

resultados. Duas abordagens são usadas no processamento de consultas, uma utiliza heurísticas e outra faz estimativa de custo. Usualmente, estas duas abordagens combinadas na otimização de uma consulta. Portanto a otimização envolve:

9

SISTEMAS GERENCIADORES DE BANCOS DE DADOS

Os Sistemas Gerenciadores de Bancos de Dados (SGBD) são responsáveis por uma série de tarefas, que, antes de sua criação, ficavam sob a responsabilidade do programador. A ideia principal é a de garantir uma série de vantagens na criação e manutenção dos BD, utilizando um SGBD, de forma transparente ao programador. Neste capítulo, algumas das principais utilidades dos SGBD serão apresentadas e a maior preocupação é a de apresentar conceitos importantes na área, sem levar em conta um ou outro produto. O foco será nas técnicas mais utilizadas atualmente.

9.1. PROCESSAMENTO DE CONSULTAS

Diferentes técnicas podem ser usadas para processar, otimizar e executar consultas de alto nível, como as consultas feitas usando comandos em SQL. Uma consulta deve ser examinada (scanner), analisada (parser) e validada.

O scanner identifica os componentes da linguagem (*tokens*) no texto da consulta, realizando uma análise léxica, enquanto o parser realiza a análise sintática da consulta para determinar se esta foi formulada seguindo as regras de sintaxe da linguagem. A consulta também será validada, através da checagem de que todos os nomes de atributos e tabelas são válidos e semanticamente significativos. Em outras palavras, verifica e valida se uma consulta foi formulada corretamente para determinado BD, utilizando nomes coerentes. Uma representação interna da consulta é criada, geralmente usando uma estrutura de árvores ou grafos, chamada de árvore de consulta ou grafo de consulta.

O SGBD deve então determinar uma estratégia de execução para recuperar os resultados da consulta nos arquivos internos do banco de dados. Nesta etapa, diferente de como era feito no passado, o programador não precisa se preocupar com a exata localização física dos dados. É o SGBD que faz isso.

Uma consulta tipicamente possui várias possíveis estratégias de execução, e o processo para escolher dentre elas uma melhor é conhecido como otimização de

Execute as operações de seleção e projeção o mais cedo possível – com isto é possível diminuir os tamanhos das tabelas intermediárias que poderiam ser geradas para acessar os dados desejados;

Execute outras transformações que permitam utilizar a regra acima ou reduzam o tamanho das relações intermediárias – algumas transformações, como a comutatividade de junções, a associatividade de junções etc, podem ser aplicadas para alcançar os mesmos objetivos da regra anterior.

A Figura 9.1 ilustra os passos que serão realizados no processamento de consultas de alto nível.

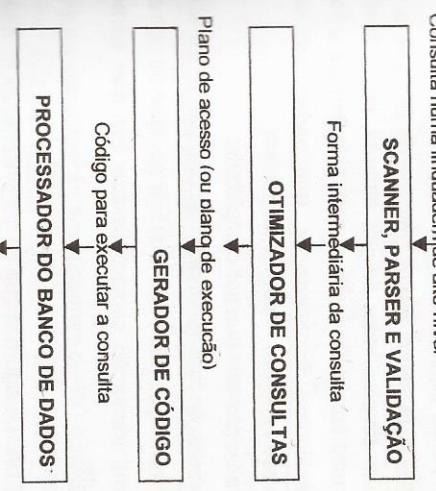


Figura 9.1: Passos do processamento de consulta.

9.2. PROCESSAMENTO DE TRANSAÇÕES

Uma transação de BD é a execução de um programa que inclui operações de acesso ao BD, por exemplo, executando comandos de leitura (*read*) ou escrita dos dados (*write*). As operações de *read*, quando os dados são recuperados sem que sejam realizadas alterações nos mesmos, e *write*, quando os dados são modificados, são as principais tarefas realizadas num BD.

Um exemplo de uma transação realizada num BD poderia ser representado da seguinte forma, supondo uma transação de transferência bancária:

```
read_item(X);
X := X - N;
write_item(X);
read_item(Y);
Y := Y + N;
write_item(Y);
```

$$\begin{array}{rcl} & \text{100} & \\ & \boxed{45} & \\ & 100 - 45 & \\ & 55 & \\ & \text{24} & \\ & \boxed{25} & \\ & 24 + 25 & \\ & 49 & \\ & \boxed{34} & \\ & 49 - 34 & \\ & 15 & \end{array}$$

Onde estão sendo realizadas as seguintes tarefas: lê um item "X" do BD (que no exemplo é o saldo da conta de onde será retirado o dinheiro) e o coloca na memória principal, realiza uma retirada do valor "N" (valor que será transferido), grava o novo valor de "X" no BD (agora subtraído da quantia "N"), lê um item "Y" do BD (que é o saldo da conta que receberá o dinheiro) e o coloca na memória principal, realiza a soma do valor "N" e grava o novo valor de "Y" no BD (agora acrescido da quantia "N").

Não importa o que significa este "item", é uma unidade do BD que, a depender da aplicação pode variar entre um simples valor de um atributo, como também poderia referir-se a todos os dados de uma tabela. Pode-se chamar este item de grão.

As transações possuem algumas propriedades, que podem ser chamadas de Propriedades ACID (devido às suas iniciais), e são as seguintes:

- Atomicidade – A transação tem que ser completada;
- Consistência – A transação tem que manter a consistência do BD;
- Isolamento – Uma transação não pode interferir na ação de outra;
- Durabilidade – Uma transação compromissada tem que garantir que seu efeito no BD perdure.

Uma transação pode estar em um dos seguintes estados:

- BEGIN_TRANSACTION – significa que uma transação foi iniciada;
- READ / WRITE – significa que alguma tarefa de leitura ou escrita está sendo realizada no BD;
- END_TRANSACTION – significa que a transação alcançou sua última operação;

A Figura 9.2 ilustra o diagrama de transição de estados para execução de transações. Uma vez iniciada (BEGIN TRANSACTION) a transação estará ativa e poderá realizar tarefas de leitura e/ou escrita dos dados (READ / WRITE). Ao chegar no final de todas as operações da transação (END TRANSACTION), ela estará parcialmente compromissada, somente depois de compromissada (COMMIT), uma transação terminada com sucesso. Estando ativa ou parcialmente compromissada, a transação deve sofrer algum tipo de falha, o que a levará ao estado ABORT e/ou ROLLBACK, respectivamente sendo terminada sem sucesso.

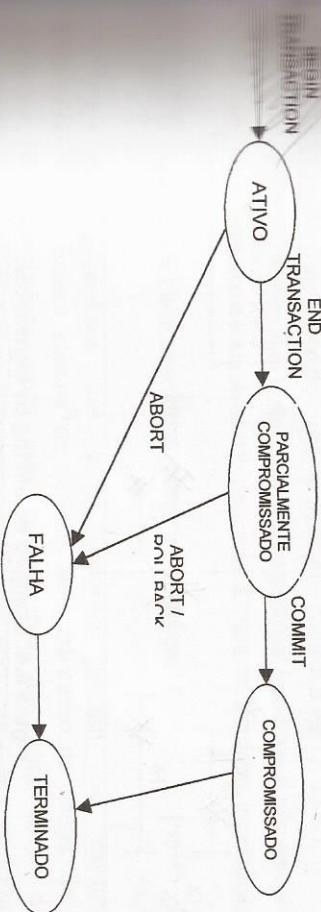


Figura 9.2: Diagrama de transição de estados para execução de transações.

Um critério de classificação dos sistemas de BD é baseado no número de usuários que podem utilizá-lo coincidentemente, ou seja, ao mesmo tempo. Um SGBD é classificado como mono-usuário se no máximo um usuário pode usar o mesmo sistema vez, e é multi-usuário se muitos usuários podem usar o mesmo sistema independentemente.

A Figura 9.3 apresenta de forma gráfica como seria a execução simultânea de duas transações, num sistema multi-usuário. Geralmente, a execução coincidente de

transações é realizada de forma entrelaçada (transações A e B) e não de forma simultânea (transações C e D), embora isto ser transparente para o programador, que poderia pensar que as transações são simultâneas.

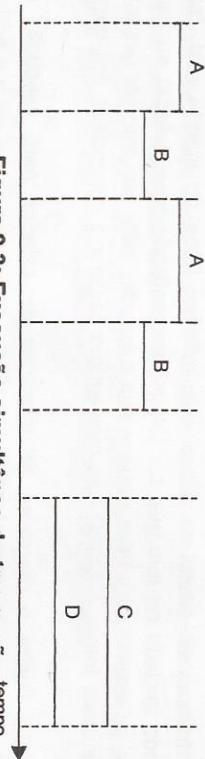


Figura 9.3: Execução simultânea de transações

O log de operações grava os resultados das operações, marcando todas as operações que foram executadas, usando pontos de sincronismo ou checkpoints. A Figura 9.4 ilustra o exemplo de um log de transações sendo executadas com o passar do tempo.

A execução de uma operação de write inclui os seguintes passos:

- 1) Encontre o endereço do bloco no disco que contém o item X do BD.
 - 2) Copie este bloco num buffer da memória principal (caso ele já não esteja lá).
 - 3) Copie o valor da variável de programa X para sua correta localização no buffer.
 - 4) Armazene o bloco alterado do buffer para o disco.

Transações submetidas por vários usuários podem ser executadas concomitantemente e podem acessar os mesmos itens do banco de dados. Se esta execução for descontrolada, pode levar a problemas, tais como tornar o BD inconsistente. Alguns problemas que poderiam ocorrer com transações coincidentes, podem ser exemplificados utilizando as seguintes transações T1 e T2.

No exemplo apresentado na Figura 9.4, no caso de ocorrer uma falha em "F", o log de transações volta até o tempo do *checkpoint* "C", pois até ali toda a execução está garantida. Porém as transações T4 e T6 terão que ser desfeitas e/ou refeitos.

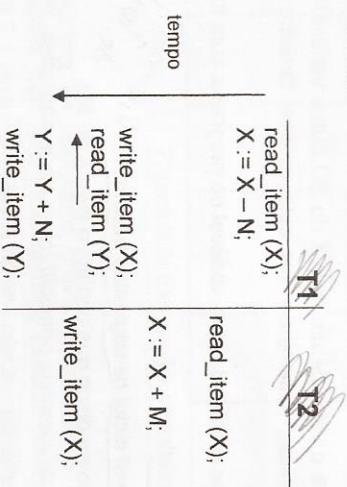
A depender das operações efetuadas sobre o BD, algum dos registros abaixo é gravado no log do sistema:

- **[start_transaction, T]** – informa que a transação T foi iniciada
- **[write_item, T, X, old_value, new_value]** – informa que a transação T modificou o valor do item X do BD, do antigo valor (old_value) para um novo valor (new_value)

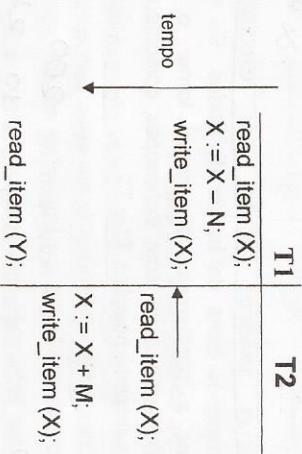
T1 = read_item(X); 500
 X := X - N; 500 - 10 = 490
 write_item(X); 490
 read_item(Y); 13
 Y := Y + N; 13 + 10 = 23
 write_item(Y); 23
(23)
 //
 T2 = read_item(X); 200
 X := X + M; 200 + 40 = 240
 write_item(X); 240
(240)
 //
(240)

Dividindo estas transações e as executando alternadamente, para que ambas possam ser executadas ao mesmo tempo, podemos chegar nos seguintes problemas:

Problema de atualização perdida – ocorre quando transações acessam o mesmo item do BD, com suas operações entrelaçadas de uma forma que faz o valor do item de BD ser incorreto.



Problema de valores gravados temporariamente – ocorre quando uma transação altera o valor de um item de BD, e depois a transação falha por alguma razão. O item alterado pode ser acessado por outra transação antes de retornar ao valor anterior correto.



Quando uma transação é submetida para execução num SGBD, o sistema é responsável por garantir que:

- ou todos as operações na transação serão completadas com sucesso e seu resultado será gravado permanentemente no BD;
 - ou a transação não terá efeito algum no BD ou em outra transação.
- O SGBD não pode permitir que algumas operações de uma transação I sejam

- Tipos de falhas que podem acontecer um sistema de BD:
- Falha no computador (system crash): um erro de hardware ou software, que pode causar a perda do conteúdo na memória.
- Erro de sistema ou de transação: alguma operação na transação pode causar uma falha, tal como, sobrecarga de valor inteiro, ou divisão por zero.
- Erros locais ou detecção de condições de exceção na transação: durante a execução, algumas condições podem ocorrer que necessitem do cancelamento da transação, por exemplo, um dado para uma operação pode não ser encontrado.
- Obrigação de controle de concorrência: o método de controle de concorrência pode decidir abortar a transação, para ser recomeçada depois, por diversos motivos, como por exemplo, deadlock.
- Falha em disco: alguns blocos do disco podem perder seus dados por causa de um defeito no cabeçote de leitura e escrita.
- Problemas físicos e sinistros: uma lista infinita de problemas que podem acontecer, tais como falta de energia, fogo, sabotagem, erros de entrada de dados, etc.

Um conceito importante no processamento de transações é o de escalonamento.

Um escalonamento (*schedule*) S de n transações T_1, T_2, \dots, T_n é uma ordenação das operações dessas transações, tal que, para cada transação T_i , as operações de T_i em S aparecem na mesma ordem em que elas aparecem em (T_i) . Ou, escalonamento é quando se pega um conjunto de transações e define-se a ordem em que elas vão ser executadas, para prover concorrência no BD.

No escalonamento somente as operações de *read* (r) e *write* (w) são importantes. Para um escalonamento S , para T_1 e T_2 , sendo as mesmas transações apresentadas simultaneamente, pode ser representado na forma:

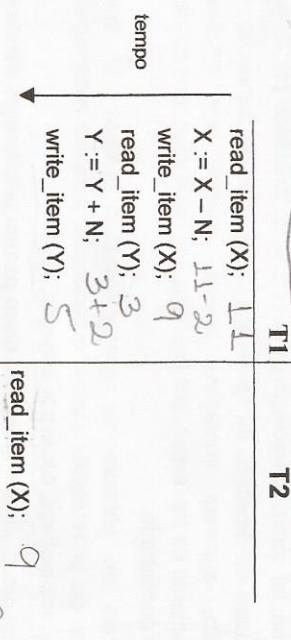
a) r1(X); r2(X); w1(X); w2(X); c2; w1(Y); c1; ou
b) r1(X); r2(X); w1(X); r2(X); w2(X); c2; r1(Y); a1;

Um escalonamento S é serial se, para toda transação T participante de S , todas as operações de T são executadas consecutivamente em S , caso contrário S é não serial.

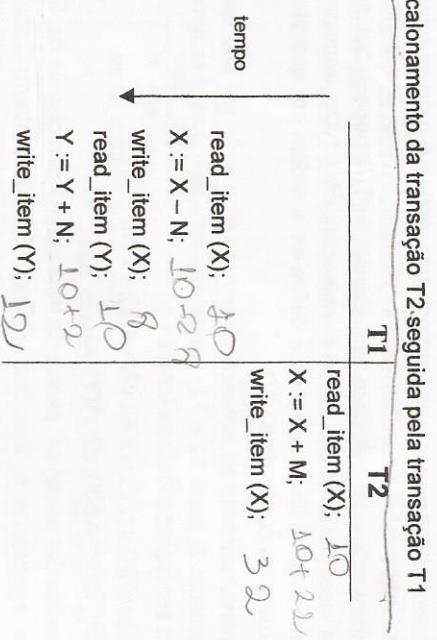
Um escalonamento S é serializável se for equivalente a um escalonamento serial S' .

Exemplos:

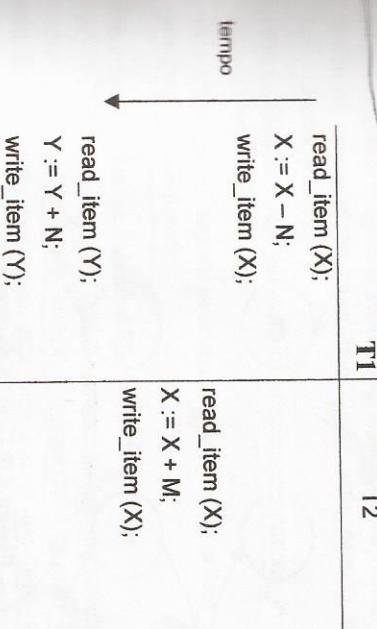
(a) escalonamento da transação T1 seguida pela transação T2



(b) escalonamento da transação T2 seguida pela transação T1



(d) escalonamento com entrelaçamento de operações



Um grafo de precedência de um escalonamento S é um grafo dirigido $G = (N, E)$, onde o conjunto de nodos N corresponde ao conjunto de transações T_1, T_2, \dots, T_n de S e o elemento de arestas E corresponde ao conjunto de arestas $(T_j \rightarrow T_k), 1 \leq j \leq n, 1 \leq k \leq n$, se T_j é chamado de nodo inicial e T_k é chamado de nodo final. O grafo de precedência é construído para testar-se um determinado escalonamento é serializável ou não. Caso seja serializável este é um escalonamento seguro, caso contrário ele é conflitante.

O seguinte algoritmo realiza a construção do grafo de precedência de um escalonamento, para realizar um teste de serialização conflitante:

- 1) para cada transação T_i participante do escalonamento S é criado um nodo chamado T_i no grafo de precedência;

2) para cada caso em S_i , onde T_j executa uma operação de $\text{read_item}(X)$, depois de um $\text{write_item}(X)$ executado por T_i

crie uma aresta $(T_i \rightarrow T_j)$ no grafo de precedência;

3) para cada caso em S_i , onde T_j executa uma operação de $\text{write_item}(X)$, depois de um $\text{read_item}(X)$ executado por T_i

crie uma aresta $(T_i \rightarrow T_j)$ no grafo de precedência;

4) para cada caso em S_i , onde T_j executa uma operação de $\text{write_item}(X)$, depois de um $\text{write_item}(X)$ executado por T_i

crie uma aresta $(T_i \rightarrow T_j)$ no grafo de precedência;

5) o escalonamento S é serializável se e somente se o

grafo de precedência não contém ciclos.

A Figura 9.5 apresenta os grafos de precedência para os exemplos (a), (b), (c) e (d) anteriores:

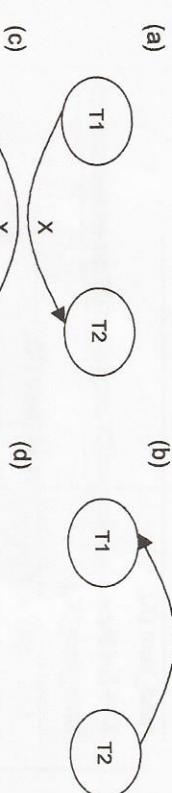


Figura 9.5: Grafos de precedência.

9.3. CONTROLE DE CONCORRÊNCIA

Nesta seção, serão discutidas técnicas de controle de concorrência que são usadas para assegurar isolamento (ou não interferência) durante a execução de transações coincidentes. As principais técnicas são as seguintes:

- Bloqueio (pessimista) ✕
 - Ordenação por marcadores de tempo ✕
 - Uso de múltiplas versões ✕
 - Validação (ótimista) ✕
- As técnicas de bloqueio e ordenação por marcadores de tempo usam o conceito de serialização, porém somente a primeira é implementada devido à complexidade da

única. As técnicas do uso de múltiplas versões e de validação também não vêm sendo implementadas. Sendo assim, somente a técnica de bloqueio é utilizada e será estudada. A principal idéia da técnica de bloqueios é bastante simples, se uma transação vai determinado item do BD, seja para leitura ou para escrita, esta transação bloqueia somente para as demais transações que estiverem sendo executadas coincidentemente. A utilização de bloqueios pode ser implementada de duas formas, utilizando bloqueio binário ou utilizando bloqueios múltiplos.

Bloqueios binários

Nesta técnica, somente dois estados são permitidos: bloqueado ou desbloqueado, e suas operações devem ser incluídas nas transações: $\text{lock_item}(X)$ e $\text{unlock_item}(X)$.

- 1) Quando o bloqueio binário é usado a transação deve obedecer às seguintes regras:
- 2) Uma transação T deve executar uma operação de $\text{lock_item}(X)$ antes de qualquer operação de $\text{read_item}(X)$ ou $\text{write_item}(X)$ ser executada em T .

- 3) Uma transação T deve executar uma operação de $\text{unlock_item}(X)$ depois de qualquer operação de $\text{read_item}(X)$ ou $\text{write_item}(X)$ ter sido executada completamente em T .
- 4) Uma transação T não vai executar a operação de $\text{lock_item}(X)$ se o item X já estiver bloqueado.
- 5) Uma transação T não vai executar a operação de $\text{unlock_item}(X)$ se o item X já estiver desbloqueado.

Bloqueios compartilhados e exclusivos

Os bloqueios são distintos para leitura ou escrita (modo múltiplo). Três operações são necessárias: $\text{read_lock}(X)$, $\text{write_lock}(X)$ e $\text{unlock}(X)$. Um read_lock é um bloqueio compartilhado, quando uma transação está bloqueando um item do BD para leitura, outras transações também podem ler este mesmo item; e um write_lock é um bloqueio exclusivo, quando uma transação está bloqueando um item para escrita, outras transações não podem ler ou escrever neste item.

Quando o bloqueio de modo múltiplo é usado o sistema deve seguir as seguintes regras:

- 1) Uma transação T deve executar uma operação de $\text{read_lock}(X)$ ou $\text{write_lock}(X)$ antes de qualquer operação de $\text{read_item}(X)$ ser executada em T .
- 2) Uma transação T deve executar uma operação de $\text{write_lock}(X)$ antes de qualquer operação de $\text{write_item}(X)$ ser executada em T .
- 3) Uma transação T deve executar uma operação de $\text{unlock}(X)$ depois de qualquer operação de $\text{read_item}(X)$ e $\text{write_item}(X)$ ter sido executada completamente em T .

Christian
Morten
Nunes

Sabatéja típica de recuperação de falhas é a seguinte:

- 4) Uma transação T não vai executar a operação de $read_lock(X)$ se o item X já estiver bloqueado compartilhado para um $read$ ou bloqueado exclusivo para um $write$.
- 5) Uma transação T não vai executar a operação de $write_lock(X)$ se o item X já estiver bloqueado compartilhado para um $read$ ou bloqueado exclusivo para um $write$.
- 6) Uma transação T não vai executar a operação de $unlock_item(X)$ se o item X já estiver desbloqueado.

Algumas vezes é possível fugir às regras 4 e 5. Por exemplo, se uma transação T estiver bloqueando um item X no modo compartilhado ($read$) e for preciso fazer um bloqueio maior exclusivo ($write$), neste mesmo item X .

Apenas o uso de bloqueios (binário ou múltiplo) não garante a serializabilidade dos escalonamentos. São necessários protocolos que determinem a ordem em que os bloqueios e desbloqueios devem aparecer nas transações.

Uma transação segue o protocolo de bloqueio em duas fases (2PL – Two Phase Lock) se todas as operações de bloqueio precedem a primeira operação de desbloqueio na transação. Portanto, de acordo com o protocolo 2PL, uma transação possui uma fase de expansão (durante a qual novos bloqueios podem ser feitos, mas não desfeitos) e uma fase de encolhimento (durante a qual bloqueios existentes podem ser desfeitos, mas nenhum novo pode ser feito).

Pode-se provar que, se toda transação em um escalonamento segue o protocolo 2PL, então a transação é serializável. Embora o protocolo 2PL garanta a serializabilidade, bloqueios podem gerar dois problemas: impasse (deadlock) e inanição (*livelock*).

Deadlock ocorre quando, existem duas transações bloqueando o mesmo item e cada uma das duas está esperando que a outra libere este bloqueio. A solução mais comum para este problema de impasse é abortar uma das transações envolvidas. Existem diversas implementações diferentes para decidir qual transação deve ser interrompida no caso de um impasse, geralmente é escolhida a transação mais nova. Isto pode causar o *livelock* (quando o sistema não consegue decidir qual das transações deve ser abortada), então é introduzida uma prioridade para as transações (por exemplo, maior prioridade para transações que já foram abortadas anteriormente).

9.4. RECUPERAÇÃO DE FALHAS

Já foi explicada a necessidade dos SGBD implementarem a recuperação de falhas. Estas recuperações nem sempre são causadas por falhas físicas, como por exemplo, falhas em disco, muitas vezes as falhas são causadas por transações abortadas e mal recuperadas.

Para a segunda estratégia duas das principais técnicas são as seguintes:

Técnica de atualização adiada (Algoritmo NO UNDO / REDO)

A atualização é feita no BD no momento exato em que foi executada, então tem que fazer tudo o que for necessário para restaurar o BD. Esta técnica não precisa desfazer nada, só refazer o que já tinha comprometido antes.

Técnica de atualização imediata (Algoritmo UNDO / REDO)

A atualização é feita no BD no momento exato em que foi executada, então tem que fazer tudo o que for necessário para restaurar o BD. Esta técnica é muito pouco utilizada, sendo assim, mencionamos somente a técnica de atualização adiada.

Técnicas de recuperação baseadas em atualização adiada:

A ideia básica é adiar a execução de qualquer atualização sobre o BD até que a transação seja concluída com sucesso e atinja o seu ponto de comprometimento.

Durante a execução de uma transação, as operações são apenas registradas no log da área de trabalho da transação. Depois que a transação atinge seu ponto de comprometimento e o log é escrito em disco, as atualizações são registradas no BD.

O protocolo típico a ser seguido é o seguinte:

- 1) Uma transação não pode alterar o BD até que atinja o seu ponto de comprometimento.
- 2) Uma transação não atinge o seu ponto de comprometimento até que suas operações de atualização estejam gravadas no log e o log esteja escrito em disco.

Protocolo básico é:

Falha física
Copy 09

Execute a operação REDO para todas as operações WRITE das transações já comprometidas na ordem em que elas aparecem no log.

Reinicie as transações ativas.

As operações de REDO funcionam da seguinte maneira:

Para cada entrada [write_item, T, X, new_value] no log, substitui o valor de X no BD por new_value.

Transações ativas não têm efeito sobre o BD e não são totalmente ignoradas durante o processo de recuperação, é o chamado rollback implícito.

Em um ambiente multi-usuário a técnica depende do método de controle de concorrência adotado. Geralmente, quanto maior o grau de concorrência, mais difícil é a tarefa de recuperação.

Exemplo:

Sejam T1, T2 e T3 três transações, tais que:

T1:	T2:	T3:
X read_item(A)	X read_item(B)	X read_item(C)
X read_item(D)	X write_item(B)	X write_item(C)
X write_item(D)	+ read_item(D)	X write_item(B)
X write_item(D)	+ write_item(D)	read_item(A)
X write_item(D)	+ write_item(A)	write_item(A)

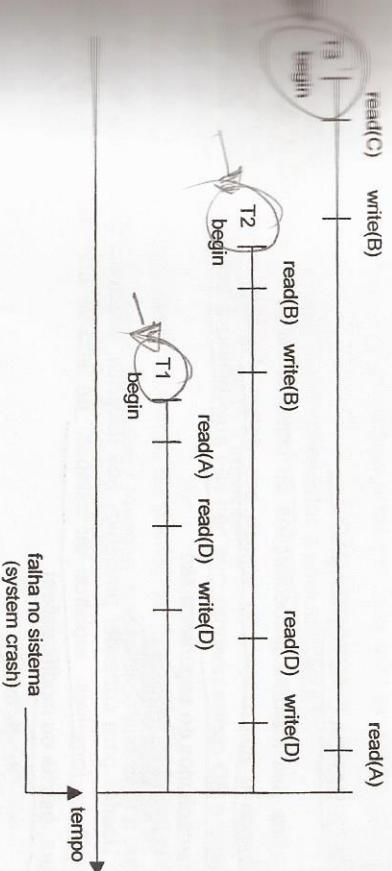
Supondo que o ocorreu uma falha e o log estava sendo escrito da seguinte forma:

A	B	C	D
30	15	40	20

```
[start_transaction, T3]
[read_item, T3, C]
[write_item, T3, B, 15, 12]
[write_item, T3, B, 15, 12]
[start_transaction, T2]
[read_item, T2, B]
[write_item, T2, B, 12, 18]
[start_transaction, T1]
[read_item, T1, A]
[read_item, T1, D]
[write_item, T1, D, 20, 25]
[end_transaction, T1]
[read_item, T2, D]
[write_item, T2, D, 25, 26]
[end_transaction, T2]
[read_item, T3, A]
```

← falha no sistema (system crash)

Figura 9.6: Exemplo de operações no log.



- Observações:
- A transação T3 tem que ser desfeita (rollback), porque ela não atingiu o ponto de comprometimento (end_transaction).
 - A transação T2 tem que ser desfeita (rollback), porque ela lê o valor do item B escrito por T3.
 - As outras operações de write no log serão somente refeitas (REDO).

Figura 9.6 representa as operações no log, antes da falha:

9.5. SEGURANÇA E AUTORIZAÇÃO

Uma das maiores preocupações de implementação de um SGBD diz respeito à segurança e autorização do acesso aos dados. Algumas técnicas são usadas para proteger o BD contra pessoas que não são autorizadas a acessar parte ou todo um BD.

Os mecanismos de segurança são:

- Discretionários
 - Usados para conceder privilégios aos usuários, incluindo direito de acesso a itens específicos (arquivos, registros ou campos) de acordo com um modo especificado (leitura, escrita ou modificações).
 - Obrigatórios

Usados para garantir múltiplos níveis de segurança através da classificação dos dados e usuários em várias classes (ou níveis), e posterior implementação de uma política de segurança.

Outros aspectos de segurança envolvidos na tecnologia de BD são:

- Controle de acesso ao BD – o mecanismo de segurança de um SGBD deve incluir opções para restringir o acesso para o BD como um todo.
- Segurança nos BD estatísticos – alguns arquivos internos do SGBD, usados para gerar informações ou resumos estatísticos sobre os BD não podem ser acessados livremente.
- Criptografia – é uma técnica de segurança geralmente utilizada em algum tipo de comunicação de dados, onde estes são codificados no envio e decodificados no recebimento.

O DBA é a autoridade central que administra um SGBD. É o responsável por conceder privilégios de acesso e classificar dados e usuários de acordo com políticas específicas.

O controle de acesso discricionário baseado em privilégios é o método típico de controle de acesso implementado em um SGBD. Geralmente implementado através de comandos específicos existentes na linguagem adotada pelo SGBD.

Em um SGBD relacional, em geral existem dois níveis de atribuições de privilégios:

- 1) Nível de conta – aplica-se a cada conta independente do BD considerado.
- 2) Nível de relação – aplica-se individualmente às relações (ou visões) de um BD, definidas através de uma linguagem, por exemplo, SQL.

Para controlar a concessão e revogação de privilégios, a cada relação é atribuída uma conta proprietária, que geralmente é a conta através da qual a relação foi criada.

Ao proprietário de uma relação são dados todos os privilégios sobre ela.



Baixas de privilégios que podem ser concedidos a uma relação são:
Selecção (consulta);
Modificação (alteração, remoção e inserção); e
Referência (RI).

O controle de concessão e revogação de privilégios, pode ser criado um grafo de autorização que define que usuário concedeu / revogou determinado privilégio a outro usuário. A Figura 9.7 é um exemplo de grafo de autorização.

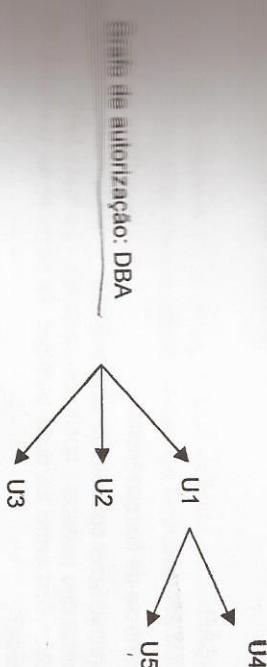
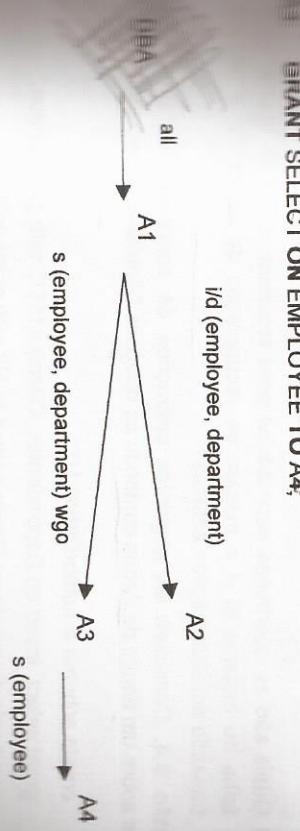


Figura 9.7: Grafo de autorização.

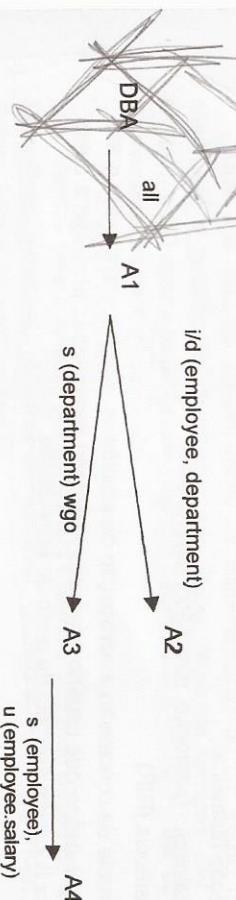
Os seguintes exemplos servem para ilustrar comandos de concessão e revogação de privilégios:

```

CREATE SCHEMA EXAMPLE AUTHORIZATION A1;
GRANT INSERT, DELETE ON EMPLOYEE, DEPARTMENT TO A2;
GRANT SELECT ON EMPLOYEE, DEPARTMENT TO A3
WITH GRANT OPTION;
GRANT SELECT ON EMPLOYEE TO A4;
  
```



A1: REVOKE SELECT ON EMPLOYEE FROM A3;
 A3: GRANT UPDATE ON EMPLOYEE (SALARY) TO A4;



9.6. EXERCÍCIOS DE FIXAÇÃO

Questão 9.1. Sobre controle de concorrência:

a) Explique porque é preciso fazer controle de concorrência.

b) Explique dois problemas que poderiam acontecer caso isto não fosse feito.

c) Qual é a principal diferença entre as duas técnicas de controle de concorrência baseadas em bloqueios?

Questão 9.2. Para que serve a cláusula GRANT OPTION do SQL?

Questão 9.3. Sejam as transações T1, T2, T3, T4 e T5 executadas sobre um BD entre os intervalos de tempo [t1,t4], [t3,t6], [t2,t7], [t4,t8] e [t4,t6], respectivamente, e tais que $t_i < t_j$ para $i < j$.

a) Mostre qual seria o conteúdo do arquivo de log ao final da execução dessas cinco transações. Considere que apenas os registros [start_transaction, Ti] e [end_transaction, Tj] são gravados nesse arquivo.

b) Quais são as operações necessárias para recuperar o BD se ocorrer uma falha no instante t5 e o método de recuperação de falhas for o método baseado em atualizações adiadas?

Questão 9.4. Considere as seguintes operações de concessão de privilégios executadas sobre um banco de dados contendo as relações Clientes e Dependentes:

DBA: create schema S authorization U1;
 U1: grant select, insert on Dependentes, Clientes to U2 with grant option;
 U1: grant select, update on Dependentes to U3 with grant option;
 U2: grant select on Dependentes to U4;

grant update on Dependentes(idade) to U4 with grant option;
 grant update on Dependentes to U5;
 grant select on Dependentes to U5;
 grant update on Dependentes to U6 with grant option;
 grant select on Dependentes to U6,
 grant update on Dependentes, Clientes to U5;

b) Considerando a situação final definida no item (a), qual seria o efeito do comando revoke select, update on Dependentes from U3, se executado pelo usuário U1?

c) Considerando a situação final definida no item (a), qual seria o efeito das operações que não foram executadas com sucesso e qual a situação final de cada usuário em relação às operações que pode executar sobre o banco de dados.

d) Considerando a situação final definida no item (a), qual seria o efeito do comando revoke select, update on Dependentes from U3, se executado pelo usuário U1?