Trabalho Prático: Banco de Dados Chave-Valor *

Prof. Pedro Henrique Penna

Departamento de Ciência da Computação
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)

1. Contexto

Um banco de dados de chave-valor usa um dicionário para armazenar dados, como ilustrado na Figura 1. Nessa organização, dados são estruturados em registros contendo dois campos: uma chave que identifica unicamente aquele registro no banco de dados; e um valor que armazena o dado propriamente dito. Chaves e valores podem ter tipos quaisquer e registros não possuem nenhuma relação entre si. Portanto, programas possuem máxima flexibilidade para gerenciar dados armazenados. Tipicamente, as seguintes operações são fornecidas:

- Inserção: insere um novo objeto no banco de dados.
- Remoção: remove um objeto do banco de dados.
- Busca: recupera um objeto no banco de dados.
- Atualização: atualiza um determinado objeto no banco de dados.

Bancos de dados de chave-valor são altamente particionáveis e permitem maior escalabilidade horizontal, em contrapartida a banco de dados relacionais. Por esse motivo, banco de dados de chave-valor têm ganhado cada vez mais espaço em aplicações atuais, principalmente no contexto da *cloud*. Alguns exemplos e banco dedados de chave-valor amplamente utilizados são o Redis, DynamoDB, LevelDB and MemcachedDB.

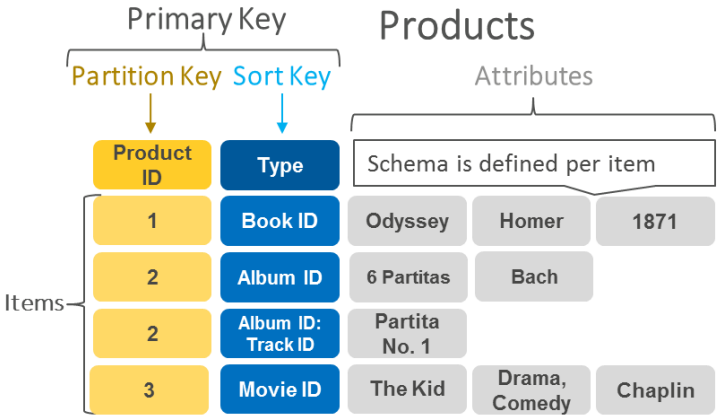

Figura 1: Arquitetura do DynomoDB.

* O presente enunciado está sujeito a correções e alterações.

2. Descrição

Nesse trabalho você deverá desenvolver (a) um banco de dados de chave-valor e (b) um programa cliente que possibilita a interação de um usuário final com o banco de dados. A seguir, os requisitos funcionais e não funcionais que devem ser implementados em cada um desses componentes são resumidos. Para mais detalhes consulte a Seção 3.

O banco de dados deve oferecer os seguintes requisitos funcionais:

- **RF-1**: Suportar a inserção de objetos no banco de dados.
- **RF-2**: Suportar a remoção objetos do banco de dados.
- **RF-3**: Suportar atualização objetos no banco de dados.
- **RF-4**: Suportar a pesquisa objetos no banco de dados.
- **RF-5**: Ser capaz de persistir os objetos do banco de dados em um arquivo.

O programa cliente deve oferecer os seguintes requisitos funcionais:

- **RF-6**: Suportar um comando de inserção de objetos no banco de dados.
- **RF-7**: Suportar um comando de remoção de objetos no banco de dados.
- **RF-8**: Suportar um comando de atualização objetos no banco de dados.
- **RF-9**: Suportar um comando de pesquisa objetos no banco de dados.

Além dos requisitos funcionais listados anteriormente, o seu banco de dados deve apresentar os seguintes requisitos não funcionais:

- **RFN-1** Suportar uma interface por linha de comandos para realizar operações diretas no banco de dados.
- **RFN-2**: Suportar a comunicação bidirecional com o programa cliente usando algum mecanismo de comunicação entre processos.
- **RFN-3** Suportar o processamento concorrente de requisições no banco de dados, usando threads.
- **RFN-4**: Manter em memória principal um número máximo de registros do banco de dados. Operações no banco de dados devem ocorrer necessariamente em objetos carregados na memória principal. Devem ser suportadas as seguintes estratégias de substituição de objetos: FIFO, Aging e LRU.

Quanto ao programa cliente do banco de dados, os seguintes requisitos não funcionais devem ser oferecidos:

- **RFN-7**: Ler as requisições do dispositivo de entrada padrão.
- **RFN-8**: Escrever a resposta recebida das requisições feitas ao banco de dados no dispositivo de saída padrão.

3. Especificações Técnicas

O programa banco de dados deve suportar a seguinte interface de linha de comando:

`simpledb [opcoes] [comando]`
Manipula registros do banco de dados de chave-valor SimpleDB. O argumento `opcoes` consiste em um lista opcional de argumentos para o banco de dados que permitem o controle de alguns de seus parâmetros. O argumento `comando` consiste em um comando opcional a ser executado no banco de dados. A seguir as opções e comandos suportados são listados.

Opções

`-cache-size=N,policy`
Inicia o banco de dados com suporte a uma cache de `N` objetos. Isto é, em qualquer momento durante sua execução, o banco de dados não mantém carregado em memória mais do que `N` objetos simultaneamente. Objetos devem ser substituídos usando a política especificada em `policy`, que pode ser `fifo`, `aging` ou `lru`.

Comandos

`--insert=<key,value>`
Insere um objeto no banco de dados. A chave de acesso (`key`) é um inteiro positivo. O objeto é codificado em uma cadeia de caracteres pela aplicação cliente. Ao concluir a operação, a chave do objeto inserido é impresso na saída padrão. Objetos são gravados no arquivo `simpledb.db`.

`--remove=<key>`
Remove do banco de dados o objeto que é identificado pela chave `key`. Objetos são removidos do arquivo `simpledb.db`.

`--search=<key>`
Busca no banco de dados objeto o que é identificado pela chave `key`. Caso o objeto seja encontrado sua seu valor é impressos na saída padrão. Objetos são buscados no arquivo `simple.db`.

`--update=<key,new-value>`
Atualiza o objeto que é identificado pela chave `key`. A chave de acesso (`key`) é um inteiro positivo. O objeto é codificado em uma cadeia de caracteres pela aplicação cliente. Objetos são buscados e alterados no arquivo `simple.db`.

O programa cliente de dados deve suportar a seguinte interface textual de comandos:

`simpledb-client`
Inicializa o programa cliente do banco de dados chave-valor SimpleDB. Esse programa lê comandos do dispositivo de entrada padrão, os envia para processamento no programa de banco de dados, e escreve na saída padrão a resposta obtida no processamento desses comandos. Os seguintes comandos são suportados:

Comandos `insert <key,value>`
Insere um objeto no banco de dados. A chave de acesso (`key`) é um inteiro positivo. O objeto é codificado em uma cadeia de caracteres pela aplicação cliente. Ao concluir a operação, o programa cliente imprime "inserted"no dispositivo de saída padrão.

`remove <key>`
Remove do banco de dados o objeto que é identificado pela chave `key`. Ao concluir a operação, o programa cliente imprime "removed"no dispositivo de saída padrão.

`search <key>`
Busca no banco de dados objeto o que é identificado pela chave `key`. Ao concluir a execução deste comando, o programa cliente imprime no dispositivo de saída padrão o valor associado à chave informada. Caso nenhum valor associado à chave informada seja encontrado, o programa cliente imprime "not found"no dispositivo de saída padrão.

`update <key,new-value>`
Atualiza o objeto que é identificado pela chave `key`. Ao concluir a operação, o programa cliente imprime "updated"no dispositivo de saída padrão.

`quit`
Encerra a execução do programa cliente do banco de dados.

O Código 1 ilustra o uso do programa cliente do banco de dados.

```
$ simpledb-client
insert 1,pedro
inserted
insert 2,banana
inserted
updated 2,apple
updated
search 3
not found
search 1
pedro
quit
```

Código 1: Exemplo de uso do programa cliente do banco de dados.

4. Entrega e Distribuição de Pontos

O projeto deve ser desenvolvido em uma das seguintes linguagens: `C`, `C++`, `C#`, `Python`, `Java`, `GoLang` ou `Rust`.

O projeto deverá ser necessariamente desenvolvido usando o sistema de versionamento Git. Para hospedar a árvore de fontes, qualquer plataforma de acesso público, como o GitHub, BitBucket ou então GitLab, pode ser usada. Na árvore de fontes hospede todos os artefatos relevante do projeto, incluindo documentação e informações suficientes para a compilação.

O link do repositório Git contendo a árvore de fontes do projeto deverá ser entregue em um arquivo texto no Canvas da PUC Minas, antes do prazo para entrega estipulado. *Commits* realizados no repositório após o prazo de entrega do trabalho serão desconsiderados.

Este trabalho deve ser desenvolvido em grupo de três a quatro integrantes e será entregue em diferentes partes, cada uma composta pela seguinte coleção de requisitos:

- **Parte A** (30% dos pontos): RF-1, RF-2, RF-3, RF-4, RF-5, RFN-1

- **Parte B** (40% dos pontos): RF-6, RF-7, RF-8, RF-9, RFN-2, RFN-3, RNF-7, RFN-8
- **Parte C** (30% dos pontos): RFN-4

O trabalho será avaliado da seguinte forma:

1. Corretude da Solução (30% dos pontos)
2. Aderência da Solução à Especificação (30% dos pontos)
3. Qualidade e Documentação de Código (20% dos pontos)
4. Participação de Todos os Integrantes do Grupo (20% dos pontos)

O "Item 4. Participação de Todos os Integrantes do Grupo" será avaliado de forma objetiva:

- Caso todos os membros do grupo tenham realizado ao menos um *commit* relevante na árvore de fontes do projeto e/ou atuado na gestão do projeto de forma relevante (*ie.* criação de *cards*, bugs, *pull requests*, *merges*) todos os os membros do grupo serão contemplados com a nota máxima no item em questão.
- Caso contrário, todos os membros do grupo serão contemplados com a nota zero no item em questão.

Discussões entre grupos são encorajadas. No entanto, qualquer identificação de cópia do trabalho, total ou parcial, implicará na avaliação em zero para todas as partes envolvidas.