Quarto Resumo - Banco de Dados I

🧒 **Aluno** - Fabio Henrique Alves Fernandes | 19.1.4128

🎓 **Professor Doutor** - Guilherme Tavares De Assis

🏠 **Instituição** - Universidade Federal de Ouro Preto | Departamento de Computação

📅 **Data De Entrega** - 14 de dezembro de 2021

Normalização

Normalização é um método de avaliação formal sobre a qualidade de um esquema de relação. Informalmente, existem normas referentes a 4 quesitos avaliativos: a semântica dos atributos, os valores redundantes em tuplas, valores nulos em tuplas e a possibilidade de geração de tuplas espúrias.

Norma informal 1

Diz respeito a semântica dos atributos de cada relação. É sempre interessante que os atributos tenham nomes com significados que estejam associados a eles. Esse significado especifica como interpretar os valores dos atributos dentro da tupla.

Formalizando:

Projete um esquema de relação de modo que seja fácil explicar seu significado. Não combine atributos de vários tipos de entidades e relacionamentos em um única relação; assim, a relação não será confusa em termos semânticos.

Norma Informal 2

Diz respeito a valores redundantes em tuplas. Quando um banco de dados é projetado, devemos tentar ao máximo minimizar o espaço de armazenamento que as relações ocupam.

Valores redundantes em tuplas aumentam o espaço de armazenamento das relações, também podendo gerar anomalias de atualização, classificadas como anomalias de inclusão, de exclusão e de modificação.

Formalizando:

Projete os esquemas de relação de modo que nenhuma anomalia de inclusão, exclusão ou modificação esteja presente nas relações. Se estiverem presentes quaisquer anomalias, mencione-as com clareza e certifique-se de que os programas que atualizam o banco de dados irão operar corretamente.

Norma Informal 3

Diz respeito a valores nulos em tuplas. Se muitos atributos de um esquema de relação não se aplicarem a todas as tuplas, tal relação possuirá vários valores nulos, podendo gerar um desperdício de espaço de armazenamento, problemas na aplicação das operações de junção e funções de agregação e podendo gerar diversas interpretações do valor nulo, como um atributo que não se aplica a tupla, valores desconhecidos e valores não registrados.

Formalizando:

Tanto quanto possível, evite colocar atributos em uma relação cujos valores possam frequentemente ser nulos. Se nulos forem inevitáveis, assegure-se de que eles se aplicam em casos excepcionais e não se aplicam à maioria das tuplas na relação.

Norma Informal 4

Diz respeito a tuplas espúrias geradas em uma junção natural entre duas relações. Quando um projeto é ruim, quando executamos uma junção natural entre duas relações, o resultado produzira um grande número de tuplas inválidas.

Formalizando:

Projete esquemas de relação de modo que possam ser juntados (JOIN) com condições de igualdade em atributos que sejam chave primária/chave estrangeira, de tal forma que garanta que nenhuma tupla espúria seja gerada. Não conserve relações que possuam atributos iguais, além das combinações de chave primária/chave estrangeira; se tais relações forem inevitáveis, não junte as mesmas nesses atributos.

Dependência Funcional

Dizemos que uma dependência funcional entre dois conjuntos de atributos (X e Y) de uma relação R é uma restrição que estabelece que, para quaisquer tuplas de t1 e t2 de R, de t1[X] = t2[X], então t1[Y] = t2[Y]. Isso significa que os valores de Y dependem ou são

determinados de forma única pelos valores de X nas tuplas de R. Simplificando, Y é funcionalmente dependente de X (X → Y). Se X for uma chave candidata da relação R, então X → Y para qualquer subconjunto de atributos Y de R. A operação X → Y não é comutativa, ou seja, X → Y ≠ Y → X.

Normalização

Pode ser vista como um processo de analise das relações com base em suas dependências funcionais e chaves primárias, no intuito de diminuir a redundância. Esse processo fornece uma série de testes de forma normal que, quando realizados em relações individuais, permite a normalização de esquemas relacionais no grau desejado.

A forma normal de uma relação diz respeito a forma normal mais elevada que ela atinge, ou seja, o grau para que ela foi normalizada. É desejável que todas as relações de um esquema relacional estejam na terceira forma normal.

Primeira Forma Normal

Uma relação está na primeira forma normal (1FN) se o domínio de todos os seus atributos possuir apenas valores atômicos, e o valor de qualquer atributo em suas tuplas ser um único valor do domínio desse atributo. Informalmente, para que uma relação esteja na 1FN, seus atributos não podem ser multivalorados, compostos ou complexos.

Segunda Forma Normal

Uma relação está na segunda forma normal (2FN) se estiver na 1FN e se todo atributo não chave possuir dependência funcional total em relação à chave primária da relação. X → Y é uma dependência funcional total se a remoção de qualquer atributo A de X significar que a dependência não mais se mantém; caso contrário, é uma dependência funcional parcial. Informalmente, para que uma relação esteja na 2FN, todo atributo não chave necessita de toda a chave primária para ser identificado.

Terceira Forma Normal

Uma relação está na terceira forma normal (3FN) se estiver na 2FN e se todo atributo não chave não possuir dependência funcional transitiva em relação à chave primária da relação. X → Y é uma dependência funcional transitiva na relação R se existir um conjunto de atributos Z em R, que não seja chave candidata de R, de tal forma que as dependências X → Z e Z → Y existem. Informalmente, para que uma relação esteja na 3FN, não pode existir 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.

Noções de Processamento de Transações, Controle de Concorrência e Recuperação de Falhas

Transação

É uma unidade lógica de processamento de operações sobre um banco de dados. Uma transação é formada por uma sequência de operações que precisam ser executadas integralmente para garantir a consistência e a precisão.

Geralmente, uma transação tem uma dessas instruções: uma ou mais instruções do tipo DML (Data Manipulation Language); uma instrução DDL (Data Definition Language); uma instrução DCL (Data Control Language).

As instruções de controle de transação são:

- COMMIT: finaliza a transação atual tornando permanentes todas as alterações de dados pendentes;
- SAVEPOINT <nome_savepoint>: marca um ponto de gravação dentro da transação atual, sendo utilizado para dividir uma transação em partes menores;
- ROLLBACK [TO SAVEPOINT <nome_savepoint>]: ROLLBACK finaliza a transação atual, descartando todas as alterações de dados pendentes. ROLLBACK TO SAVEPOINT descarta o ponto de gravação determinado e as alterações seguintes ao mesmo.

Uma transação começa quando for executada a primeira instrução SQL executável e termina com um dos seguintes eventos:

Quando acontece um COMMIT ou um ROLLBACK;

Instrução DDL ou DCL é executada;

Desconexão do usuário, causando um COMMIT automático;

Ou o sistema falha e ocorre um ROLLBACK automático.

Quando uma transação termina, o próximo comando SQL inicia automaticamente a próxima transação.

Processamento de Transações

Quando várias transações são emitidas simultaneamente, ocorre um entrelaçamento entre elas. Algumas operações de uma transação são executadas; em seguida, seu processo é

suspenso e algumas operações de outra transação são executadas. Depois, um processo suspenso é retomado a partir do ponto de interrupção, executado e interrompido novamente para a execução de uma outra transação.

Para podermos processar transações concorrentes, devemos seguir algumas propriedades (propriedades ACID):

- Atomicidade: uma transação é uma unidade atômica de processamento; é realizada integralmente ou não é realizada.
- Consistência: uma transação é consistente se levar o banco de dados de um estado consistente para outro estado também
  consistente.
- Isolamento: a execução de uma transação não deve sofrer interferência de quaisquer outras transações que estejam sendo executadas concorrentemente.
- Durabilidade (ou persistência): as alterações aplicadas ao banco de dados, por meio de uma transação confirmada, devem persistir no banco de dados, não sendo perdidas por nenhuma falha.

O modelo simplificado para processamento de transações envolve as operações read_item(X), que lê um item X do banco de dados e transfere para uma variável X da memória e a write_item(X), que escreve um valor de uma variável X de memória em um item X do banco de dados.

Controle de Concorrência

Podemos usar técnicas de controle de concorrência para que transações concorrentes sejam executadas sem nenhum problema. Quando transações concorrentes são executadas sem tratamento podem ocorrer problemas como perda de atualização, leitura suja, agregação incorreta, leitura não-repetitiva, entre outras.

Perda de Atualização

Ocorre quando duas transações que acessam os mesmos itens do banco de dados possuem operações entrelaçadas, de modo que torne incorreto o valor de algum item do banco de dados.

Leitura Suja

Ocorre quando uma transação atualiza um item do banco de dados e, por algum motivo, a transação falha; no caso, o item atualizado é acessado por uma outra transação antes do seu valor ser retornado ao valor anterior.

Agregação Incorreta

Se uma transação estiver calculando uma função de agregação em um número de itens, enquanto outras transações estiverem atualizando alguns desses itens, a função agregada pode considerar alguns itens antes que eles sejam atualizados e outros depois que tenham sido atualizados.

Leitura Não-Repetitiva

Ocorre quando uma transação T lê um item duas vezes e o item é alterado por uma outra transação T' entre as duas leituras de T. Portanto, T recebe diferentes valores para suas duas leituras do mesmo item.

Recuperação de Falhas

O Sistema de Gerenciamento do Banco de Dados geralmente não pode permitir operações que causem transações concorrentes. Isso pode acontecer quando uma transação falha após executar algumas de suas operações.

As falhas que podem ocorrer são:

- falha no computador;
- erro de transação ou de sistema;
- imposição do controle de concorrência;
- falha no disco;
- problemas físicos e catástrofes.

Para as três primeiras falhas, o sistema tem que manter informações suficientes para que possa ser feita uma recuperação.

Para cada transação feita no banco de dados, o gerenciador de recuperação registra dentro do log as operações que afetam os valores dos itens dentro do banco:

- [start_transaction, T]: indica que a transação T iniciou sua execução.
- [write_item, T, X, [write_item, T, X, old_value old_value, new_value new_value]: indica que a : indica que a transação T alterou o valor do item X do banco de dados de old_value (valor antigo) para new_value (novo valor).
- [write_item, T, X, [write_item, T, X, old_value old_value, new_value new_value]: indica que a : indica que a transação T alterou o valor do item X do banco de dados de old_value (valor antigo) para new_value (novo valor).
- [read_item, T, X]: indica que a transação T leu o valor do item X do banco de dados.

- [read_item, T, X]: indica que a transação T leu o valor do item X do banco de dados.
- [commit, T]: indica que a transação T foi finalizada com sucesso.
- [abort, T]: indica que a transação T foi abortada.

Quando ocorre uma falha, as transações inicializadas, mas que não gravaram seus registros de COMMIT no log serão desfeitas. Já as transações que gravaram seus COMMITs no log podem ser refeitas a partir dos registros gravados. Para isso, quando temos falhas em uma transação T, contamos com as operações:

- UNDO: desfaz a transação, percorrendo o log e retornando todos os itens alterados aos seus antigos valores;
- REDO: refaz a transação, percorrendo o log e ajustando todos os itens alterados para seus novos valores.

Escalonamento e Recuperabilidade

Um escalonamento S de n transações é uma ordenação das operações das operações dessas transações, onde cada transação Ti que participa de S, as operações de Ti em S devem aparecer na mesma ordem que ocorrem em Ti.

Simplificando as operações:

ri(X): read_item(X) na transação Ti.
wi(X): write_item(X) na transação Ti.
ci: commit na transação Ti.
ai: abort na transação Ti.

Duas operações em um escalonamento são ditas conflitantes se:

pertencem a diferentes transações;
possuem acesso ao mesmo item X;
pelo menos uma delas é uma operação write_item(X).

Um escalonamento S é recuperável se nenhuma transação T em S entrar em estado confirmado até que todas as transações de T' entrem em estado confirmado. Em um escalonamento recuperável, pode ocorrer um ROLLBACK em cascata, onde uma transação não confirmada tenha que ser desfeita porque leu um item de uma transação falha.

Um escalonamento evita ROLLBACKs em cascata se todas as transações no escalonamento lerem somente itens já escritos por transações já confirmadas.

Seriabilidade de Escalonamentos

Um escalonamento é serial quando, para todas as transações T participantes, todas as operações de T forem executadas consecutivamente no escalonamento, senão, dizemos que é não-serial.

Um escalonamento serial:

- possui somente uma transação ativa de cada vez;
- não permite nenhum entrelaçamento de transações;
- é considerado correto, independente da ordem de execução das transações;
- limita a concorrência;
- na prática, é inaceitável.

Um escalonamento S de n transações T é serializável se for equivalente a algum escalonamento serial de todas as n transações T. Dizer que um escalonamento não-serial S é serializável equivale a dizer que ele é correto, já que equivale a um escalonamento serial que é considerado correto. Dois escalonamentos são ditos equivalentes se a ordem de quaisquer duas operações conflitantes for a mesma nos dois escalonamentos.

Teste de Seriabilidade

Uma forma de testar a seriabilidade de um escalonamento é construindo um grafo de precedência. Um grafo de precedência seria um grafo dirigido G = (N, E), onde N é o conjunto de nós e E é um conjunto de arcos dirigidos, tal que cada nodo Ti corresponde a uma transação de S e cada arco ej liga uma transição Tj que possui uma operação conflitante com uma transição Tk.

Técnicas de Bloqueio

Bloqueios são usados como um meio de sincronizar o acesso aos itens do banco de dados por transações concorrentes. Um bloqueio consiste em uma variável associada a um item dado, que descreve o status do item em relação a possíveis operações que podem ser aplicadas ao mesmo. De forma geral, existe um bloqueio para cada item de dado no banco de dados. Existem duas técnicas de bloqueio: o bloqueio binário e o bloqueio múltiplo.

Bloqueio Binário

Um bloqueio binário possui dois estados: bloqueado (locked) e desbloqueado (unlocked). Suas operações necessárias são lock_item(X), que bloqueia o item X e unlock_item(X), que desbloqueia o item X.

Se um determinado item for bloqueado por uma transação, nenhuma outra transação poderá acessar o item até que a transação atual o desbloqueie, o colocando em stand-by. Esse processo de stand-by coloca a segunda transação em uma fila de espera pelo item, até que o mesmo seja desbloqueado.

Para que a técnica de bloqueio binário possa ser usada, uma transação T deve obedecer às seguintes regras:

- T deve emitir um lock_item(X) antes que qualquer read_item(X) ou write_item(X) seja executado;
- T deve emitir um unlock_item(X) depois que todos os read_item(X) e write_item(X) tenham sido completados em T;
- T não poderá emitir lock_item(X) se X estiver bloqueado por T;
- T poderá emitir um unlock_item(X) apenas se tiver bloqueado X.

O bloqueio binário é o mecanismo mais simples e mais restrito de controle de concorrência, sendo que sua implementação requer uma tabela de bloqueios e uma fila de espera.

Bloqueio Múltiplo

Um esquema de bloqueio múltiplo (read/write ou compartilhado/exclusivo) permite que um item de dado seja acessado por mais de uma transação para leitura. Já as operações necessárias são a read_lock(X) que bloqueia o item X para leitura, permitindo que outras transações leiam o item X (bloqueio compartilhado); a write_lock(X) que bloqueia o item X para gravação, mantendo o bloqueio sobre o item X (bloqueio exclusivo) e a unlock(X) que desbloqueia o item X.

A implementação do bloqueio múltiplo requer uma tabela de bloqueios e uma fila de espera. Para que possamos usar o bloqueio múltiplo, devemos seguir algumas regras:

- T deve emitir um read_lock(X) ou write_lock(X) antes que qualquer read_item(X) seja executado em T;
- T deve emitir um write_lock(X) antes que qualquer write_item(X) seja executado em T;
- T deve emitir um unlock(X) depois que todos os read_item(X) e write_item(X) tenham sido executados em T;
- T não emitirá nenhum read_lock(X) ou write_lock(X) se X já estiver bloqueado por T (de forma compartilhada ou exclusiva);
- T poderá emitir um unlock(X) apenas se tiver bloqueado X (de forma compartilhada ou exclusiva).

Bloqueio em Duas Fases

Para garantir escalonamentos serializáveis, as operações de bloqueio e desbloqueio nas transações devem seguir protocolos. O protocolo mais usado é o protocolo de bloqueio em duas fases (Two-Phase Locking). Todas as operações de bloqueio (read_lock e write_lock) precedem a primeira operação de desbloqueio (unlock). As transações são divididas em duas fases:

- expansão: quando são emitidos todos os bloqueios;
- contração: quando os desbloqueios são emitidos e nenhum novo bloqueio pode ser emitido.