UFRN - Universidade Federal do Rio Grande do Norte
IMD0012 - Introdução às Técnicas de Programação
2018.2 - 3ª Avaliação


SGBD ITP

Introdução - Sistema de gerenciamento de banco de dados

Um Sistema de Gerenciamento de Banco de Dados (SGBD) — do inglês *Data Base Management System* (DBMS) — é o conjunto de softwares responsáveis pelo gerenciamento de um banco de dados. Seu principal objetivo é retirar da aplicação cliente a responsabilidade de gerenciar o acesso, a persistência, a manipulação e a organização dos dados. O SGBD disponibiliza uma interface para que seus clientes possam incluir, alterar ou consultar dados previamente armazenados. Em bancos de dados relacionais a interface é constituída pelas APIs (Application Programming Interface) ou drivers do SGBD, que executam comandos na linguagem SQL (Structured Query Language). Um modelo de SGBD define como os dados serão armazenados no banco de dados. Os quatro modelos mais conhecidos são:

- hierárquico;
- em rede;
- relacional;
- orientado a objetos

Banco de dados relacional

Um banco de dados relacional é um banco de dados que modela os dados de uma forma que eles sejam percebidos pelo usuário como tabelas, ou mais formalmente relações. O termo é aplicado aos próprios dados, quando organizados dessa forma, ou a um SGBD Relacional que é um programa de computador que implementa a abstração.

Os Bancos de dados relacionais (BDR) surgiram em meados da década de 1970. Porém, apenas alguns anos mais tarde as empresas passaram a utilizá-los no lugar de arquivos simples (do inglês flat file), bancos de dados hierárquicos e em rede. Em 1985, Edgar Frank Codd, criador do modelo relacional, publicou um artigo onde definia 13 regras para que um Sistema Gerenciador de Banco de Dados (SGBD) fosse considerado relacional.

Regra mental

Um SGBD relacional deve gerir os seus dados usando apenas suas capacidades relacionais seguindo as seguintes regras mentais.

- Regra da informação:

    - Toda informação deve ser representada de uma única forma, como dados em uma tabela.
- Regra da garantia de acesso:
    - Todo o dado (valor atómico) pode ser acedido logicamente (e unicamente) usando o nome da tabela, o valor da chave primária da linha e o nome da coluna.
- Tratamento sistemático de valores nulos:
    - Os valores nulos (diferente do zero, da string vazia, da string de caracteres em brancos e outros valores não nulos) existem para representar dados não existentes de forma sistemática e independente do tipo de dado.
- Catálogo dinâmico on-line baseado no modelo relacional:
    - A descrição do banco de dados é representada no nível lógico como dados ordinários (isto é, em tabelas), permitindo que usuários autorizados apliquem as mesmas formas de manipular dados aplicada aos dados comuns ao consultá-las.
- Regra da sub-linguagem abrangente:
    - Um sistema relacional pode suportar várias linguagens e formas de uso, porém deve possuir ao menos uma linguagem com sintaxe bem definida e expressa por cadeia de caracteres e com habilidade de apoiar a definição de dados, a definição de visões, a manipulação de dados, as restrições de integridade, a autorização e a fronteira de transações.
- Regra da atualização de visões:
    - Toda visão que for teoricamente atualizável será também atualizável pelo sistema.
- Inserção, atualização e eliminação de alto nível:
    - Qualquer conjunto de dados que pode ser manipulado com um único comando para retornar informações, também deve ser manipulado com um único comando para operações de inserção, atualização e exclusão. Simplificando, significa dizer que as operações de manipulação de dados devem poder ser aplicadas a várias linhas de uma vez, ao invés de apenas uma por vez.
- Independência dos dados físicos:
    - Programas de aplicação ou atividades de terminal permanecem logicamente inalteradas quaisquer que sejam as modificações na representação de armazenagem ou métodos de acesso internos.
- Independência lógica de dados:
    - Programas de aplicação ou atividades de terminal permanecem logicamente inalteradas quaisquer que sejam as mudanças de informação que permitam teoricamente a não alteração das tabelas base.
- Independência de integridade:
    - As relações de integridade específicas de um banco de dados relacional devem ser definidas em uma sub-linguagem de dados e armazenadas no catálogo (e não em programas).
- Independência de distribuição:
    - A linguagem de manipulação de dados deve possibilitar que as aplicações permaneçam inalteradas estejam os dados centralizados ou distribuídos fisicamente.
- Regra da Não-subversão:
    - Se o sistema relacional possui uma linguagem de baixo nível (um registro por vez), não deve ser possível subverter ou ignorar as regras de integridade e restrições definidas no alto nível (muitos registros por vez).

Por que usar um Banco de Dados Relacional?

Os Bancos de Dados Relacionais foram desenvolvidos para prover acesso facilitado aos dados, possibilitando que os usuários utilizassem uma grande variedade de abordagens no tratamento das informações. Pois, enquanto em um banco de dados hierárquico os usuários

precisam definir as questões de negócios de maneira específica, iniciando pela sua raiz , nos Bancos de Dados Relacionais os usuários podem fazer perguntas relacionadas aos negócios por meio de vários pontos. A linguagem padrão dos Bancos de Dados Relacionais é a Structured Query Language, ou simplesmente SQL, como é mais conhecida.

O Modelo Relacional

Um Banco de Dados Relacional segue o Modelo Relacional. A arquitetura de um banco de dados relacional pode ser descrita de maneira informal ou formal. Na descrição informal estamos preocupados com aspectos práticos da utilização e usamos os termos tabela, linha e coluna. Na descrição formal estamos preocupados com a semântica formal do modelo e usamos termos como relação (tabela), tupla(linhas) e atributo(coluna).

Tabelas (ou relações, ou entidades)

Todos os dados de um banco de dados relacional (BDR) são armazenados em tabelas. Uma tabela é uma simples estrutura de linhas e colunas. Em uma tabela, cada linha contém um mesmo conjunto de colunas. Em um banco de dados podem existir uma ou centenas de tabelas, sendo que o limite pode ser imposto tanto pela ferramenta de software utilizada, quanto pelos recursos de hardware disponíveis no equipamento. As tabelas associam-se entre si por meio de regras de relacionamentos, que consistem em associar um ou vários atributos de uma tabela com um ou vários atributos de outra tabela.

Exemplo: A tabela funcionário relaciona-se com a tabela cargo. Por este relacionamento, esta última tabela fornece a lista de cargos para a tabela funcionário. Modelo teórico usado para representar conceitualmente um BD, Idealizado por Codd (1970). Baseado numa estrutura de dados simples chamada relação. É o modelo mais amplamente usado, principalmente em aplicações convencionais de BD.

Registros (ou tuplas)

Cada linha formada por uma lista ordenada de colunas representa um registro, ou tupla. Os registros não precisam conter informações em todas as colunas, podendo assumir valores nulos quando assim se fizer necessário. Resumidamente, um registro é uma instância de uma tabela, ou entidade. O start da modelagem se dá a partir das ENTIDADES. Uma entidade é uma representação de um conjunto de informações sobre determinado conceito do sistema. Toda entidade possui ATRIBUTOS, que são as informações que referenciam a entidade. Para exemplificar no sistema de controle de Biblioteca, partimos do conceito principal que é o empréstimo de obras por usuários da biblioteca. A partir deste conceito inicial, vamos ramificando e descobrindo novos conceitos. Podemos iniciar nosso raciocínio da seguinte forma:

"Uma biblioteca possui Obras literárias que podem ser tomadas em empréstimos pelos usuários credenciados."

Podemos rapidamente enxergar um cadastro de livros, um cadastro de usuários e um registro de empréstimos, certo? É essa visão que temos que ter ao modelarmos um banco, isto é, devemos detectar as informações que devemos armazenar.

Para identificar se aquele conceito pode ser uma entidade você deve apenas se perguntar: "Eu desejo armazenar quais informações sobre este conceito?" Se houverem informações a serem armazenadas, você tem uma ENTIDADE. Exemplificando: Eu desejo armazenar os seguintes dados do livro: Título, Autor, Editora, Ano, Edição e Volume. Temos então a entidade Livro.

Exemplo: O empregado Pedro é uma instância (registro) da tabela funcionário, e a função Analista Comercial é a instância (registro) da tabela cargo. Uma associação entre estas duas tabelas criaria a seguinte instância de relacionamento: Pedro é Analista Comercial, onde o verbo ser representa uma ligação entre os registros distintos.

Colunas (atributos)

As colunas de uma tabela são também chamadas de atributos. Ex.: O campo Nome, ou endereço de uma tabela de um BD relacional.

Chave

As tabelas relacionam-se umas às outras através de chaves. Uma chave é um conjunto de um ou mais atributos que determinam a unicidade de cada registro.

Por exemplo, se um banco de dados tem como chaves Código do Produto e ID Sistema, sempre que acontecer uma inserção de dados o sistema de gerenciamento de banco de dados irá fazer uma consulta para identificar se o registro já não se encontra gravado na tabela. Neste caso, um novo registro não será criado, resultando esta operação apenas da alteração do registro existente.

A unicidade dos registros, determinada por sua chave, também é fundamental para a criação dos índices. Temos dois tipos de chaves:

Chave primária: (PK - Primary Key) é um identificador exclusivo de todas as informações de cada registro dando-lhe unicidade. A chave primária nunca se repetirá.[1]

Chave Estrangeira: (FK - Foreign Key) é a chave formada através de um relacionamento com a chave primária de outra tabela. Define um relacionamento entre as tabelas e pode ocorrer repetidas vezes. Caso a chave primária seja composta na origem, a chave estrangeira também o será.

Fonte: Wikipédia, a enciclopédia livre.

Descrição do projeto

O projeto consiste em implementar um Sistema de Gerenciamento de Banco de Dados simplificado baseado no modelo relacional. Não é o objetivo desse projeto criar um SGBD para ser utilizado em produção por sistemas de informação, mas “apenas” explorar os conceitos abordados nas disciplinas ITP/PTP.

O SGBD ITP (nome dado ao SGBD simplificado) deverá ser capaz de realizar as seguintes operações:

1. Criar um tabela
    a. os tipos de dados para as colunas poderão ser os tipos primitivos em C (char, int, float e double) e strings
    b. os valores deverão ser armazenados em arquivo
    c. na criação da tabela deverá ser solicitado um nome de coluna para ser a chave primária
        i. a chave primária deverá ser obrigatoriamente do tipo inteiro sem sinal
2. Listar todas as tabelas
    a. deverá mostrar para o usuário as tabelas existentes
3. Criar uma nova linha na tabela
    a. usuário deve informar o nome da tabela
    b. sistema deve solicitar os valores de cada uma das colunas
    c. sistema deve verificar a chave primária
        i. em uma tabela deve existir um e apenas um valor de chave primária. Se o usuário informar uma chave que já existe, sistema deve emitir uma mensagem de erro e não deve inserir o registro
4. Listar todos os dados de uma tabela
    a. usuário deve informar qual a tabela para serem listados os dados
    b. os dados deverão ser obtidos a partir do arquivo que armazena as tabelas
5. Pesquisar valor em uma tabela
    a. usuário deverá informar o nome da tabela onde realizará a pesquisa
    b. sistema deverá fornecer as colunas disponíveis na tabela o usuário deverá selecionar uma delas
    c. sistema deverá solicitar o valor para pesquisar, disponibilizando algumas opções
        i. valores maior que o valor informado
        ii. valores maior ou igual que o valor informado
        iii. valores igual o valor informado
        iv. valores menor que o valor informado
        v. valores menor ou igual que o valor informado
        vi. valores próximo ao valor informado
            1. se aplica apenas se a coluna for do tipo string
6. Apagar valor de uma tabela
    a. usuário deve informar o nome da tabela e a chave primária a ser apagada
7. Apagar uma tabela
    a. usuário deverá fornecer o nome da tabela a ser apagada

Requisitos da linguagem

O projeto de SGBD ITP deve ser desenvolvido em linguagem C, a ser executado, em sua versão mais simples, através de linha de comando (entrada e saída em um console/terminal). O projeto deve atender os seguintes critérios de programação:

1. **Alocação dinâmica de arranjos e/ou matrizes;**

2. **Uso de registros (struct);**
3. **Definição de novos tipos de dados através de typedef;**
4. **Leitura/escrita a partir de arquivos;**
5. **Modularização do programa em diferentes funções (modularização interna) e arquivos (uso de diferentes arquivos .c e .h, cada um com sua funcionalidade - modularização externa);**
6. **Boas práticas de programação: Definição de um padrão de indentação do código fonte e de nomenclatura das sub-rotinas e variáveis;**
7. **Emitir mensagens para stderr em situações de erro: falha na alocação de memória, falha na abertura do arquivo.**
8. **Documentação adequada do código-fonte (uso de comentários).**

Cronograma do projeto

Checkpoint 01 - 19/11/2018
- Operações #1 e #2

Checkpoint 02 - 23/11/2018
- Operações #3 e #4

Checkpoint 03 - 26/11/2018
- Operação #5

Checkpoint 04 - 30/11/2018
- Operações #6 e #7

Check point - atrasados - última chance! 03/12/2018

Sobre as duplas

Os alunos têm ATÉ o dia 31 de outubro de 2018 para comunicar aos professores de ITP e PTP se farão o trabalho em dupla (e a composição da mesma) ou se farão individualmente.

Critérios de pontuação

O desenvolvimento do projeto aqui descrito vale 100% da nota da terceira unidade de ITP/PTP.

A pontuação da avaliação seguirá os critérios e distribuição abaixo:

- Atendimento dos requisitos funcionais: 50%
  - O SGBD ITP encontra-se funcional, com todas as regras definidas implementadas corretamente?
- Uso dos recursos da linguagem C: 20%
  - A dupla demonstrou saber usar de forma adequada os recursos da linguagem C (arranjos, structs, typedefs, ponteiros, arquivos, etc)?
- Organização do código e documentação: 20%
  - O código está documentado? A indentação e uso de { } seguem um padrão? O programa está devidamente modularizado em diferentes arquivos?
- Extras: 10%
  - Funcionalidades extras que não foram descritas nesse documento. A avaliação das funcionalidades extras será comparativa, ou seja, um dos grupos terá a nota máxima e os demais que fizerem funcionalidades extras receberão a nota proporcional.

Entrega do projeto

O projeto deve ser submetido pelo SIGAA até a data 03 de dezembro de 2018 (8:40) em um arquivo comprimido (.ZIP) contendo os arquivos fontes do projeto (.c e .h) e um arquivo README.TXT. Este arquivo deve ter as seguintes informações:

- O que foi feito (quais funcionalidades solicitadas foram implementadas e alguma funcionalidade extra que tenha sido implementada);
- O que não foi feito (caso não tenha sido possível implementar alguma funcionalidade solicitada);
- O que faria de forma diferente (quando aprendemos à medida que vamos implementando, por vezes vemos que poderíamos ter implementado algo de forma diferente. Assim, esse tópico é para vocês refletirem sobre o que vocês aprenderam durante o processo e, se fossem fazer o mesmo projeto novamente, fariam diferente);
- Como compilar o projeto, deixando explícito se foi utilizada alguma biblioteca externa que precise ser instalada, que versão e quais parâmetros extras são necessários para o compilador.
- Em caso de duplas:
    - Identificação dos autores;
    - Contribuição de cada integrante no desenvolvimento do projeto (quem fez o quê).
- O que foi feito de funcionalidade extra e a justificativa da adição dessa funcionalidade.