Universidade Federal Fluminense

Instituto de computação

Técnicas de Programação Avançada – TCC00175

Professor: Luiz André P. Paes Leme

Gerenciador de Dados

Grupo:

João Felipe Nicolaci Pimentel

Lucas de Souza Nadalutti

Luis Antônio Vieira Junior

Rodrigo Castro Azevedo

I – Introdução

Atualmente, é comum o uso de sistemas gerenciadores para manipular bancos de dados (SGBD's). Operações como criação e exclusão de tabelas, inserção, exclusão e modificação de registros, busca de registros segundo certas características, entre outras, são implementadas diretamente por um SGBD para facilitar a comunicação do usuário com o banco.

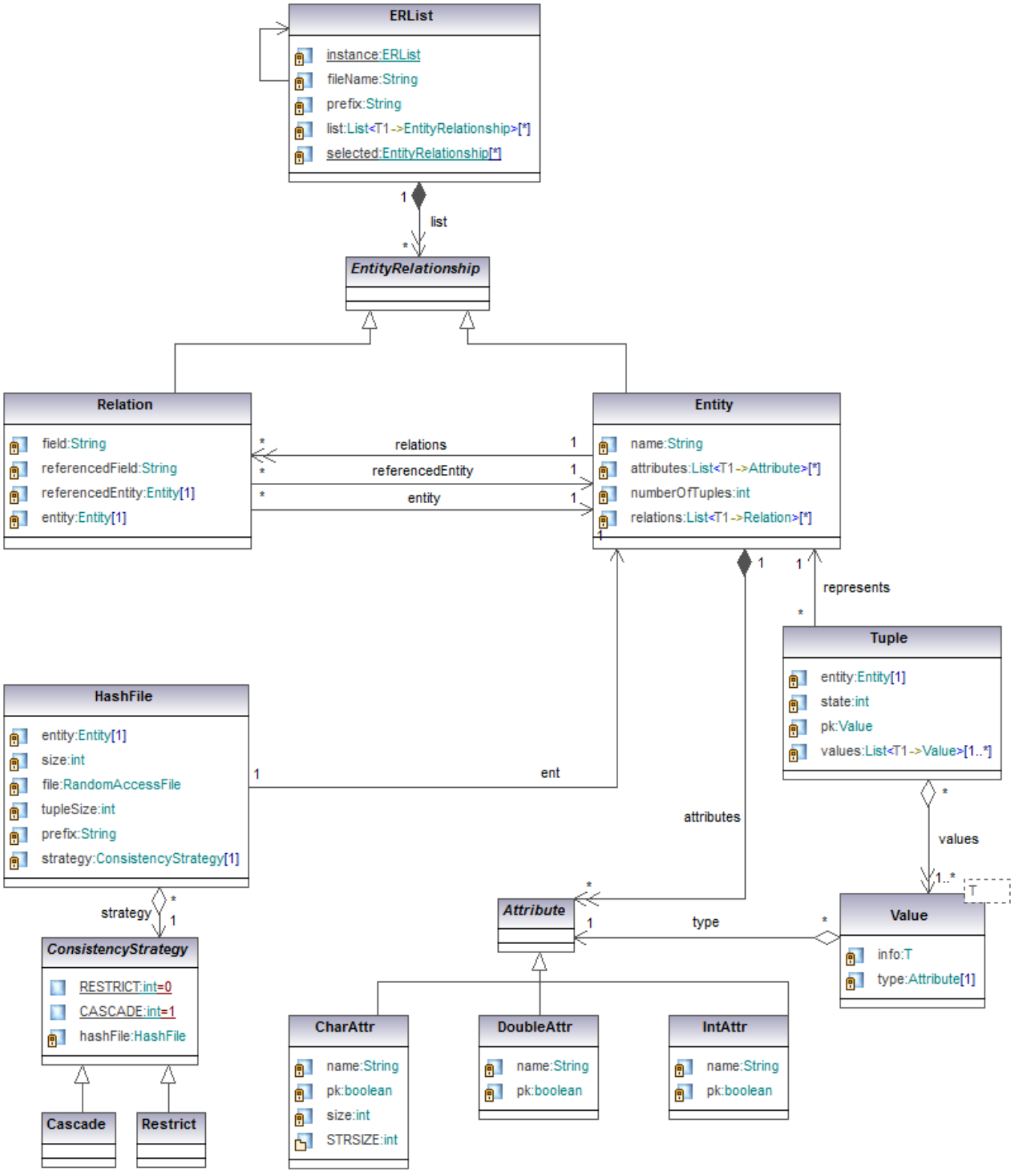
Este trabalho visa a realizar algumas operações comuns em sistemas gerenciadores. Estão disponíveis as seguintes opções:

- Criação de uma nova tabela, declarando seu nome e seus campos (atributos). A declaração de cada campo se dá da seguinte forma: o usuário define o nome e o tipo do atributo, podendo este ser um inteiro, um real ou uma sequência fixa de caracteres. O usuário deve declarar um dos atributos como chave primária, e também pode fazer com que um ou mais deles referenciem outras tabelas.
- Remoção de tabelas.
- Inserção de registros em uma tabela, isto é, instâncias de cada um dos atributos da tabela. Por exemplo, um registro possível para uma tabela cujos atributos são "CPF", "nome" e "idade" seria {123.456.789-01, Maria, 30}.
- Remoção de registros em uma tabela. Remove-se um conjunto de registros segundo uma determinada condição, como por exemplo: remover todos os registros cujo atributo "idade" seja menor que 30. Para remover registros individualmente bastos remover sob a condição de que a chave primária seja igual à chave do registro que se quer excluir, como por exemplo: remover do banco a pessoa cujo CPF é 123.456.789-01.
- Modificação de registros em uma tabela, seguindo condições estabelecidas pelo usuário, analogamente à remoção.
- Consulta de registros de uma tabela, de forma que sejam retornados para o usuário todos os registros que atendem todas as condições impostas pelo usuário para a consulta.

Utilizamos no projeto uma estrutura de dados chamada Arquivo de Hash para salvar cada tabela que é criada, de forma que seus registros podem ser buscados de uma forma otimizada. Estes arquivos Hash são indexados por outro arquivo, chamado Arquivo de Metadados, que possui o endereçamento de cada arquivo relacionado com as tabelas do banco de dados.

II – Modelos de Classes UML

1 – Models



O diagrama de classes dos Models do projeto tem como base a classe ERList, classe que é a lista de Entidades e Relacionamentos do banco de dados. Esta classe possui os dados que serão salvos no arquivo de metadados, além de saber como são feitas as buscas, remoções, inserções, e como salvar estes dados neste mesmo arquivo. Além disso, ela pode nos informar qual é a tabela que está atualmente selecionada.

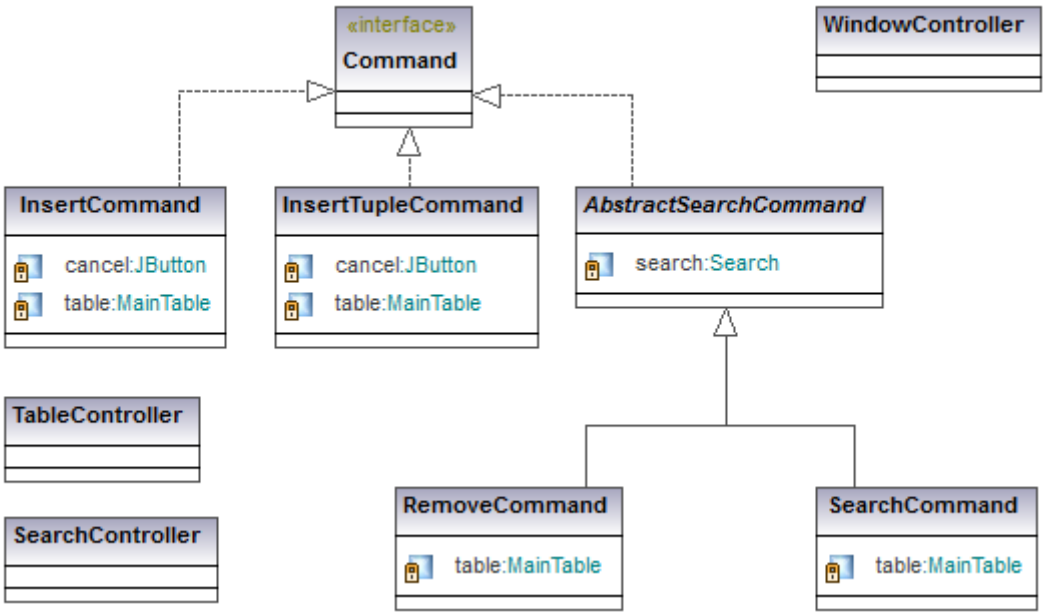
As Entidades e Relacionamentos salvos na ERList são objetos das classes Entity e Relation, que herdam da classe abstrata EntityRelationship, que tem como funcionalidades básicas a criação, remoção das Entidades e dos Relacionamentos. Outra funcionalidade da classe EntityRelationship que é herdada por ambas as classes filhas, é a operação de save, na qual é definido como se salva a entidade e o relacionamento na tabela de metadados.

As tabelas do banco de dados são compostas de tuplas, representadas através da classe Tuple, que possui os valores que serão registrados em uma das linhas da tabela. A classe Tuple também tem outras funcionalidades como: recuperar a chave primária, recuperar a sua tabela referente, além de outras operações básicas.

Os valores salvos em cada tupla pertencem à classe Value, classe que contém o valor e o tipo de atributo a ser salvo em um dos campos da tupla. Esse atributo é um objeto da classe Attribute, que é uma classe abstrata, herdada pelas classes CharAttr, DoubleAttr e IntAttr, que representam respectivamente os atributos de cadeia de caracteres, Double e Integer.

Todas as tuplas de uma entidade ou um relacionamento são salvos em um arquivo que é um objeto da classe HashFile, classe que contém todos os tipos de operação básica de um arquivo, além de poder: recuperar sua entidade referente, tamanho de cada tupla, buscar por valores, reorganizar e salvar o arquivo e também podemos setar que tipo de ação queremos que ele tome ao se modificar ou deletar uma de suas tuplas, se queremos que ele execute um cascade ou restrict.

2 – Controllers



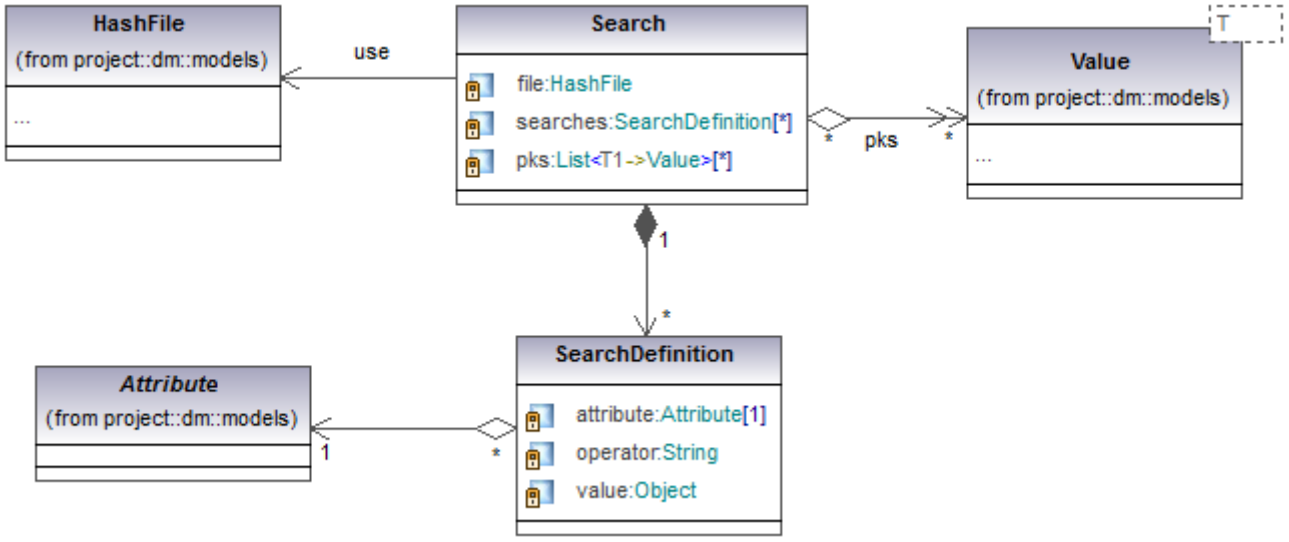
O diagrama de Controllers possui a interface Command, que define diferentes tipos de comandos. Um comando pode ser de inserção de uma linha na tabela (InsertCommand), ou inserção de um registro na mesma (InsertTupleCommand). Existe também a classe abstrata AbstractSearchCommand, da qual herdam os comandos de remoção de registros (RemoveCommand) e consulta de registros (SearchCommand), ambas as operações feitas segundo as condições impostas pelo usuário.

Outra classe de controle do diagrama é a SearchController, que contém todos os métodos relacionados à consulta. A classe inicia a busca, recebendo a tabela de condições, e além disso é responsável também por efetuar modificações nos registros, quando for requisitado que as mesmas sejam operadas.

TableController controla todas as operações relacionadas às tabelas do banco de dados, incluindo criação, remoção e referenciação entre elas.

Finalmente, WindowController controla basicamente o que acontece com a janela de interface, como quando ela deve abrir, fechar, e inclusive quando terminar o programa.

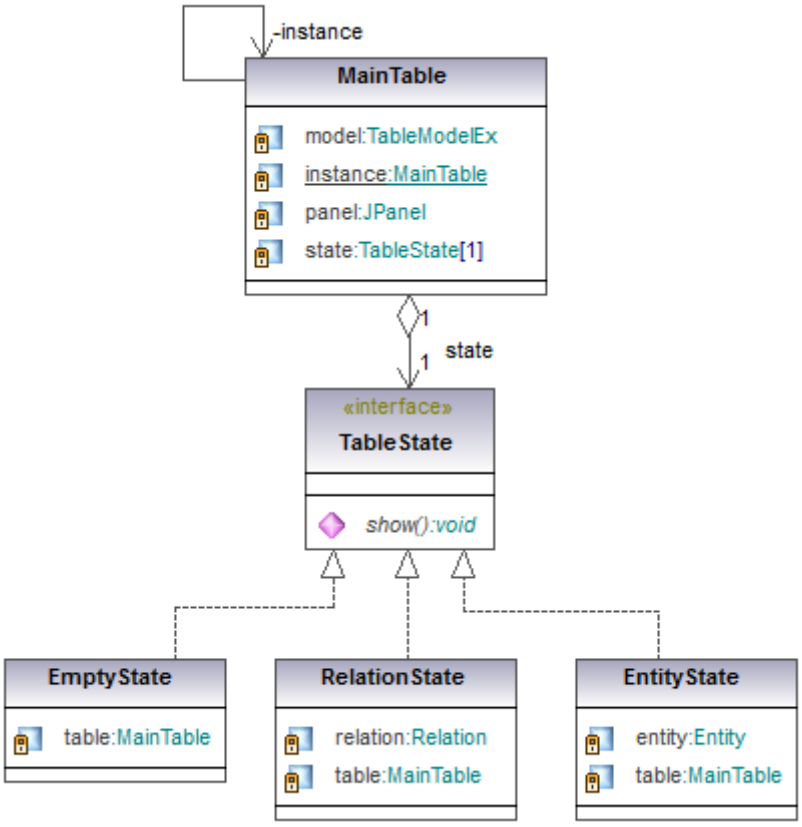
3 – Search



O diagrama Search possui a classe Search que tem métodos que realizam a busca em si, verificando cada condição individualmente e filtrando os registros que serão buscados. A busca é então compilada, ou seja, realizada propriamente dita no arquivo.

SearchDefinition é a classe que define cada condição utilizada na busca, como o atributo a ser filtrado, o operador a ser aplicado e o valor buscado.

4 – MainTable



O diagrama MainTable contém as classes que gerenciam a tabela principal de exibição dos registros das entidades e relacionamentos. A classe MainTable possui um atributo do tipo TableState, que é uma interface utilizada para definir o estado atual da tabela.

Existem também 3 outras classes que implementam TableState, que são: EmptyState, utilizada quando não há nenhuma entidade ou relacionamento a ser exibido, RelationState, utilizada quando há uma relação a ser exibida, e EntityState, para exibir uma entidade.

III – Padrões de Projeto

1 – MVC

O padrão MVC é utilizado em todo o projeto, separando as classes nos grupos de models, controllers e views.

Os models são as classes que lidam com os objetos a serem persistidos, tais como: Entity, Relation, ERList, Attribute, Tuple, HashFile, etc.

Os controllers são as classes que fazem a comunicação entre os models e as views. Algumas dessas classes são: Command, SearchController, WindowsController, etc.

Por fim, as views são as classes que implementam a interface em si do projeto, a comunicação final do sistema com o usuário. Os métodos dos controllers são chamados na view, para por fim estes chamarem os models que realizam várias funções fundamentais para o sistema.

2 – Command

O padrão Command é utilizado para atribuir funções diferentes para certo componente da interface em tempo de execução. A classe Command é apenas uma interface para suas implementações, que fazem ações diferentes no programa.

InsertCommand insere uma nova linha no final da MainTable e altera a interface para permitir a inclusão de um novo registro. InsertTupleCommand por sua vez, que é utilizado no mesmo botão da interface, pega os dados inseridos pelo usuário na nova linha e grava-os no banco de dados.

AbstractSearchCommand tem ainda 2 generalizações, RemoveCommand, que executa a ação para remover registros do banco, de acordo com a busca realizada, e SearchCommand, que exibe o resultado da busca realizada na MainTable.

3 – Singleton

Neste padrão, utiliza-se uma variável estática do mesmo tipo da classe como instância única. O objetivo deste padrão é nunca ter mais de uma instância criada ao mesmo tempo no programa e ter acesso a essa instância em qualquer parte do mesmo. Ela é implementada através da variável estática do mesmo tipo e do método getInstance(), que retorna a instância atual ou cria uma nova, caso não tenha nenhuma instância criada.

As classes do projeto que utilizam o padrão Singleton são: MainTable, ERList, MainWindow e DataManager.

4 – State

O padrão State é utilizado para definir estados diferentes para certa classe, esta podendo assumir comportamentos diferentes dados certos estados.

É utilizado na classe MainTable, para decidir como esta deve ser exibida. Os estados possíveis são: EmptyState, que exibe a tabela vazia quando não há nenhuma entidade ou relacionamento escolhido, RelationState, que exibe na tabela as informações do relacionamento e nenhuma opção extra, e EntityState, que exibe na tabela os registros de uma entidade, e adiciona também componentes na interface para realizar busca, alteração e remoção de registros.

5 – Iterator

O padrão Iterator é usado para percorrer os registros de uma tabela seguindo a ordem de conflitos definida pelo cálculo do hash por dispersão dupla.

A classe IterableTuple foi criada para poder usar o TupleIterator dentro de um foreach.

6 – Strategy

Strategy foi utilizado para escolher como os registros seriam manuseados, mais especificamente para definir o método que garante a integridade dos dados.

Existem duas estratégias sendo utilizadas pelo sistema: Cascade e Restrict. Cascade faz com que ao modificar ou remover um registro, se a integridade referencial dele for quebrada, a modificação ou deleção é passada também para o registro referenciado, a fim de manter a integridade. Restrict simplesmente impede que o usuário complete a ação caso a integridade seja quebrada.

Para se definir a estratégia a ser utilizada, foi utilizada uma configuração na interface, que se aplica a todas as tabelas, referências e operações do sistema. O usuário pode alterar a estratégia a qualquer momento.

7 – Builder

O padrão builder é utilizado na consulta de registros, de forma que o usuário pode adicionar quantas condições ele quiser, e a consulta vai criando uma lista de condições para depois ser utilizada na busca (Search). Nesse processo, a cada condição sendo adicionada, Search retorna uma instância de si mesma, adicionando a nova condição à lista de condições.

No final do processo é executado o método compile, que realiza a busca no arquivo.

8 – Factory Method

O padrão Factory Method é utilizado pelo Attribute, para criar novos atributos. O método createAttribute cria o atributo baseado no tipo escolhido, selecionando a classe que vai gerenciar o atributo, podendo ser: CharAttr, IntAttr ou DoubleAttr.

EntityRelationship também utiliza este padrão, para criar uma EntityRelationship que pode ser uma Entity ou uma Relation.

9 – Template Method

Template Methods são métodos que são definidos e utilizados em uma classe abstrata, mas a implementação deles ocorre apenas nas classes que a herdam. A classe ConsistencyStrategy utiliza este padrão, com os métodos verifyAndApplyModify e verifyAndApplyRemove, que são chamados na própria classe ConsistencyStrategy mas apenas são implementados nas classes Cascade e Restrict.