UNIVERSIDADE FEDERAL DE ITAJUBÁ

Banco de Dados II

COM 231

Transações

Vanessa Cristina Oliveira de Souza

Transação

"Transação é uma unidade lógica de trabalho, envolvendo diversas operações de bancos dados."

(C. J. Date - Introdução a Sistemas de Bancos de Dados

Transação

- Está envolvida com os comandos **Update, Delete e Insert**.
  - Leituras não comprometem a consistência dos dados no banco
  - Atualizações sim!!!

- Do ponto de vista do SGBD, uma transação é uma sequência de operações que são tratadas como um bloco único e indivisível (atômico) no que se refere à sua recuperação.

- Assegura que atualizações inacabadas ou atividades corrompidas não sejam executadas no banco de dados.

Transação ACID

- A integridade de uma transação depende de 4 propriedades, conhecidas como ACID:
  - Atomicidade
  - Consistência
  - Isolamento
  - Durabilidade

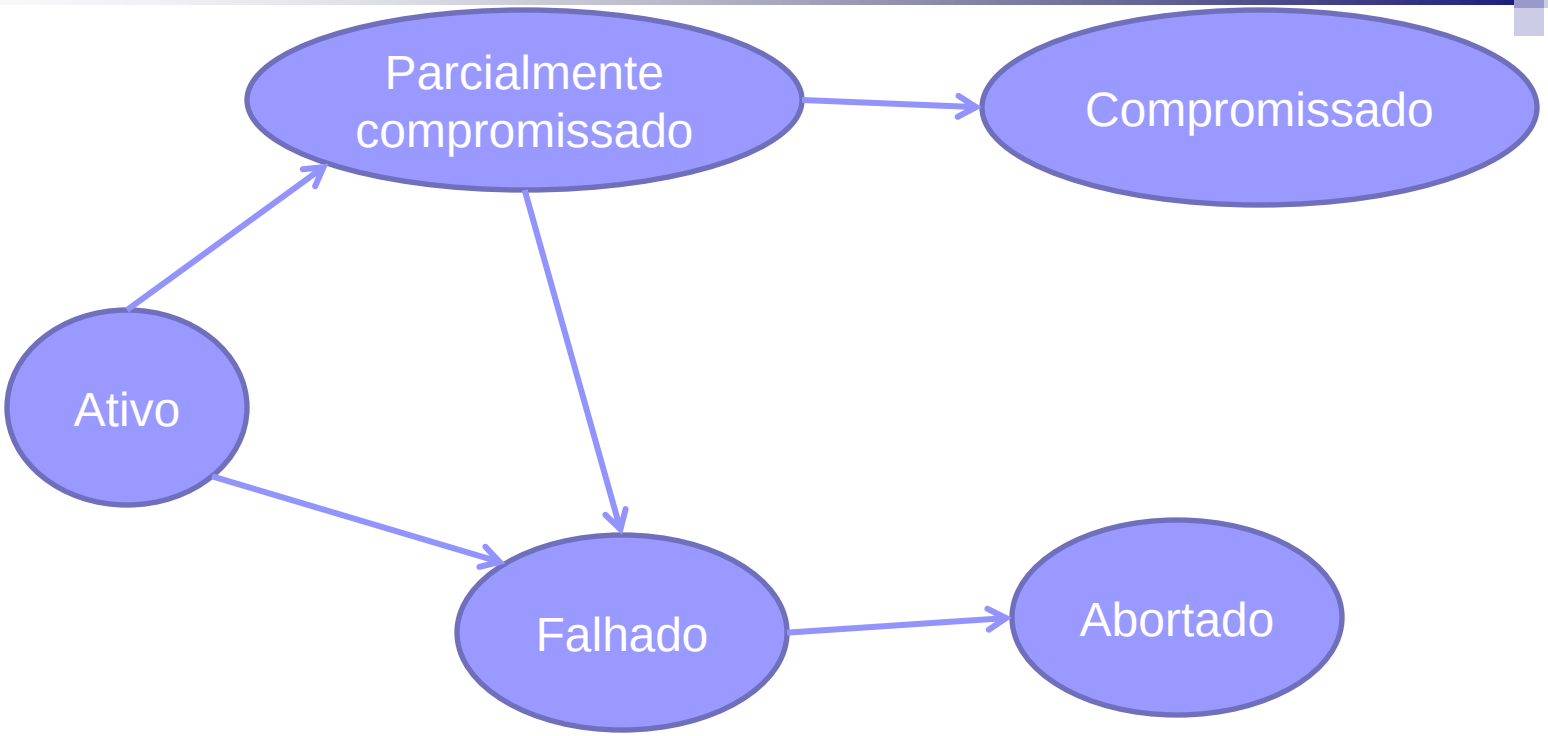
Transação ACID

- *UPDATE funcionarios SET salario = (salario * 1.3);*
  - 1 comando UPDATE
  - Quantas atualizações?
    - *1 ou ∞ = 1 transação*
  - Se o sistema cair durante a atualização 5.000 de 20.000?
    - ACID

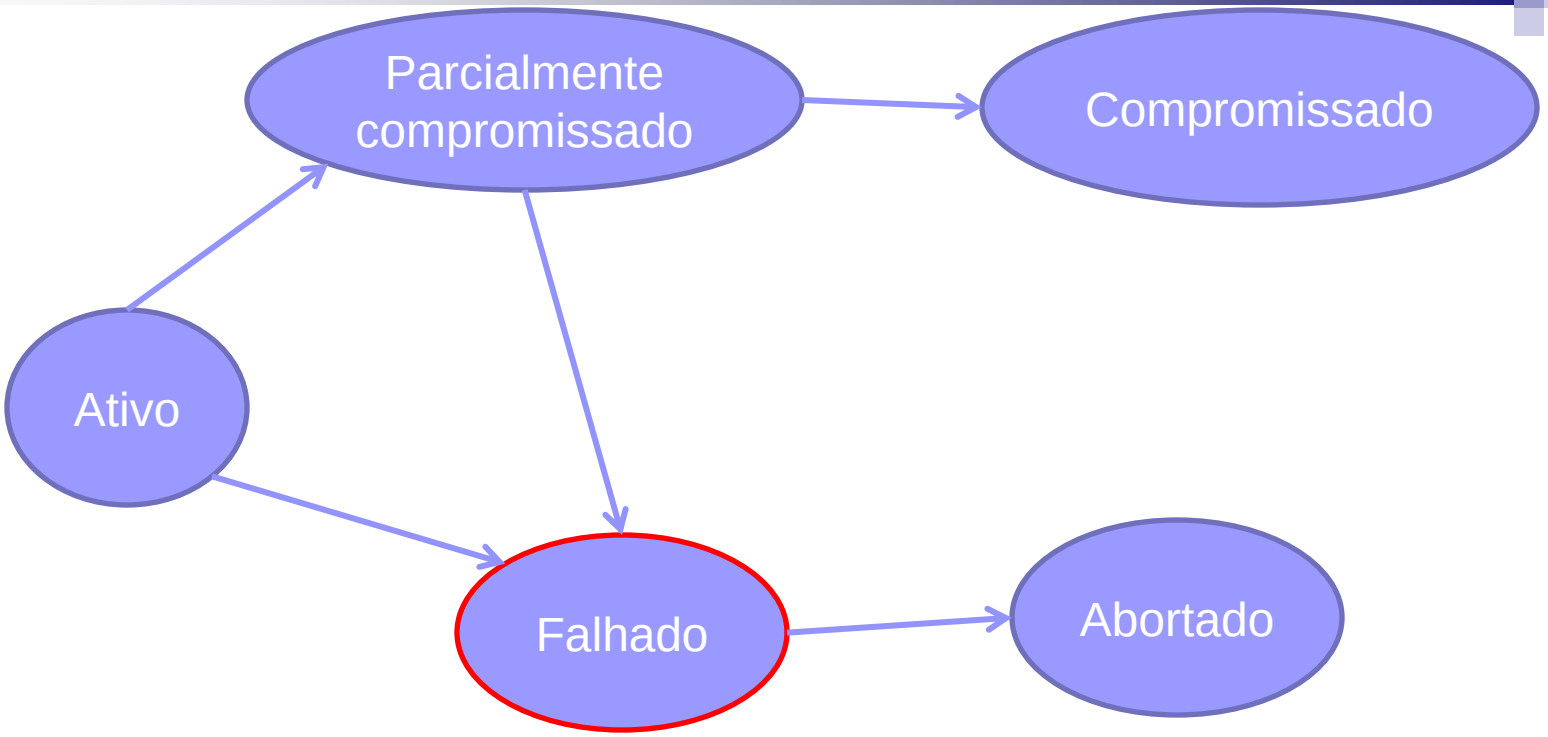
Transação ACID

- Exemplo:
  - A transferência de fundos de uma conta corrente pra uma conta poupança é uma operação única do ponto de vista do cliente, porém, dentro do sistemas de banco de dados, ela envolve várias operações.
    - Quais???

Estados de Transações



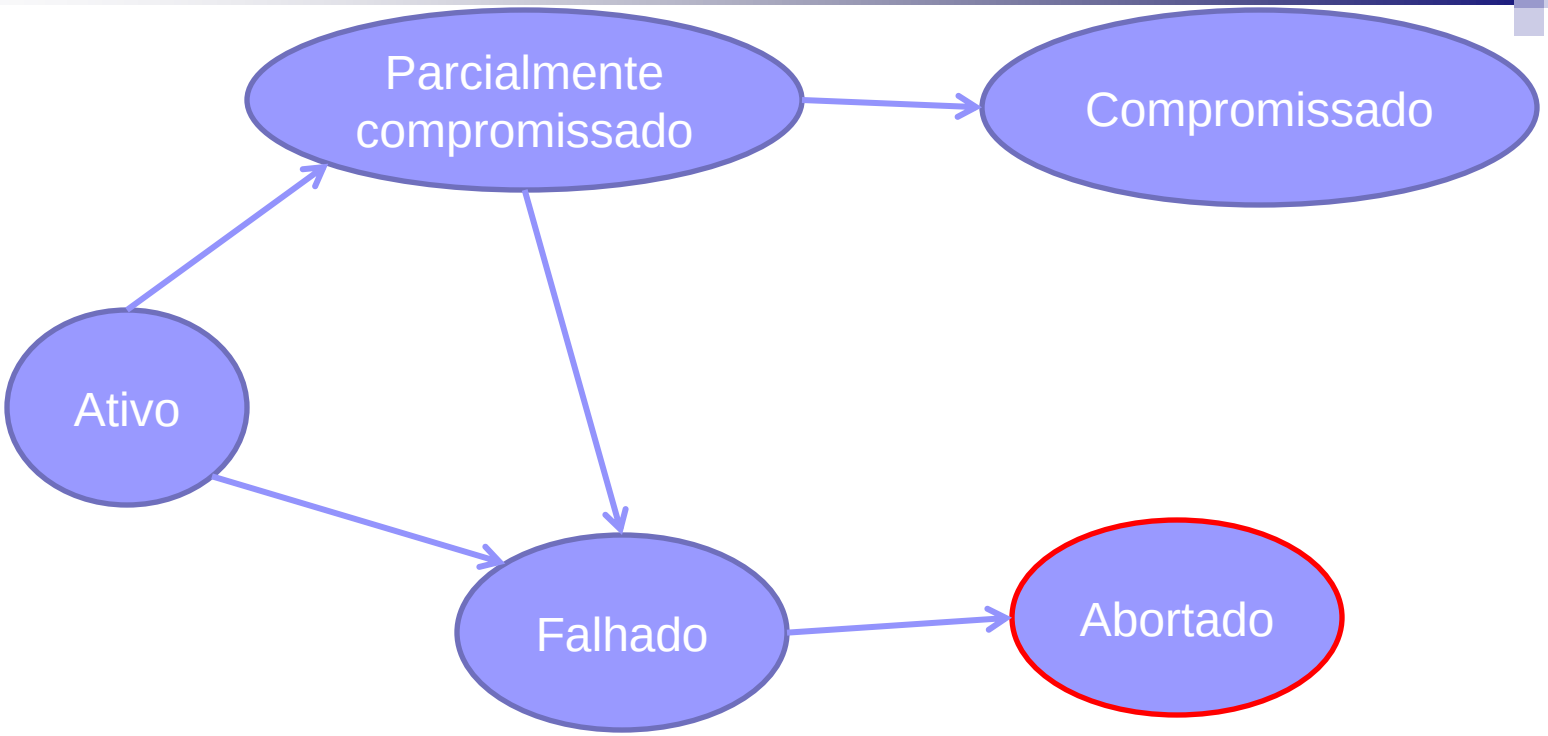
Estados de Transações



Estado Falhado

- Uma transação entra no estado falhado depois de ser determinado que a transação não pode mais prosseguir com sua execução normal (por exemplo, devido a erros de hardware ou a erros lógicos).
- Tal transação precisa ser desfeita.
- Uma vez que isso é conseguido, a transação entra no estado abortado.

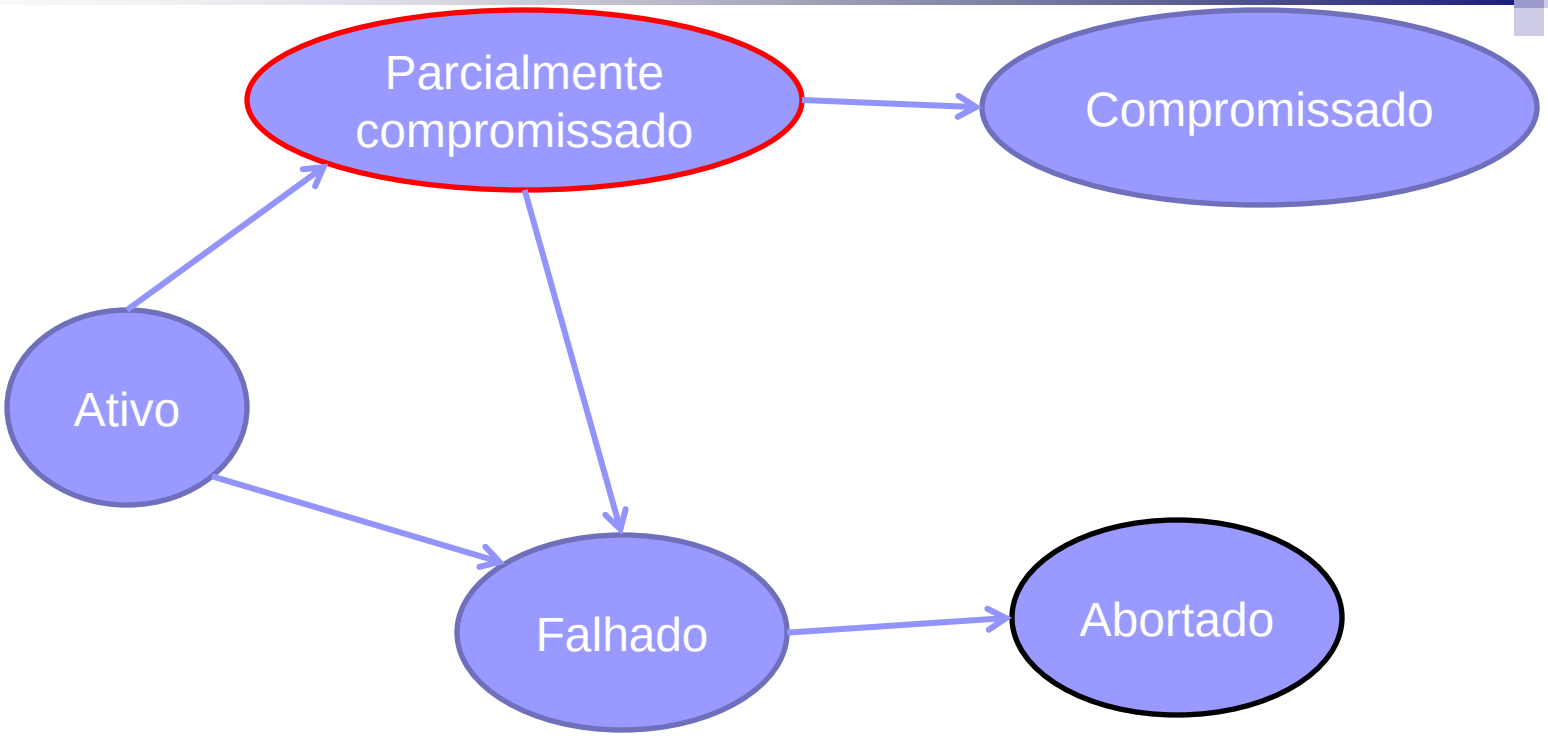
Estados de Transações



Estado Abortado

- Nesse ponto, o sistema tem duas opções:
  - Reiniciar a transação
    - Apenas se a falha foi causada por um erro foi de hardware ou de software
    - Uma transação reiniciada é uma nova transação
  - Matar a transação
    - Normalmente, quando a falha foi causada por um erro de lógica

Estados de Transações

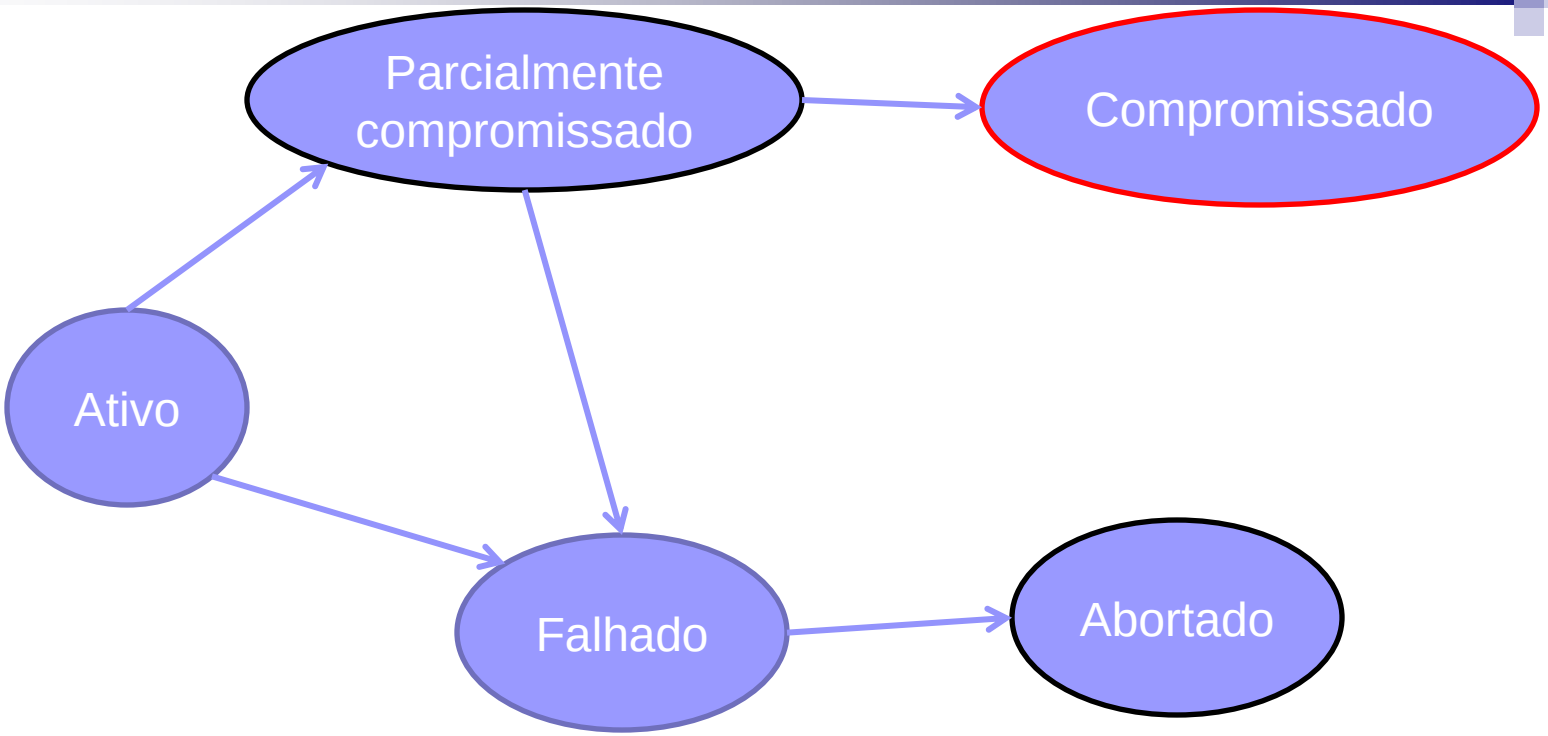


Estado parcialmente compromissado

- Uma transação entra no estado parcialmente compromissado quando alcança sua última instrução.

- Nesse ponto, a transação completou sua execução, mas ainda é possível que ela venha a ser abortada.

Estados de Transações



Estado Compromissado

- ‘Commited’

- Uma transação entra no estado compromissado se já estiver parcialmente compromissada e for garantido que nunca será abortada, garantindo sua atomicidade e durabilidade.

Commit

- Operação de confirmação de que correu tudo bem e que todos os comandos que fazem parte da transação foram executados com sucesso e o banco de dados encontra-se em um estado consistente;
- Salvar a(s) operação(ões) realizadas até o momento;
- As alterações feitas na transação atual tornam-se permanentes e visíveis a outros usuários.

Transações Implícitas

- São aquelas que têm um COMMIT interno e já tornam quaisquer modificações nos dados permanentes no(s) arquivo(s) de dados em disco.

Transações Explícitas

- São aquelas que têm um início e fim explicitado pelo desenvolvedor.

Autocommit

- Por padrão, os SGBDs executam em modo autocommit.
  - Variável de Ambiente
  - No psql -> \echo :AUTOCOMMIT
- Isto significa que assim que você executa uma instrução que modifica uma tabela, a mesma é feita no disco.

Autocommit

- Autocommit = on;
  - as transações implícitas sempre terão um COMMIT interno.

- Autocommit = off;
  - o SGBD esperará por um COMMIT explícito para tornar permanente as últimas manipulações nos dados.

Comportamento das Transações

- Todas as modificações da transação são “temporárias”
- Modificações serão “persistidas” apenas após o ***Commit***
- A qualquer momento antes do *commit* as modificações podem ser canceladas através de um ***Rollback***

Rollback

- Desfaz as operações até o início da transação, caso tenha havido problema com algum comando dentro de uma transação;
  - Quais operações????
- A transação é finalizada sem sucesso.

Autocommit

- \set AUTOCOMMIT off
  - Insert into departamento VALUES (2, “Recursos Humanos”, 1, ‘2007-01-01’);
  - Select * from departamento;
  - Rollback;
  - Select * from departamento;
- \set AUTOCOMMIT on
  - Insert into departamento VALUES (3, “TI”, 6, ‘2008-01-01’);
  - Select * from departamento;
  - Rollback;
  - Select * from departamento;

START TRANSACTION

- Desabilitar o modo autocommit para uma única série de instruções (autocommit = off);
- O autocommit volta a estar habilitado quando se termina a transação;

START TRANSACTION ou BEGIN

- Inicia uma transação multi-instrução;

```
START TRANSACTION;
Comando SQL;
Comando SQL;
Comando SQL;
...
COMMIT;
```

START TRANSACTION ou BEGIN

- Inicia uma transação multi-instrução;
  - START TRANSACTION
  - BEGIN TRANSACTION
  - BEGIN
  - SET TRANSACTION

START TRANSACTION ou BEGIN

START TRANSACTION;

INSERT INTO departamento VALUES (3, “TI”, 6, ‘2008-01-01’);

SELECT * FROM departamento;

UPDATE empregado SET salario = (salario*1.3);

SELECT * FROM empregado;

ROLLBACK;

SELECT * FROM departamento;

SELECT * FROM empregado;

START TRANSACTION ou BEGIN

```
START TRANSACTION;
INSERT INTO departamento VALUES (3, “TI”, 6, ‘2008-01-01’);
SELECT * FROM departamento;
UPDATE empregado SET salario = (salario+(salario*0.3));
SELECT * FROM empregado;
COMMIT;
SELECT * FROM departamento;
SELECT * FROM empregado;
```

SAVEPOINT

- Permite criar "pontos de salvamento" em meio a uma transação, possibilitando que esta seja recuperada até determinado ponto.

- Pode ser feito em transações implícitas e em meio a uma transação explícita, dentro de procedimentos, independente do valor de AUTOCOMMIT.

SAVEPOINT

Bloco Transacional

```
START TRANSACTION;
INSERT INTO cursos VALUES (5, 'EEL', 6, 2);
SAVEPOINT p1;
UPDATE cursos SET coordenador = 1 WHERE codigo=5;
ROLLBACK TO SAVEPOINT p1;
UPDATE cursos SET coordenador = 1 WHERE codigo=5;
COMMIT;
```

Rollback não salva tudo

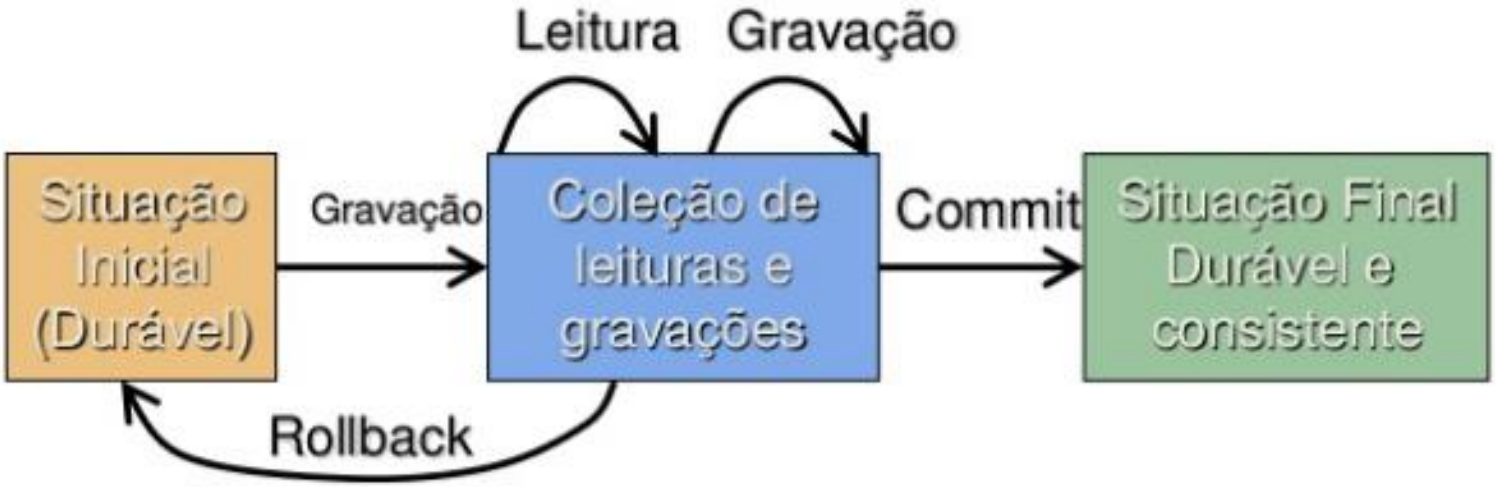
- O rollback não desfaz comandos da DDL, como create table, create database, alter table...

- Esses comandos devem ser evitados dentro de transações

Transação

- As transações interagem com o sistema de banco de dados transferindo dados de variáveis de programa para o banco de dados e, do banco de dados para as variáveis de programa.

Transação


Leitura
Gravação
Situação Inicial (Durável)
Gravação
Coleção de leituras e gravações
Commit
Situação Final Durável e consistente
Rollback

Hierarquia de Armazenamento

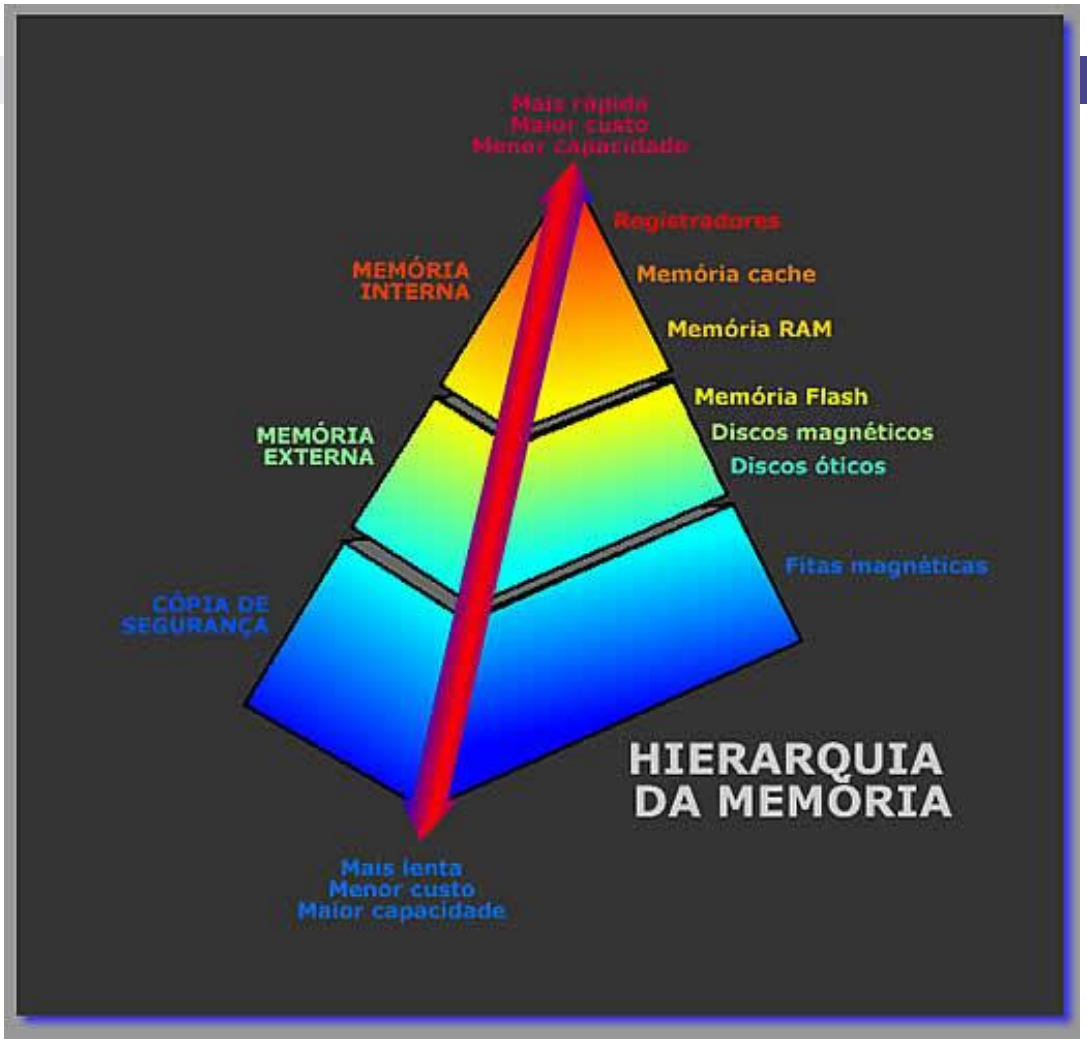
- As transações transferem blocos de informações do disco para a memória principal e depois os devolvem para o disco.

- Os blocos residentes no disco são chamados blocos físicos, e os residentes na memória, de blocos de *buffer*.

Tipos de Armazenamento

- Armazenamento volátil
  - Memória principal e cache
- Armazenamento não-volátil
  - Discos e fitas
- Armazenamento estável
  - Duplicar informações em diversos meios não-voláteis, com modos independentes de falhas, e atualizar a informação de uma maneira controlada.

Hierarquia de Armazenamento



Fonte: http://www.bpiropo.com.br/fpc20070903.htm

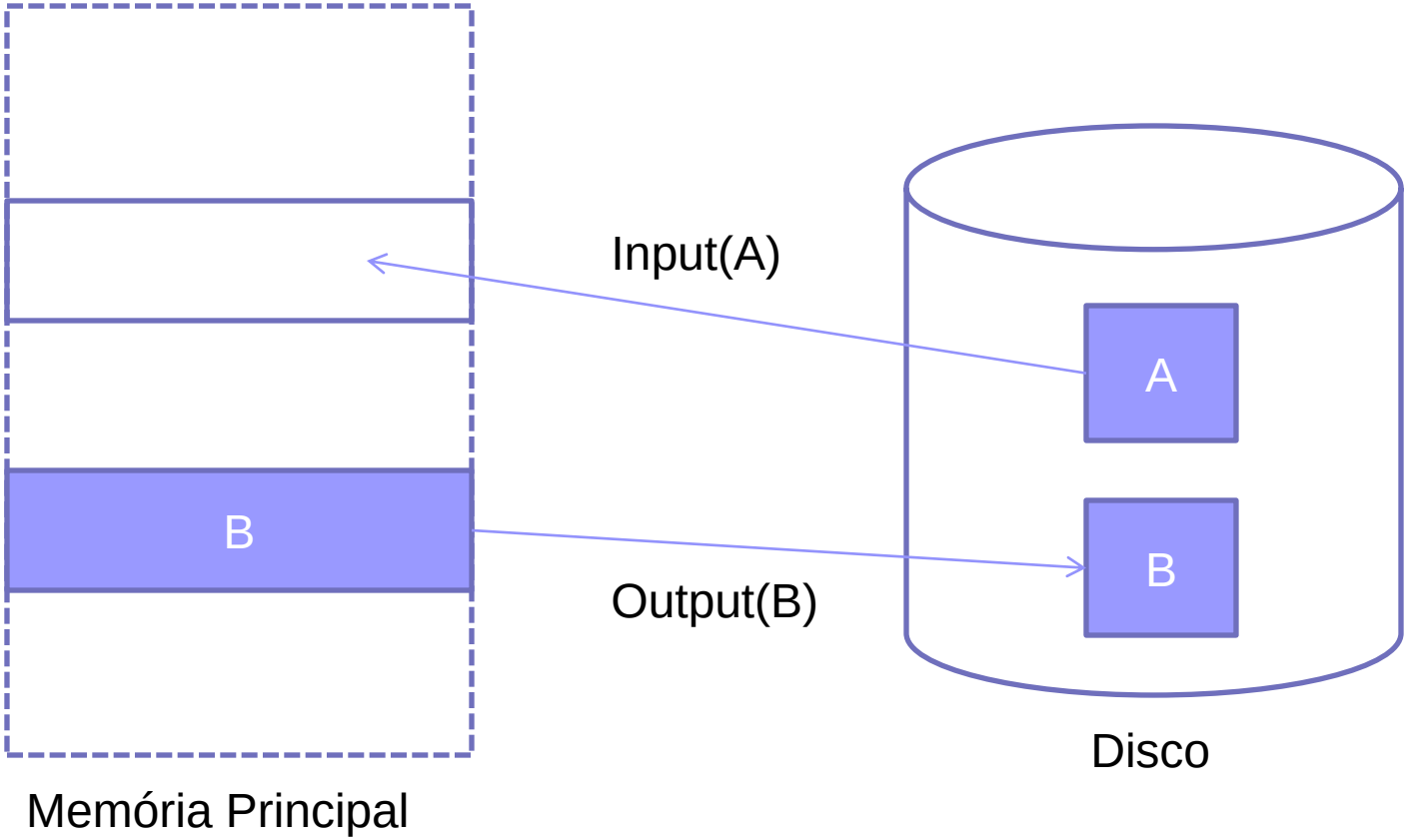
Armazenamento estável

- Informação armazenada em **RAM** é perdida se faltar energia ou se a máquina falhar.
- Informação armazenada em **disco** é perdida se a cabeça do disco falhar.
- Informação em **armazenamento estável** sobrevive a *tudo*, exceto enchentes, terremotos, ...

Hierarquia de Armazenamento

- Os movimentos de blocos entre o disco e a memória são feitos por duas operações:
  - Input(X)
    - Transfere do disco para a memória o bloco físico X
  - Output(X)
    - Transfere da memória para o disco o bloco de buffer X e substitui o bloco físico apropriado.

Hierarquia de Armazenamento



Hierarquia de Armazenamento

- As transações interagem com o sistema de banco de dados transferindo dados de variáveis de programa para o banco de dados (**arquivo!**) e, do banco de dados para as variáveis de programa.

Operações de Leitura e Escrita

- Esta transferência de dados é realizada usando outras duas operações:
  - Read(X, xi)
    - Atribui o valor de X para a variável local xi

      Se o bloco no qual X reside não estiver na memória, então emite input(X)

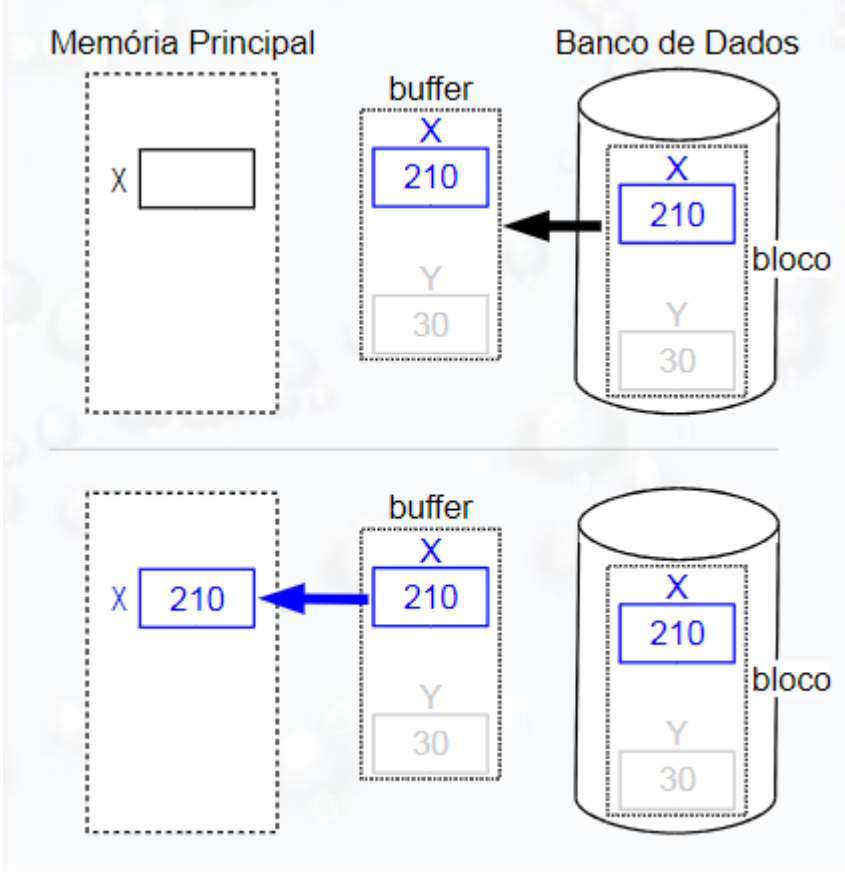
      Atribui o valor do bloco de buffer X para xi
  - Write(X,xi)
    - Atribui o valor da variável local xi para o bloco de buffer X

      Se o bloco no qual X reside não estiver na memória, então emite input(X)

      Atribui xi para o valor de X do bloco de buffer

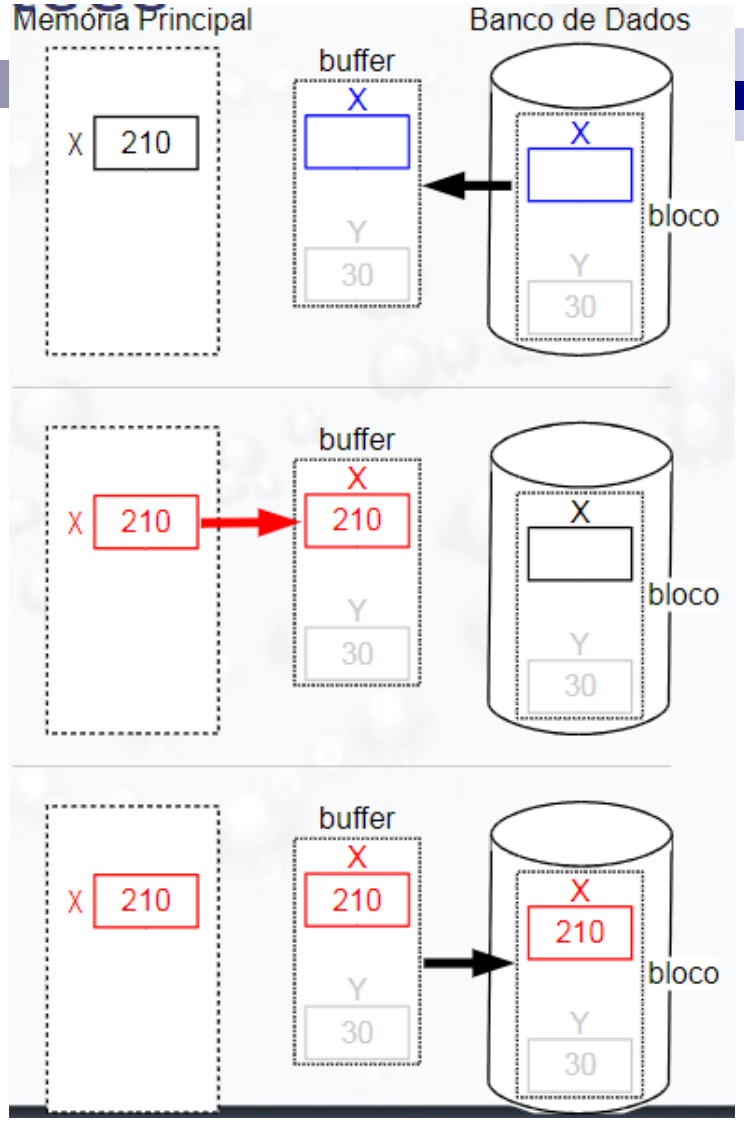
Operações de Leitura e Escrita

- Executar read(x):
  - Encontrar o endereço do bloco de disco que contém x
  - Copiar este bloco de disco para dentro do buffer/memória principal, se ele já não estiver lá;
    - INPUT
  - Copiar o item x do buffer para a variável de programa.



Operações de Leitura e Escrita

- Executar write(x):
  - Encontrar o endereço do bloco de disco que contém x;
    - Para onde x deve ser copiado
  - Copiar o bloco de disco para a memória principal se ele já não estiver lá;
    - INPUT
  - Copiar o item x da variável de programa x para a localização correta no buffer;
  - Copiar o bloco alterado do buffer de volta para o disco (imediatamente ou mais tarde).
    - OUTPUT



Operações de Leitura e Escrita

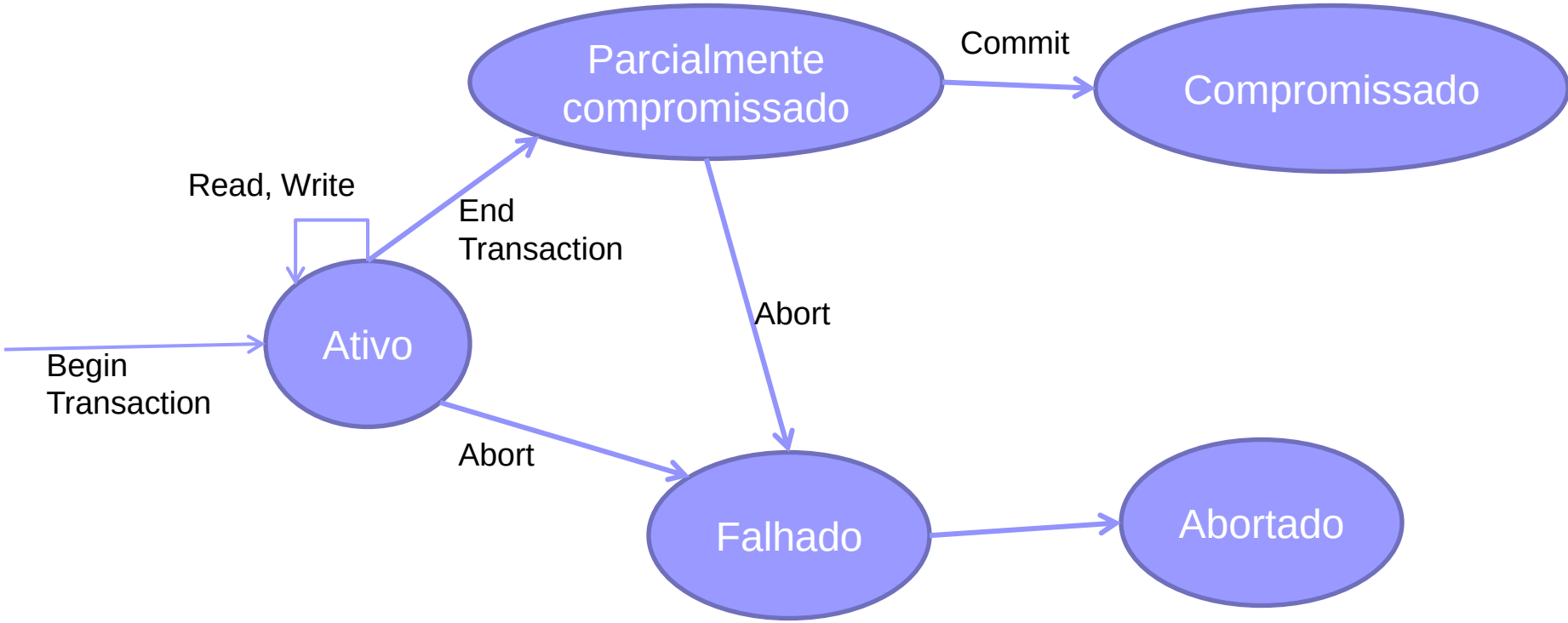
- Exemplo
  - *UPDATE funcionarios SET salario = (salario * 1.3) WHERE matricula = 569871;*

read(x)
x := x + N
write(x)

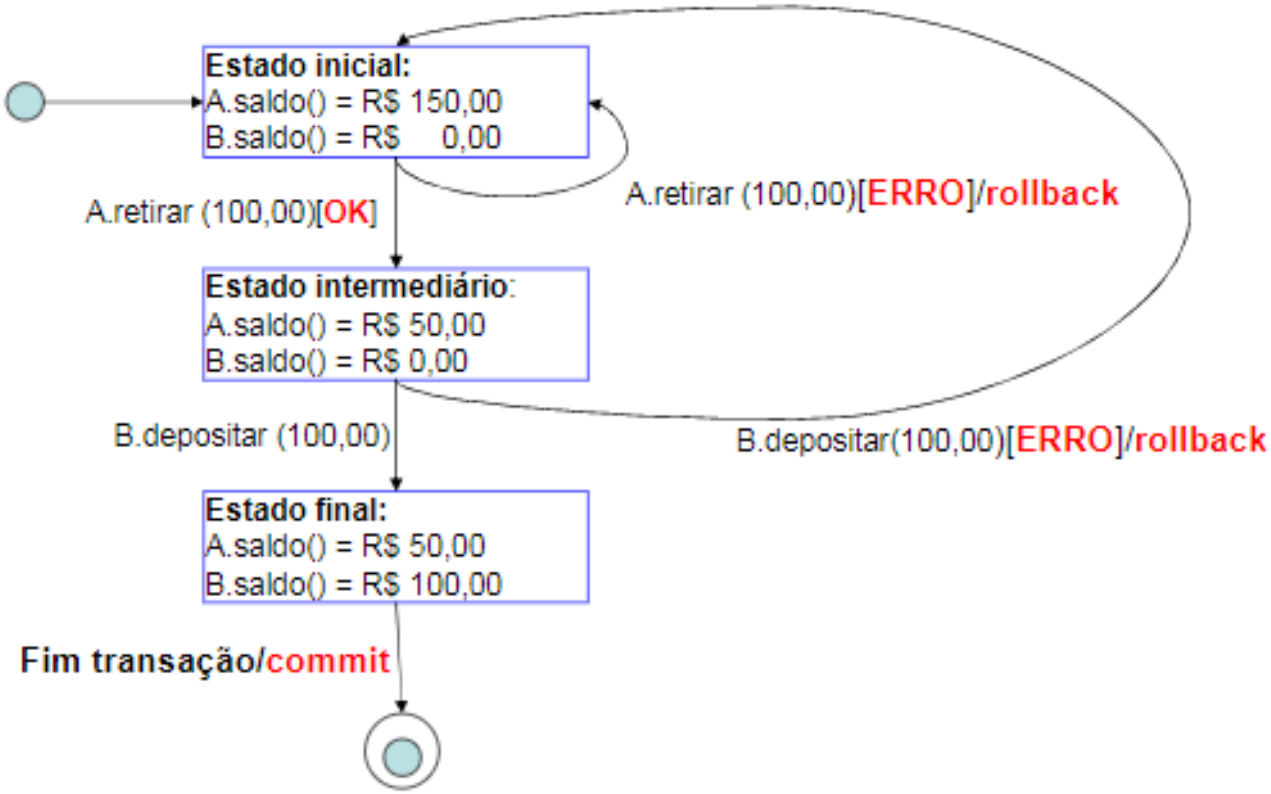
Operações de Leitura e Escrita

- Note que ambas as operações podem requerer uma operação Input, mas não requerem especificamente uma operação output.
- Um bloco de buffer é gravado eventualmente no disco porque o gerenciador de buffer precisa de espaço na memória ou porque o SGBD deseja refletir a mudança de X no disco.
- Uma operação output não precisa ser feita imediatamente depois que write é executado, uma vez que o bloco no qual X reside pode conter outros dados aos quais ainda se está fazendo acesso.
- Se o sistema cair depois da operação write e antes da operação output, o novo valor de X nunca será escrito no banco.

Estados de Transações



Estados de Transações



Transação ACID

- A integridade de uma transação depende de 4 propriedades, conhecidas como ACID:
  - Atomicidade
  - Consistência
  - Isolamento
  - Durabilidade

Atomicidade

- Princípio do “Tudo ou Nada”
  - Ou todas as operações da transação são efetivadas com sucesso no BD, ou nenhuma delas será efetivada.
    - Preserva a integridade do banco
- Responsabilidade do **subsistema de recuperação contra falhas** (*recovery*) do SGBD
  - Desfaz as ações de transações parcialmente executadas.

Triggers e Transações

```
-- QUESTÃO 3
CREATE TRIGGER atualizaEstoque BEFORE INSERT OR UPDATE ON vendas.ItensVenda
    FOR EACH ROW EXECUTE PROCEDURE validaQuantidade();
--
CREATE FUNCTION validaQuantidade() RETURNS trigger AS $emp_stamp$
    DECLARE
        estoque numeric;
    BEGIN
        estoque = (SELECT quantEstoque FROM vendas.Produtos AS pr WHERE pr.codigoProduto = new.codigoProduto);
        estoque = estoque - new.quantidade;
        IF (estoque >= 0) THEN
            UPDATE vendas.Produtos SET quantEstoque = estoque WHERE codigoProduto = new.codigoProduto;
        ELSE
            RAISE EXCEPTION '% Quantidade de estoque insuficiente', NEW.codigoProduto;
        END IF;
        RETURN NULL;
    END;
$emp_stamp$ LANGUAGE plpgsql;
```

Consistência

- Uma transação sempre conduz o BD de um estado consistente para outro estado também consistente.

- Responsabilidade conjunta :
  - DBA
    - Definir todas as regras de integridade
  - **subsistema de recuperação contra falhas** (*recovery*) do SGBD
    - Desfazer as ações de transações parcialmente executadas.

Isolamento

- Uma transação não deve sofrer interferência de outras que estejam sendo executadas de forma concorrente.

- Responsabilidade conjunta **subsistema de controle de concorrência** (*scheduler*) do SGBD
  - Garantir **escalonamentos** seguros.

Durabilidade

- Deve garantir que as modificações realizadas por uma transação que concluiu com sucesso persistam no BD.
  - Nenhuma falha posterior ocorrida no BD deve perder essas modificações
- Responsabilidade do subsistema de recuperação contra falhas (*recovery*) do SGBD
  - Refazer as transações que executaram com sucesso em caso de falha no BD.

Exercício

- Verificar a saída de cada select no código abaixo:

```
START TRANSACTION;
INSERT INTO northwind.categories
    VALUES(9, 'Higiene', 'Higiene e Perfumaria');
SELECT * FROM northwind.categories;
ROLLBACK;
SELECT * FROM northwind.categories;
```

Exercício

- Verificar a saída de cada select no código abaixo:

```
START TRANSACTION;
INSERT INTO northwind.categories
    VALUES(9, 'Higiene', 'Higiene e Perfumaria');
SELECT * FROM northwind.categories;
COMMIT;
SELECT * FROM northwind.categories;
```

Exercício

- Verificar a saída de cada select no código abaixo:

```
START TRANSACTION;
SAVEPOINT P1;
DELETE FROM northwind.categories WHERE categoryid < 5;
SELECT * FROM northwind.categories;
ROLLBACK TO SAVEPOINT P1;
SELECT * FROM northwind.categories;
DELETE FROM northwind.categories WHERE categoryid = 9;
SELECT * FROM northwind.categories;
COMMIT;
SELECT * FROM northwind.categories;
```

Para Casa

- Livro : Sistemas de Banco de Dados – Projeto, Implementação e Administração
  - Peter Rob e Carlos Coronel
  - 8ª Edição
  - Cengage Learning

- Estudar o Capítulo 10, até o tema ‘Gerenciamento de Transações SQL’