5
VelvetH-DB

Os Sistemas Gerenciadores de Banco de Dados (SGBDs), foram criados com o intuito de permitir o armazenamento e manipulação de grandes volumes de dados, fornecendo uma aplicação que garanta, de forma transparente para o usuário, um gerenciamento de memória eficiente além da garantia das propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade), em suas transações.

Como parte do objetivo desta dissertação, propomos a implementação do VelvetH-DB, que é uma abordagem baseada em banco de dados para a geração dos Roadmaps, que tem como objetivo criar uma solução robusta e eficaz da fase de geração de anotações do VelvetH.

Quando o volume de dados a ser processado pelo VelvetH não cabe em memória principal, o sistema operacional passa a fazer o uso do SWAP em disco, tornando a aplicação drasticamente mais lenta[9], e em caso de falha, todo o processamento realizado, que podem ser dias de trabalho [9] são perdidos e o processo não pode ser reiniciado a partir do ponto de interrupção. No VelvetH-DB, conseguiremos a robustez desejada através da propriedade de Consistência, garantindo desta forma a continuidade no processamento mesmo em casos onde há falha na máquina.

Outra funcionalidade alcançada com a implementação proposta está ligada à propriedade de Durabilidade. Quando surgem novas bibliotecas de sequências para serem processadas, todo o trabalho de geração do *Roadmap* tem que ser feito novamente, não sendo possível processar apenas o conjunto novo de dados e incluir ao final do processamento já realizado. Na abordagem em banco de dados, iremos encurtar o tempo de processamento de novas bibliotecas evitando desperdício de tempo de processamento de análises já realizadas, procedendo de maneira incremental.

5.1.
Fluxo para geração do *Roadmap*

O fluxo para a geração do *Roadmap* através da implementação do VelvetH original, como descrito na figura 20, é relativamente simples. O usuário, através de linha de comando, executa a aplicação com seus parâmetros e a lista dos arquivos contendo as sequências a serem processadas. O programa executa as etapas, já descritas na secção 4.4, salvando em disco os arquivos de saída, sequences e *Roadmap*.

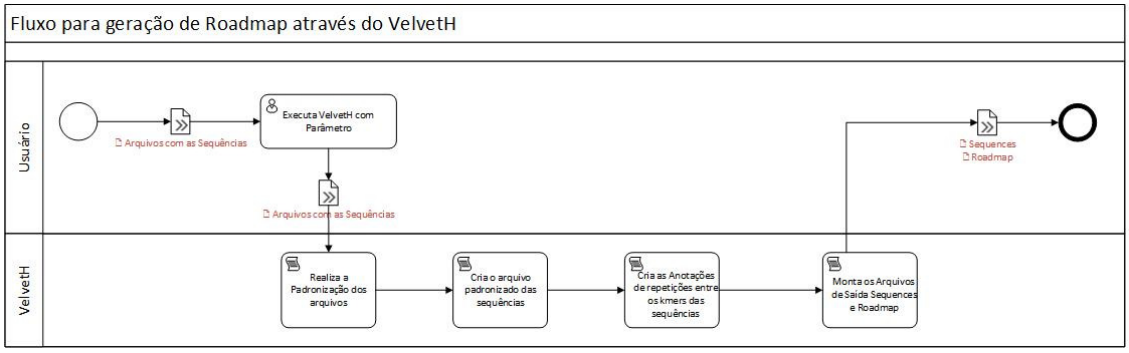

Figura 20 Fluxo de geração do *Roadmap* através do VelvetH sem modificações.

A solução proposta nesta dissertação, o VelvetH-DB, utiliza para fase inicial de padronização das sequências uma versão alterada do VelvetH, usando a saída deste como entrada para a nova implementação com persistência em banco de dados, conforme demostrado na Figura 21.

É pensado como contribuição futura a implementação desta fase de padronização dos dados existente no VelvetH na solução VelvetH-DB.

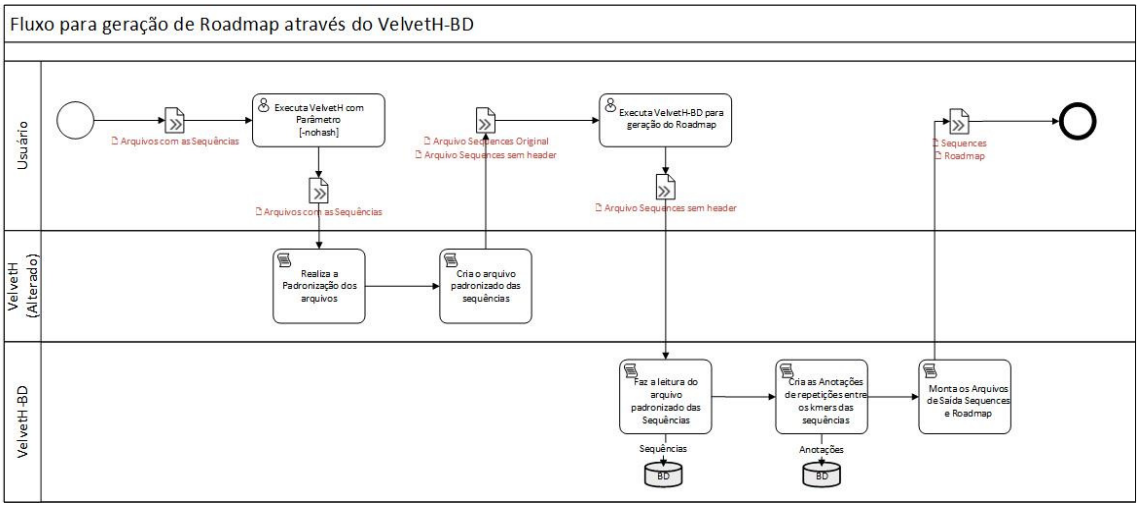

Figura 21 Fluxo para a geração do *Roadmap* através do VelvetH-BD.

5.2. Implementação

Para a implementação deste trabalho foi escolhido o SGBD de código aberto *PostgreSQL*, versão 9.3. Foi escolhido este SGBD pois, além de ser um software estabilizado e difundido, há no grupo de pesquisas BioBD PUC-Rio, experiência em alterações no código deste SGBD.

A implementação aqui proposta, consiste em uma alteração no VelvetH original para gerar um outro formato para o arquivo padronizado de sequências e um conjunto de funções de banco de dados, escritas em *PL/pgSQL*, que é a linguagem estruturada do próprio *PostgreSQL*.

As funções criadas cobrem a fase 2 do Velvet, processando o arquivo padronizado com as sequências, dividindo cada sequência em k-mers e gerando as anotações das repetições entre as sequências já processadas. Ao final escreve-se um arquivo Roadmap que possuiu a mesma estrutura do Roadmap gerado pela aplicação original e pode ser utilizado pelo VelvetG, para a geração do grafo e completitude da montagem.

5.2.1. Alteração do VelvetH

Foi realizada uma alteração no código fonte do VelvetH com o intuito de gerar um outro arquivo de saída, sem o cabeçalho das sequências, colocando todas as informações em uma única linha, conforme mostrado na Figura 22 a seguir, visando

facilitar o processo de importação das sequências e suas informações na fase de carregamento das sequencias na solução proposta.

```
>HWUSI-EAS-100R_0001:7:1:1:701#TGACCA/1 0   0   TCGTACCGTAAGGAACGGTGGACTGGNTACGAGTGAGAATGTTGGCATCAGTAGCGCGATGTGGGTGAGAATCCCCCAG
>HWUSI-EAS-100R_0001:7:1:1:1658#TGACCA/1    0   0   TTGGGATATCGCCAAGCGGTAAGGCAACGGACTTTGACTCCGTCATTCGCAGGTTCGAATCCTGCTATCCCAGCCAAAA
>HWUSI-EAS-100R_0001:7:1:2:928#TGACCA/1 0   0   CCCGAAGGGATTTCCTCGATTAAGAGTAATGCAAGGGATGTCAAGTGTAGGTAAGGTTCTTCGCGTTGCTTCGAATTAA
>HWUSI-EAS-100R_0001:7:1:2:758#TGACCA/1 0   0   TATTTTATTTACAATACTAAGTACTATAGCTTGAAATTGATCTAAAAGCTTTTCTACTAAATCAGCCTTTGTCCATCCA
>HWUSI-EAS-100R_0001:7:1:2:606#TGACCA/1 0   0   TCAAATTCTCTTAAAGTTCCCAAATTCCATCCTTTCCTATTAAGATTTTATATCTTAAATTAGATCGGAAGAGCACAC
>HWUSI-EAS-100R_0001:7:1:2:1023#TGACCA/1    0   0   TGTCCACAACCCCGGAAACAAGTTTCCGGTTTGGGCTCTTTCCCTTTCGCTCGCCGCTACTACGAAACTCGATTTTTCT
>HWUSI-EAS-100R_0001:7:1:2:1502#TGACCA/1    0   0   CTTTTTAGCAAGACAGGGAAGATCCTTAAATGTTGGATGTATATTTAACGGGCATTCAGTTCTTGACATACCAACTGAA
>HWUSI-EAS-100R_0001:7:1:2:1888#TGACCA/1    0   0   TCTCCTATTCTAAACCCTGGGTCTAAGTCTATTAAGTGGAAATTATTTTTTATAACTTCTAAACAAAAAGAATGCATAG
>HWUSI-EAS-100R_0001:7:1:3:36#TGACCA/1  0   0   CCCTATTCAGACTCGGTTTCCCTTCGGCTCCGTACCTTAAGTACTTAACCTTGCTACATACCCTAACTCGCTGGCTCGT
>HWUSI-EAS-100R_0001:7:1:3:1538#TGACCA/1    0   0   GATTTAGAATCAAACTATAAAGTTTTAGAAGATAAAGTAACAACTTATACATGGGGATTCGGAGTTAAAAAAGTAGATT
>HWUSI-EAS-100R_0001:7:1:3:1095#TGACCA/1    0   0   CCTGAAATTACTGATGGAAATACTAATAAAGTTACACTTACCTATACTAATAATCCAGATTTAGAATCAAACTATAAAG
>HWUSI-EAS-100R_0001:7:1:3:124#TGACCA/1 0   0   AATCTATATAAAAAGAACTTTAGTGCCATAGGATCACAAATACTAGCAGTAGCCATAATATATGCAACCACAGCTTTTC
>HWUSI-EAS-100R_0001:7:1:3:1053#TGACCA/1    0   0   AGCATAGAAGGTAAACAAACTACTCCTAGGTATATTAGGAATAATAATATAGATATAGATTCTAATAGTAACTCAAAAG
>HWUSI-EAS-100R_0001:7:1:3:1821#TGACCA/1    0   0   ACTATTATTAGCTATAACTTCTTCAAGACTTGAAAAATCTGATTTTTTAACTAATCTTTCTATTGCTTTTTCTATAACA
>HWUSI-EAS-100R_0001:7:1:3:1328#TGACCA/1    0   0   CCGGAGCGAAGCAATCCTATGAAGAAACAGGAATATTTCCATCAATAACCCTGGCACAAGCTATATTAGAAAGTGGTTG
>HWUSI-EAS-100R_0001:7:1:3:429#TGACCA/1 0   0   TTGCCCCTAAAGTACATTCATCTATTTTTTCACTTCTTAAGCTTTTCTCTTCACAAGTTTCCTTAGGTAACTTATAGCA
>HWUSI-EAS-100R_0001:7:1:3:1534#TGACCA/1    0   0   CGGTCCTCCACGAGGTTTTACCCTCGCTTCAACCTGGCCATGGATAGGTCACCCGGTTTCGGGTCTCCGGCCTGCAACT
>HWUSI-EAS-100R_0001:7:1:4:1564#TGACCA/1    0   0   GCATCTTCTATAGATTGCTTAAATTTCTCATATAATGGATGCTCCTTAACACTTTCTTCAAGTTCATTTAAAGCTTGTA
>HWUSI-EAS-100R_0001:7:1:4:1232#TGACCA/1    0   0   CTAGGGGGCATAGCGCTTACCGAACCTTATCAAACTCCGAATGCCACATACTTTGAGCTCGGCAGTCAGCCTACGAATG
>HWUSI-EAS-100R_0001:7:1:4:538#TGACCA/1 0   0   CTGTCTCACGACGTTCTGAACCCAGCTCGCGTGCCGCTTTAATGGGCGAACAGCCCAACCCTTGGGACCTACCTCAGCC
>HWUSI-EAS-100R_0001:7:1:4:614#TGACCA/1 0   0   TATGGTCATTACAAAATAGAAGAAGTTAAAACACTTTGGAATTACGTTATTAACAAAGAGCCTATTTTCTTTGATGTTA
>HWUSI-EAS-100R_0001:7:1:5:126#TGACCA/1 0   0   ATTGAGAATAAGGATAAAAGCAAAGCTGAAATTTTTGAAGAACAAAATAAAACTGAAAATCTAAAATCAAATATAAATG
>HWUSI-EAS-100R_0001:7:1:5:1561#TGACCA/1    0   0   TTAAAAGACCTAAAAGGGAAAAAAGAGTGGGAAAAAGAAATGATTGGTAGAAAAAGAAAAACATTGGCATTATGTCATG
```

Figura 22 Arquivo de saída do VelvetH alterado, colocando todas as informações da sequência em uma única linha

O código fonte desta alteração está disponível em http://www.inf.puc-rio.br/~msilva/labbio/velvet_1.2.10-Alterado/.

5.3. Modelagem em banco de dados relacional

O esquema relacional conta com 5 tabelas, conforme ilustra o esquema na Figura 23:

- **SEQUENCES**: Armazena as informações originais das sequências geradas pelo VelvetH modificado.
    - **ID**: Chave primária auto incremental da tabela, é o identificador único da sequência
    - **CAT**: Categoria da sequência, informação gerada pelo VelvetH durante a padronização
    - **READ**: Conteúdo da sequência, conjunto de caractere
    - **DUMMY**: Identificador da sequência gerada pelo VelvetH, armazenado para conferência, mas não utilizada pois a aplicação permite inserir novas bibliotecas
- **PRESEQUENCES**: Armazena as informações das sequências ajustadas, onde é substituido o caractere ‘N’ pelo caractere ‘A’, conforme realizado no VelvetH original.

- **ID**: Identificador único da sequência
- **READ**: Conteúdo da sequência com os caracteres ‘N’ trocados por ‘A’.

- **IDENTIFICATION:** Armazena as informações dos k-mers únicos identificados nas sequências
  - **K-MER**: Conteúdo do k-mer, conjunto de caractere
  - **IDINT**: Posição relativa do k-mer na sequência
  - **ID**: Identificador único da sequência
  - **POS**: Posição relativa do k-mer na sequência
- **ROADMAP**: Armazena as informações das anotações de sobreposições encontradas entre as sequências
  - **ID**: Identificador único da sequência que foi processada
  - **OVERLAPPED_READ**: Identificador único da sequência onde foi encontrado o k-mer repetido
  - **POS_IN_READ**: Posição relativa do k-mer na sequência processada
  - **POS_IN_OVERLAPPED_READ**: Posição relativa do k-mer na sequência onde foi encontrado o k-mer repetido
  - **NUMB_OF_CONTINUOS_KMER_OVERLAPPED**: Quantidade de k-mers sobrepostos em sequência processada.
  - **ANNOTATION_ORDER**: Ordem da anotação
- **TEMPFILE**: Armazena temporariamente a estrutura de dados para a geração dos arquivos
  - **LINE**: Conteúdo da linha a ser colocada no arquivo
  - **ID**: Número sequencial da linha, para ordenação na hora de exportar o arquivo para disco.

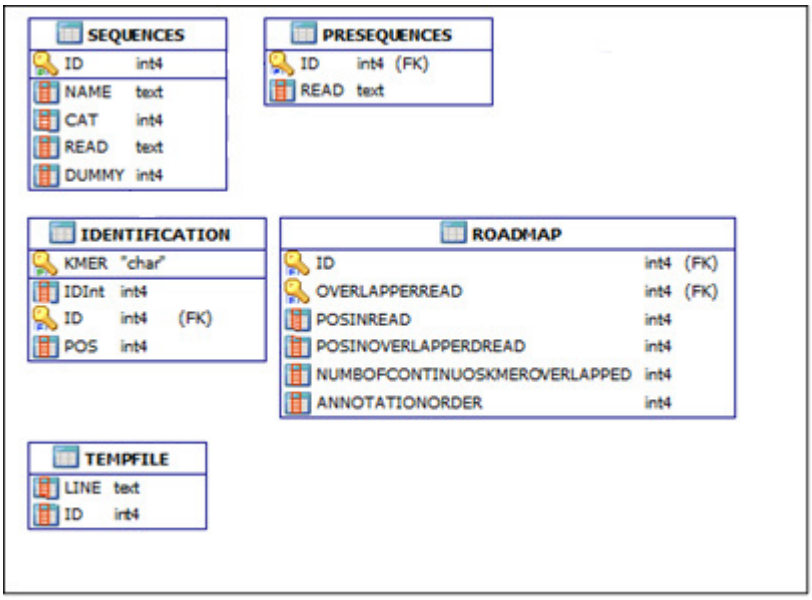

Figura 23 Esquema relacional do VelvetH-DB

5.4. Funções no SGBD

Para fazer o processamento das sequências e gerar o Roadmap, foram criadas 4 funções:

- Maestra
- Importa Arquivo
- Processa Roadmap
- Gera Output

Todas escritas em PL/pgSQL e na Figura 24 mostra-se o diagrama de sequência para a execução da aplicação.

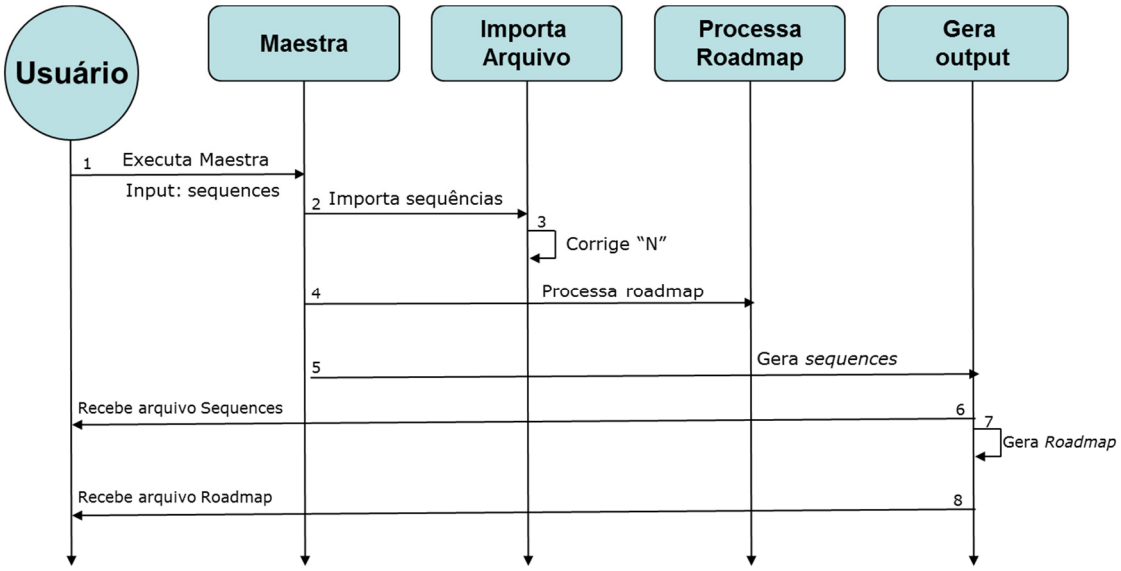

Figura 24 Diagrama de sequência das funções do VelvetH-BD

5.4.1. Maestra

É a função principal e possui a responsabilidade de fazer às chamadas as outras funções da aplicação. Tem como parâmetros de entrada o arquivo padronizado gerado pelo VelvetH alterado e o tamanho do k-mer, retorna os arquivos do *Roadmap* e *Sequences*.

5.4.2. Importa Arquivo

Função responsável pela leitura do arquivo padronizado das sequências, persistindo os dados na tabela SEQUENCES e gerando a tabela PRESEQUENCES com a substituição do caractere "N" das sequências pelo caractere "A". Esta alteração é realizada de forma arbitrária pelo Velvet, não foi encontrada explicação em documentações e artigos dos criadores.

Quando existe o caractere "N" em uma sequência. Isso significa que aquela base nitrogenada não pode ser lida no processo de sequenciamento. A aplicação assume como erro e altera para o caractere "A".

5.4.3. Processa Roadmap

Nesta função as sequências registradas na tabela PRESEQUENCES são recuperadas e os k-mers são extraídos e incluídos na tabela IDENTIFICATION quando o kmer já se encontrada na tabela, uma anotação é criada e escrita na tabela

ROADMAP registrando a repetição, lógica semelhante a estudada na secção 4.4.2.2.

5.4.4. GeraOutput

Ao final do processamento das sequências, esta função gera na tabela TEMPFILE as tuplas no formato do arquivo *sequences* e gera um arquivo texto em disco. Em seguinte, apaga os dados desta tabela e gera as linhas no formato do arquivo *Roadmap*, também exportando um arquivo texto em disco, finalizando assim o processo.

5.5.Testes de Execução

Afim de demonstrar a eficácia e eficiência da aplicação VelvetH-DB, realizamos um teste de execução com duas bibliotecas de dados, uma pequena e executável por uma máquina típica de prateleira, e um grupo de dados não executável neste tipo de máquina.

Foi criada uma máquina virtual, através do Virtual Box (V4.3.34), com uma instalação de sistema operacional Ubuntu 64 (V14.04 LTS) e 8GB de memória RAM disponíveis, além de 2 processadores (Intel® Core™ i7-4510U CPU @ 2.00GHz × 2) e 1TB de espaço em disco. Configurada para rodar sem SWAP ativo no sistema operacional, afim de medir o limite de uso da memória RAM.

O resultado da execução das duas aplicações, apresentado na Figura 25, expõe que a implementação proposta é menos eficiente que a aplicação tradicional quando o conteúdo cabe na memória principal da máquina. Entretanto, a aplicação VelvetH-DB se mostra eficaz nos casos onde a memória RAM da máquina não é suficiente para comportar toda massa de dados. No caso de teste executado o tempo para a montagem do arquivo sequences e Roadmap foram de 156h.

Número de Sequências	VelvetH			VelvetH-DB		
	Tempo de execução	Qtd. sequências processados	Execução sucedida	Tempo de execução	Qtd. sequências processados	Execução sucedida
25.000	<1s	25 K	Sim	10 m	25 K	Sim
10.151.440	N/A	7 M	Não	156h	10 M	Sim

Figura 25 Resultados da execução do VelvetH e o VelvetH-BD

5.6. Conclusão

Neste capítulo foram apresentadas as motivações de uma implementação baseada em banco de dados. Descrito o fluxo de funcionamento proposto por esta nova aplicação, o esquema de dados relacional e as funções criadas para fazer o mapeamento das sobreposições das sequências em um banco de dados relacional.

No próximo capítulo apresentaremos as conclusões e objetivos alcançados deste trabalho de dissertação de mestrado.