João Vitor Gonçalves da Silva – 18.1.4006

BCC321 – Banco de Dados

Resumo IV – Normalização e Noções sobre Transação, Concorrência e Falha

1. **Normalização**

A normalização consiste em um conjunto de boas práticas para garantir a avaliação formal de qualidade de um esquema relacional. Existem, no total, 4 quesitos de avaliação em um esquema formal:

* Semântica dos atributos;
* Valores redundantes em tuplas;
* Valores nulos em tuplas;
* Possibilidade de geração de tuplas espúrias.
  1. **Semântica dos Atributos**

Esse quesito parte de que todos os atributos devem ter algum significado. Esse significado deve ser dedutível pelo nome do atributo. Ex:

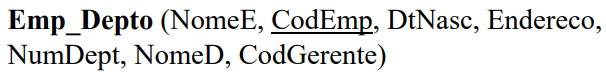


Na relação “Empregado”, os atributos remetem, respectivamente, ao nome do empregado, código, data de nascimento, endereço e o número do departamento que aquele empregado trabalha.

Dessa maneira, a norma informal 1 garante que o esquema relacional seja de fácil explicação de seus significados. Além disso, não se deve combinar atributos de vários tipos de entidades e relacionamentos em uma única relação.

* 1. **Valores redundantes em tuplas**

A segunda norma visa minimizar espaços de armazenamento “extras” que valores redundantes possam causar, além das anomalias de atualização que a mesma possa causa. Ex:



Na relação “Emp\_Depto”, os atributos de Empregado são repetidos nessa relação de forma desnecessária, já que possui o código do empregado e isso seria suficiente para referencia-la.

A norma informal 2 indica que os esquemas de relação sejam projetados sem nenhuma anomalia de atualização. Caso for interessante a fim de melhorias de desempenho, deve-se especificar com clareza e garantir que as atualizações estejam sendo feitas corretamente.

* 1. **Valores nulos em tuplas**

Essa norma indica evitar que relações gerem vários valores nulos, já que estes podem causar desperdício de armazenamento, problemas em aplicações de operações de JOIN e de funções de agregação como COUNT, além das várias maneiras que o nulo pode ser interpretado.

Deve-se certificar que a relação não possua valores que possam ser frequentemente nulos. Se forem inevitáveis, deve-se assegurar que aconteça apenas em atributos específicos e não na maioria da relação.

* 1. **Possibilidade de Geração de Tuplas Espúrias**

Deve-se certificar que as relações sejam projetadas de forma que ao aplicar junções, essas sejam feitas por meio de chave primaria/estrangeira, para que não gere tuplas invalidas.

1. **Forma Normal**

A normalização de dados é um processo que possui três normas a fim de garantir a diminuição de redundância no projeto relacional. Ele analisa baseado com as dependências funcionais e chaves primarias.

* 1. **Primeira Forma Normal**

Para que a relação esteja na 1FN, esta não deve ter atributos multivalorados, compostos ou complexos. Deve-se analisar o caso de haver atributos não-atômicos para resolver da maneira correta.

* 1. **Segunda Forma Normal**

Para estar na 2FN, a relação deve estar na 1FN e todos os atributos não chave devem estar dependentes totais de sua chave primaria, ou seja, precisa da chave primária para ser identificado. Para transformar a relação na 2FN, cada dependência parcial dará origem a uma nova relação, para que os atributos possuam dependência funcional total.

* 1. **Terceira Forma Normal**

Para que uma relação esteja na 3FN, deve estar na 2FN e não possuir relações que tenha uma dependência funcional indireta entre um determinado atributo não chave com a chave primária da relação, por meio de um conjunto de atributos não chaves. Para transformar na 3FN, a dependência indireta dará origem a duas novas relações, com ligação de chave primária/estrangeira entre elas.

1. **Transações**

Uma transação é formada por uma sequencia de operações que precisam ser executadas com frequência para garantir a consistência e a precisão do banco de dados. Normalmente, é composta por uma ou mais instruções DML, uma instrução DLL e uma instrução DCL. Ela começa assim que uma instrução SQL é executada e termina quando há um comando de COMMIT ou ROLLBACK é acionado, automático ou não.

Algumas instruções de transação são:

* COMMIT: finaliza a transação e torna as alterações permanentes;
* SAVEPOINT <nome>: marca um ponto de gravação dentro da transação atual;
* ROLLBACK [TO SAVEPOINT <nome>]: finaliza a transação atual e descarta as alterações de dados pendentes.

**3.3. Processamento de Transações**

Algumas propriedades devem ser seguidas ao processar uma transação:

* Atomicidade: uma transação é uma unidade atômica de processamento;
* Consistência: uma transação é consistente se ela levar o banco de dados de um estado consistente para outro também consistente.
* Isolamento: uma execução não deve colidir com outras transações;
* Durabilidade: as alterações aplicadas ao banco de dados devem persistir no banco de dados.

Um processamento de transação também pode possuir as seguintes operações:

* Read\_item(x): lê um item X do banco de dados e transfere pra uma variável X da memória;
* Write\_item(x): escreve um valor de uma variável X de memória em um item X no banco de dados.

1. Controle de Concorrência

O controle de concorrência entre as transações deve ser utilizado para garantir que elas executem de forma correta. Alguns problemas que possam ocorrer caso as transações concorrentes não estejam controladas:

* Problema de perda de atualização: Ocorre quando duas transações que acessam os mesmos itens no banco de dados possuem operações conjuntas;
* Problema de atualização temporária: Ocorre quando uma transação atualiza um item no banco de dados e, por algum motivo, a transação falha;
* Problema da agregação incorreta: Ocorre quando uma transação calcula uma função de agregação em um grupo de itens e outra transação esteja atualizando esses itens ao mesmo tempo;
* Problema da leitura não-repetitiva: Ocorre quando uma transação lê um item duas vezes e uma outra transação altera esse item entre as duas leituras.

1. Recuperação de Falhas

O SGBD não pode permitir que algumas transações ocorram enquanto outras não ocorrerem, que é recorrente onde uma transação falha e a outra continua com sua execução. Caso a falha seja relacionada ao software, o sistema deve manter informações suficientes para se recuperar da falha. Alguns tipos são:

* Falha no computador;
* Erro na transação ou de sistema;
* Imposição do controle de concorrência;
* Falhas no disco;
* Problemas físicos e catástrofes.

Para a identificação dessas falhas, o SGBD mantém um histórico de todas as transações feitas, chamada de log. Algumas operações que são escritas no log são:

* [start\_transaction, T]: indica que a transação T iniciou sua execução;
* [write\_item, T, X, old\_value, new\_value]: indica que houve uma atualização de valores na transação;
* [read\_item, T, X]: indica que a transação leu o valor do item X do banco de dados;
* [commit, T]: indica que a transação foi feita com sucesso;
* [abort, T]: indica que a transação foi interrompida.

Quando ocorre uma falha, deve-se verificar o log da execução e todas as transações que não finalizaram seu processamento (commit, T) devem ser desfeitas. Para tais ações, existem os operadores que executam sobre falhas, que são:

* UNDO: desfaz a transação;
* REDO: refaz a transação.