	30/12/2016	30/12/2016	10:34:06	ISS-WO	Consumo Produção ou Troc
14	3991130	2000	3-734#EBJ82M-B608	Bell Lavado	Washed Bell	30/12/2016	30/12/2016	10:34:06	ISS-WO	Consumo Produção ou Troc
15	3991129	2000	29-3512#BJ82-B507	Eixo Lavado	Whashed Shaft	30/12/2016	30/12/2016	10:34:06	ISS-WO	Consumo Produção ou Troc
16	3991128	2000	29-3512#BJ82-B507	Eixo Lavado	Whashed Shaft	30/12/2016	30/12/2016	10:34:06	ISS-WO	Consumo Produção ou Troc
17	3991127	2000	20-61BJ79	Abracadeira Grande IB	IB Large Clamp	30/12/2016	30/12/2016	10:34:06	ISS-WO	Consumo Produção ou Troc
18	3991126	2000	20-40#BJ79	Abracadeira Peq. IB	IB Small Clamp	30/12/2016	30/12/2016	10:34:06	ISS-WO	Consumo Produção ou Troc
19	3991125	2000	20-144#BJ75L	Abracadeira G. Bell	OB large Clamp	30/12/2016	30/12/2016	10:34:06	ISS-WO	Consumo Produção ou Troc

Figura 3.5: arquivo de produção (layout).

Os conjuntos de dados são classificados da seguinte forma:

- ISS-WO: Saída por Consumo na Produção
- RCT-WO: Entrada por Consumo na Produção

Cada planilha possui em torno de 20 mil linhas. Dessa massa de dados precisamos considerar apenas os registros em que o campo “Natureza” contém os seguintes valores:

- xxrebkfld.p

- poporc.p
- zzpoporc.p
- rebkfl.p

O sistema de cálculo contábil da Planning Service requer que os dados de produção sejam organizados em um formato específico para carregamento. O “layout” de entrada de dados de produção requer que os dados estejam num determinado formato e requer colunas específicas.

Após análise dos dados enviados pelo cliente, o mapeamento dos campos pode ser feito da seguinte forma:

Campo de Entrada do Sistema	**Campo da Planilha do Cliente**	**Planilha do Cliente**
Número do Documento	ID	RCT-WO
Data de Entrada do Item Produzido no Estoque	Data Efetiva	RCT-WO
Código do Item Produzido	Número de Item	RCT-WO
Quantidade Produzida	Alterar Qde Lugar	RCT-WO
Código do Item Consumido	Número de Item	ISS-WO
Quantidade Consumida	Alterar Qde Lugar	ISS-WO

- **Objetivo**: aplicar uma consulta SQL sobre os arquivos originais do cliente para gerar uma planilha única que possa ser utilizada para carregamento dos dados de produção correspondentes ao período de setembro a dezembro de 2016, diretamente no sistema de cálculo contábil.
- **Desafios**:
    - Unificar as planilhas de consumo ISS-WO de setembro a dezembro em uma única planilha de consumo.

      Isso pode ser feito utilizando a cláusula UNION da linguagem SQL.

- Vincular as planilhas de consumo ISS-WO unificadas com a planilha de produção RCT-WO, através da coluna ID. O relacionamento entre as planilhas RCT-WO e ISS-WO é de 1 pra N. Isso pode ser feito utilizando a cláusula INNER JOIN da linguagem SQL.
- Projetar apenas as colunas necessárias para a planilha de entrada de dados do sistema contábil. Isso pode ser feito utilizando a cláusula SELECT (projeção) da linguagem SQL.

3.3.3.1. Utilizando o FileDB

Em um terminal de comando do Linux, entrar na pasta onde estão os arquivos do cliente e executar o comando **filedb** para abrir o prompt de comando da Aplicação.

Automaticamente será escaneado a pasta atual em busca de arquivos com extensão XLSX. Em seguida é possível utilizar o comando **tables** para verificar a lista atual de tabelas:

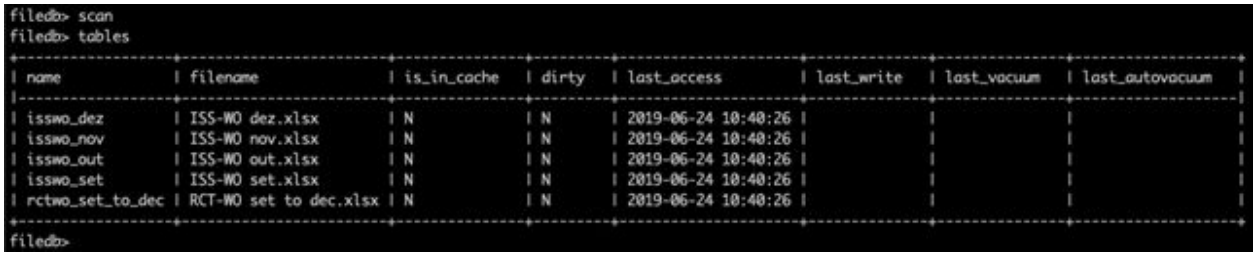

Figura 3.6: Prompt de comando da Aplicação.

No prompt de comando, é possível criar uma planilha de produção com o seguinte comando SQL:

```
CREATE TABLE producao_set_a_dez AS
SELECT p.id              AS numero_documento,
```

```
        p.data_efetiva    AS data_entrada,
        p.numero_de_item  AS item_produzido,
        p.alterar_qde_lugar AS quantidade_produzida,
        c.numero_de_item  AS item_consumido,
        c.alterar_qde_lugar AS quantidade_consumida
FROM rctwo_set_to_dec p
INNER JOIN (
  SELECT * FROM isswo_set
  UNION
  SELECT * FROM isswo_out
  UNION
  SELECT * FROM isswo_nov
  UNION
  SELECT * FROM isswo_dez
) c
ON c.id = p.id;
```

Assim que a consulta terminar, o arquivo producao_set_a_dez.xlsx estará criado na mesma pasta e pronto para ser carregado para o sistema de cálculo contábil da Planning Service.

É importante também notar que utilizando o aplicativo, muitas outras análises podem ser feitas rapidamente utilizando o poder da linguagem SQL, por exemplo:

- Produção sem Consumo: Registros que estão na planilha RCT-WO mas não estão nas planilhas ISS-WO;
- Consumo sem Produção: Registros que estão nas planilhas ISS-WO mas não estão na planilha RCT-WO;
- Item Produzido em Maior Quantidade;
- Item Consumido em Maior Quantidade;
- Entre outras análises.

4. Conclusão

O presente trabalho de graduação teve como objetivo o desenvolvimento de um aplicativo a fim de atender as necessidades da empresa Planning Service, para a melhoria da manipulação dos dados fornecidos pelos seus clientes.

A empresa nutria uma grande insatisfação quanto à morosidade para o tratamento destas informações manualmente e dependendo de aplicações lentas, que por mais que se fossem utilizados os computadores mais robustos, não observava-se melhora no desempenho do editor de planilhas.

Quanto à relevância do tema, foi possível comprovar que essa necessidade existia por parte da empresa. Isso foi positivo para o trabalho, pois dessa forma o projeto pôde ser desenvolvido mediante a necessidade que o estabelecimento apresentou.

Para a criação deste aplicativo, foram utilizados os conceitos aprendidos durante o curso de Ciência da Computação, bem como informações fornecidas pela empresa.

O desenvolvimento do trabalho se deu a partir da utilização da linguagem Python e banco de dados SQLite.

A partir disso, pôde-se obter o resultado esperado em relação ao que foi inicialmente proposto, através de scripts desenvolvidos para manipulação dos arquivos disponibilizados pelos clientes. Estes scripts foram executados pela aplicação e a preparação das informações para o layout de importação do sistema da Planning Service ocorreu de forma automática, ou seja, uma vez investido tempo de desenvolvimento para implementar o script para manipulação de determinado arquivo, toda nova geração de informações ocorreu de forma transparente e sem intervenções manuais.

Ao final, o que se esperou realizar e concretizar com a elaboração do trabalho foi a possibilidade da melhoria da qualidade e da agilidade durante o processo de manipulação de dados através do FileDB.

Espera-se que esse trabalho possa contribuir com o desenvolvimento da Planning Service, bem como com a comunidade acadêmica.

Referências Bibliográficas

[1] Planning Service Transfer Pricing LTDA. https://www.planningservice.com.br, 2019.

[2] O que é ERP?, Totvs. https://www.totvs.com/blog/o-que-e-erp/, 2019.

[3] PgLoader. https://pgloader.io/, 2019.

[4] Rows. http://turicas.info/rows/, 2019.

[5] QueryCSV. https://pythonhosted.org/querycsv/, 2019.

[6] Q. http://harelba.github.io/q/, 2019.

[7] TextQL. https://github.com/dinedal/textql, 2019.

[8] FDW. https://www.postgresql.org/docs/current/file-fdw.html , 2019.

[9] MS-XLSX. Excel (.xlsx) Extensions to the Office Open XML SpreadsheetML File Format. https://docs.microsoft.com/en-us/openspecs/office_standards/ms-xlsx/2c5dee00-eff2-4b22-92b6-0738acd4475e , 2019.

[10] Preeti Dhanda, Neetu Sharma. Extract Transform Load Data with ETL Tools, 2016.

[11] Raghu Ramakrishnan, Johannes Gehrke. Sistemas de Gerenciamento de Banco de Dados. Ed. McGraw-Hill Interamericana do Brasil, 2008.

[12] PostgreSQL. https://www.postgresql.org/, 2019.

[13] Oracle. https://www.oracle.com/index.html, 2019.

[14] MySQL. https://www.mysql.com/, 2019.

[15] MariaDB. https://mariadb.org/, 2019.

[16] Microsoft SQL Server. https://www.microsoft.com/pt-br/sql-server/sql-server-2019, 2019.

[17] Firebird, https://firebirdsql.org/, 2019.

[18] SQLite. https://www.sqlite.org/index.html 2019.

[19] IBM DB2. https://www.ibm.com/br-pt/products/db2-database, 2019.

[20] WAL. http://pgdocptbr.sourceforge.net/pg80/wal.html, 2019.

[21] Python. https://www.python.org/, 2019.

[22] Definição da linguagem SQL. ANSI/ISO/IEC International Standard (IS). Database Language SQL — Part 2: Foundation (SQL/Foundation), 1999.

[23] ACID: Concurrency Control of Open Transactions. https://mariadb.com/kb/en/library/acid-concurrency-control-with-transactions/ The MariaDB Project Documentation, 2019.