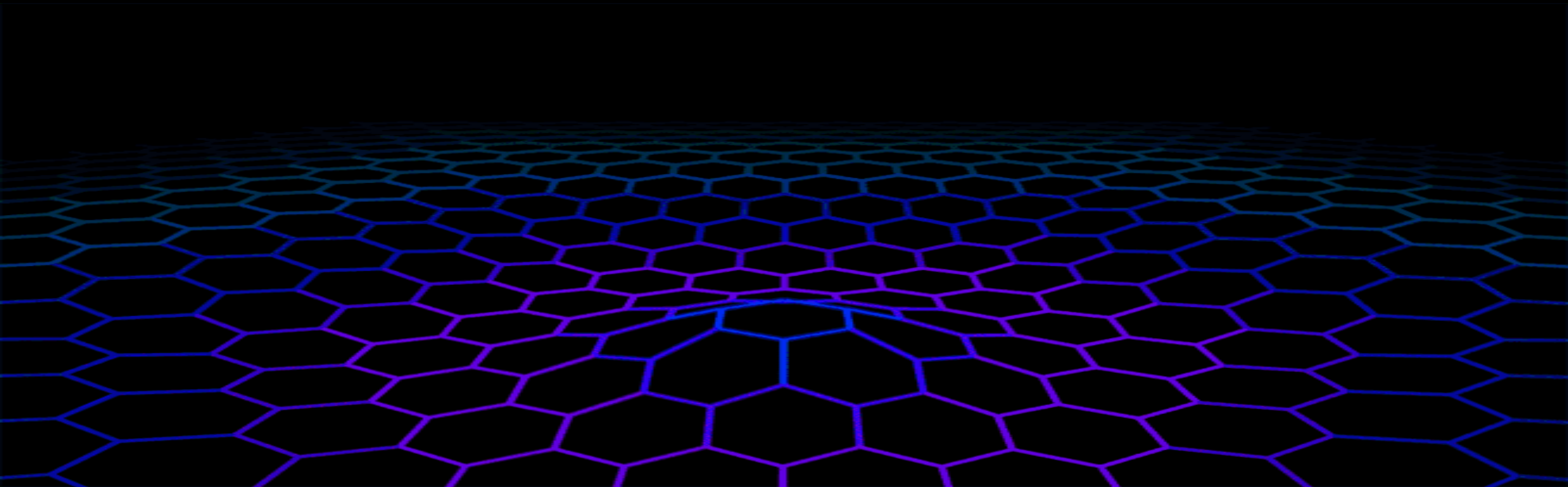
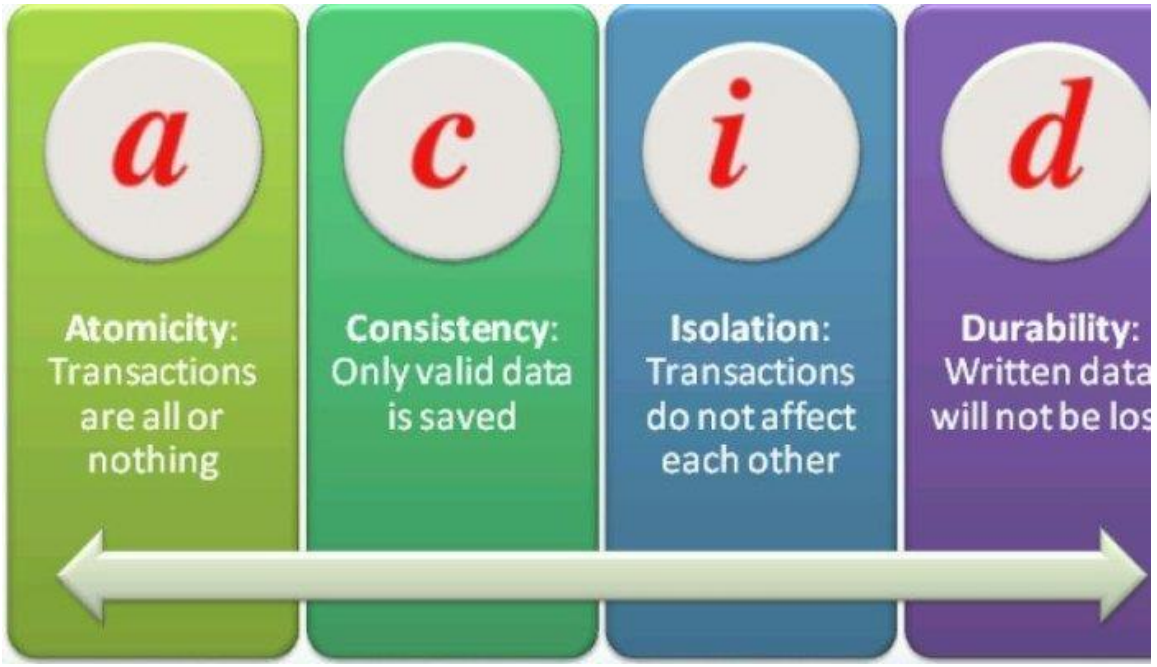
Banco de Dados 2



ACID

ACID

Conceito utilizado em ciência da computação para caracterizar uma transação.



União das iniciais de:

- **A**tomicidade
- **C**onsistência
- **I**solamento
- **D**urabilidade

Obs.: Todo Sistema Gerenciador de Banco de Dados aplica os conceitos de ACID. Caso isto não ocorra ele não pode ser considerado um SGBD.

ACID
Atomicidade

Na transação ou se faz tudo, ou nada, sem meio termo.

Em uma transação podemos ter mais de uma operação.

ACID
Atomicidade

Exemplo:

Em uma transação realizamos

- Inclusão de um cliente novo
- Geração de uma nota fiscal
- Baixa no estoque do produto vendido

Ao final, devemos confirmar a transação por inteiro e gravar todas estas operações, se esta transação não se confirmar ao final, nenhuma destas operações pode ser gravada no banco de dados, garantindo assim a atomicidade da transação".

ACID
Consistência

Garantir que o banco de dados antes da transação esteja consistente e, que após a transação o banco permaneça consistente, sem problemas de integridade.

Podemos contribuir com o trabalho do banco de dados, criando mecanismos que evitem problemas de integridade no banco.

ACID
Consistência

Exemplo:

O cliente de um banco possui um saldo de R$ 50,00
O cliente não tem limite de crédito (não pode ficar negativo)

Se a transação for uma retirada de R$ 60.00, ela não pode ser concluída pois a consistência do banco de dados não estaria garantida deixando a conta com um saldo negativo.

ACID
Isolamento

Garantir que nenhuma transação seja interferida por outra até que ela seja completada.

Existem transações que podem ocorrer de forma simultânea sob os mesmos dados, como por exemplo consultas.

ACID
Isolamento

Exemplo:

Duas transações são iniciadas
Ambas estão ligadas diretamente ao mesmo registro no banco de dados
A primeira atualizando
A segunda consultando

O isolamento nos garantirá que a transação de consulta somente será executada após a transação de atualização ser completada.

ACID
Isolamento

Exemplo:

No ato de consultas, podemos imaginar um sistema de vendas, em que o mesmo produto pode ser consultado várias vezes ao mesmo tempo, visando saber o valor deste.

ACID
Durabilidade

Garante que a informação gravada no banco de dados dure de forma imutável até que alguma outra transação de atualização, ou exclusão afete-a.

Garante que os dados não sejam corrompidos, ou seja, desapareçam ou se modifiquem sem motivo aparente.

Transações

Uma transação é um meio de executar uma ou mais instruções SQL com uma única unidade de trabalho, onde todas ou nenhuma das instruções são bem sucedidas.

Se todas as instruções foram concluídas sem erros, você poderá registrar estas em definitivo no banco de dados.

Caso ocorra algum erro, você poderá retornar a um ponto de salvamento (*savepoint*) ou efetuar um *rollback*, cancelando a alteração.

Exemplo

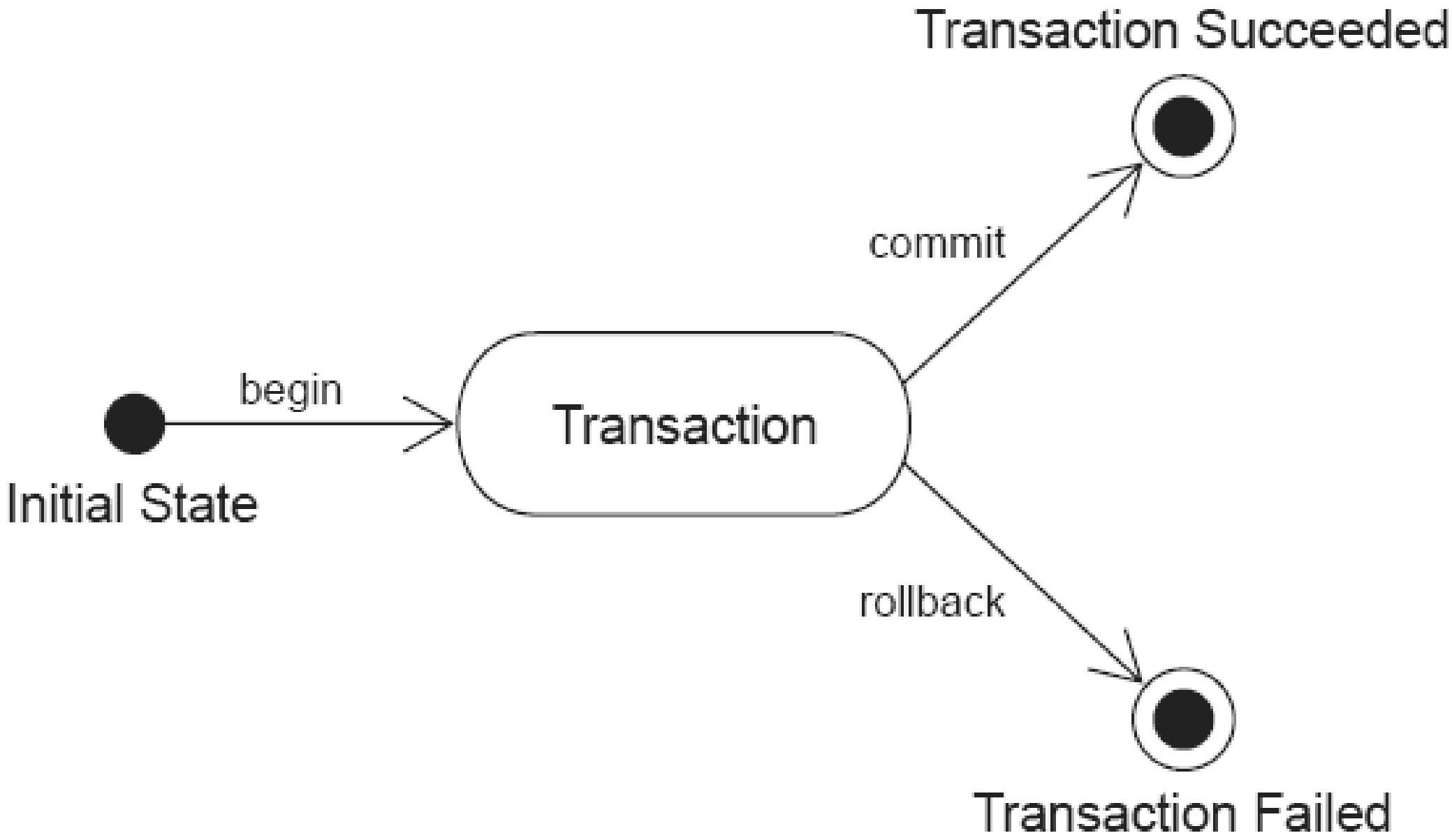
```
START TRANSACTION;
    SELECT @A:=SUM(salario)
    FROM funcionarios
    WHERE cargo='Programador';

    UPDATE resumo
    SET somatorio=@A
    WHERE cargo='Programador';
COMMIT;
```

Transações

No MySQL, apenas o InnoDB suportará transações em nível ACID.
Estas opções não são validas ou não tem o efeito esperado em outros *Engines*.

Transações


Transaction Succeeded
commit
begin
Transaction
Initial State
rollback
Transaction Failed

Transações - Principais instruções

Start Transaction (ou BEGIN):

Inicia explicitamente uma nova transação, mantendo ela aberta até que seja fechada (concluída) por um COMMIT ou ROLLBACK.

Transações - Principais instruções

SavePoint

Um ponto delimitado na transação, que oferece a possibilidade de um *rollback* de qualquer comando executado após este ponto salvo.

Atenção: Ao efetuar um novo savepoint com o mesmo nome, o antigo será sobreescrito pelo novo criado.

Transações - Principais instruções
Rollback to Savepoint

Efetua o *rollback* da transação até o ponto criado.

Este ponto continuará a existir mesmo após o *rollback*, permitindo que ele seja reutilizado.

Transações - Principais instruções

Rollback

Cancela todas as alterações efetuadas na transação.

Transações - Principais instruções

Commit

Torma permanente as alterações tratadas na transação ao banco de dados.

Transações - Principais instruções
AUTOCOMMIT

É ativado por padrão, logo o MySQL confirma implicitamente cada instrução como uma transação.

Uma vez desativado, será possível manusear cada transação, porém fica a critério do DBA.

Transações - Principais instruções
AUTOCOMMIT

Para desativar o autocommit:

```
set session autocommit=0;
set autocommit=0;
```

Transações - Principais instruções
AUTOCOMMIT

Para verificar a configuração através do SELECT:

```
SELECT @@autocommit;
```

Transações - Principais instruções
AUTOCOMMIT

Obs.: A partir da versão 5.5.8 do MySQL, é possível definir o autocommit = 0 de forma global na seção do [mysqld] no arquivo de configuração (`my.cnf`).

```
[mysqld]
autocommit = 0
```

Níveis de Isolamento

O padrão SQL define quatro níveis de isolamento de transação:
Read uncommitted
Read committed
Repeatable read
Serializable

Para evitar três fenômenos entre transações simultâneas:
dirty read (leitura suja)
nonrepeatable read (leitura que não pode ser repetida)
phantom read (leitura fantasma)

Níveis de Isolamento - Fenômenos
Dirty read (leitura suja)

Suponhamos que a transação “A” modifique algum campo da tabela, porém que ainda não o tenha “commitado”.

Se uma transação “B” efetua um SELECT neste campo e vê o valor modificado pela transação “A” sem ter o **commit** efetuado, essa é uma leitura suja.

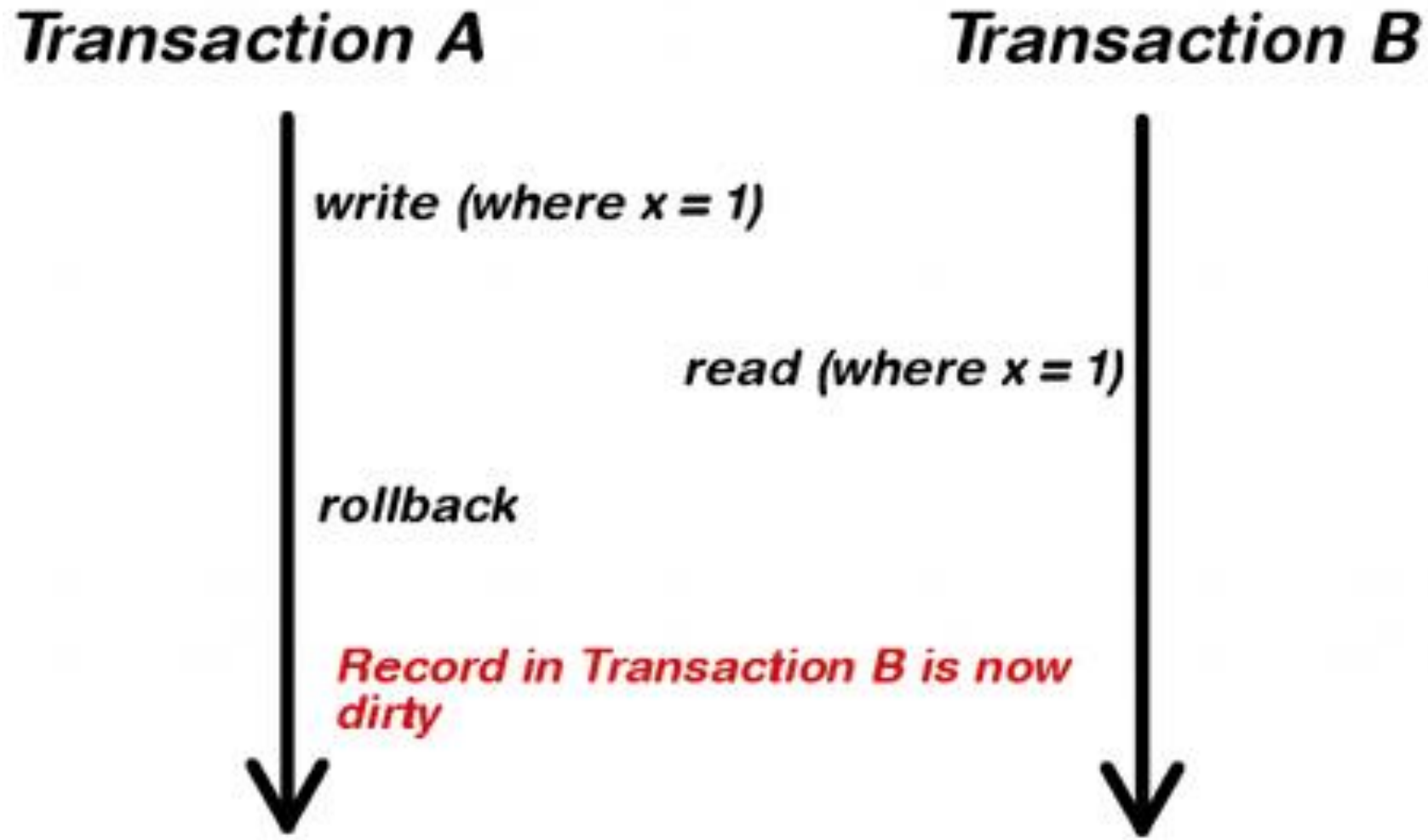
Níveis de Isolamento - Fenômenos
Dirty read (leitura suja)

Isso é um problema em ambientes de tomada de decisão, Relatórios entre outros.

Caso a transação “A” sofra um *Rollback*, a transação “B” não saberá disso e já terá informado o valor “errôneo”.

Níveis de Isolamento - Fenômenos
Dirty read (leitura suja)

dirty read
(leitura suja)



Níveis de Isolamento - Fenômenos

Nonrepeatable read (leitura que não pode ser repetida)

Ocorre quando um SELECT(leitura) reproduz resultados diferentes quando ela é repetida posteriormente na mesma transação.

Níveis de Isolamento - Fenômenos

Nonrepeatable read (leitura que não pode ser repetida)

Exemplo:

A transação “A” lê o valor de um campo.
Outra transação “B” pode atualizar este valor e “commitá-lo” no banco.
Caso a transação “A” volte a consultar o mesmo campo, ela trará o valor “comitado” pela transação “B”, ou seja, trata valores diferentes do mesmo campo na mesma transação executada.

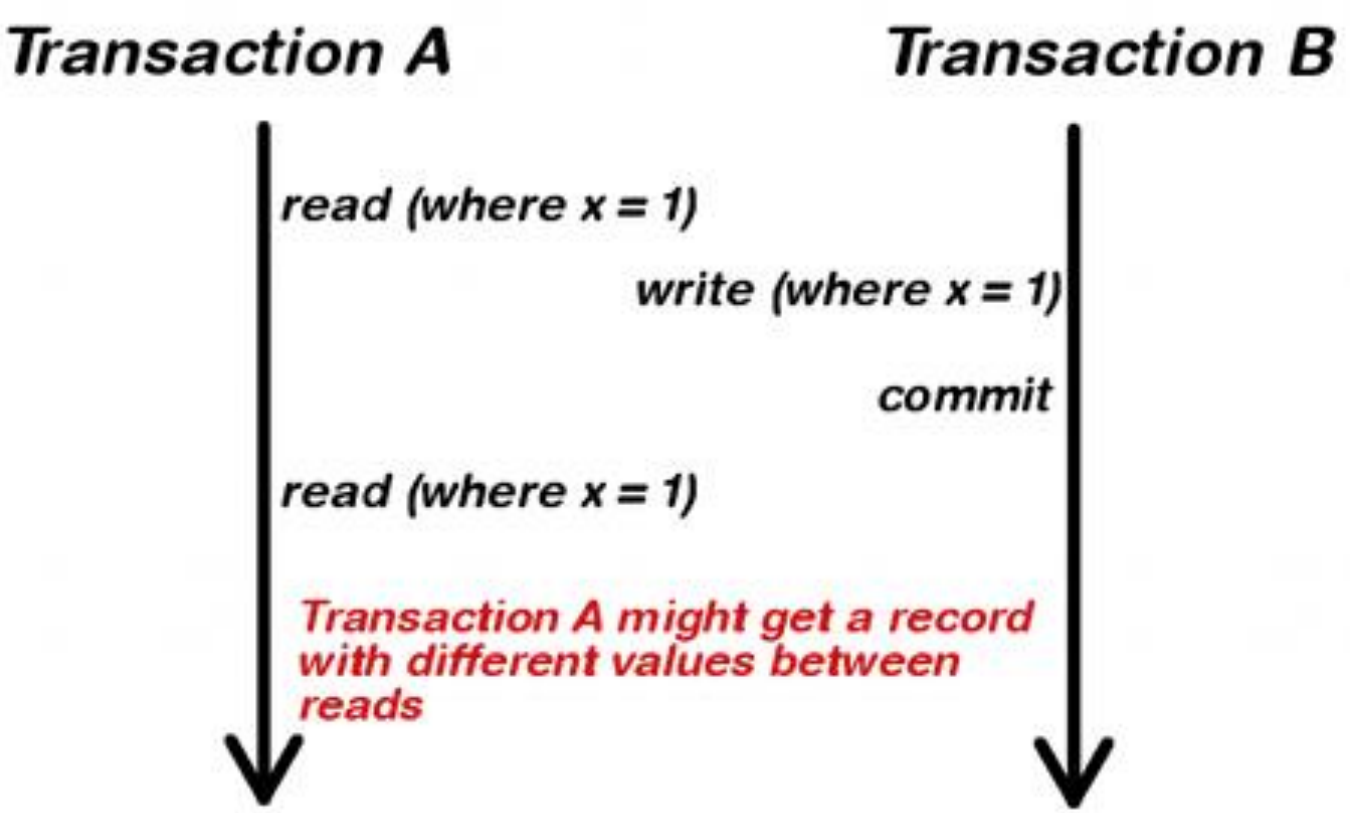
Níveis de Isolamento - Fenômenos

Nonrepeatable read (leitura que não pode ser repetida)

Isso é grave caso você opte por alterar algum registro com a condição de um campo x, onde ele pode assumir um valor y em seguida.

Níveis de Isolamento - Fenômenos

Nonrepeatable read (leitura que não pode ser repetida)



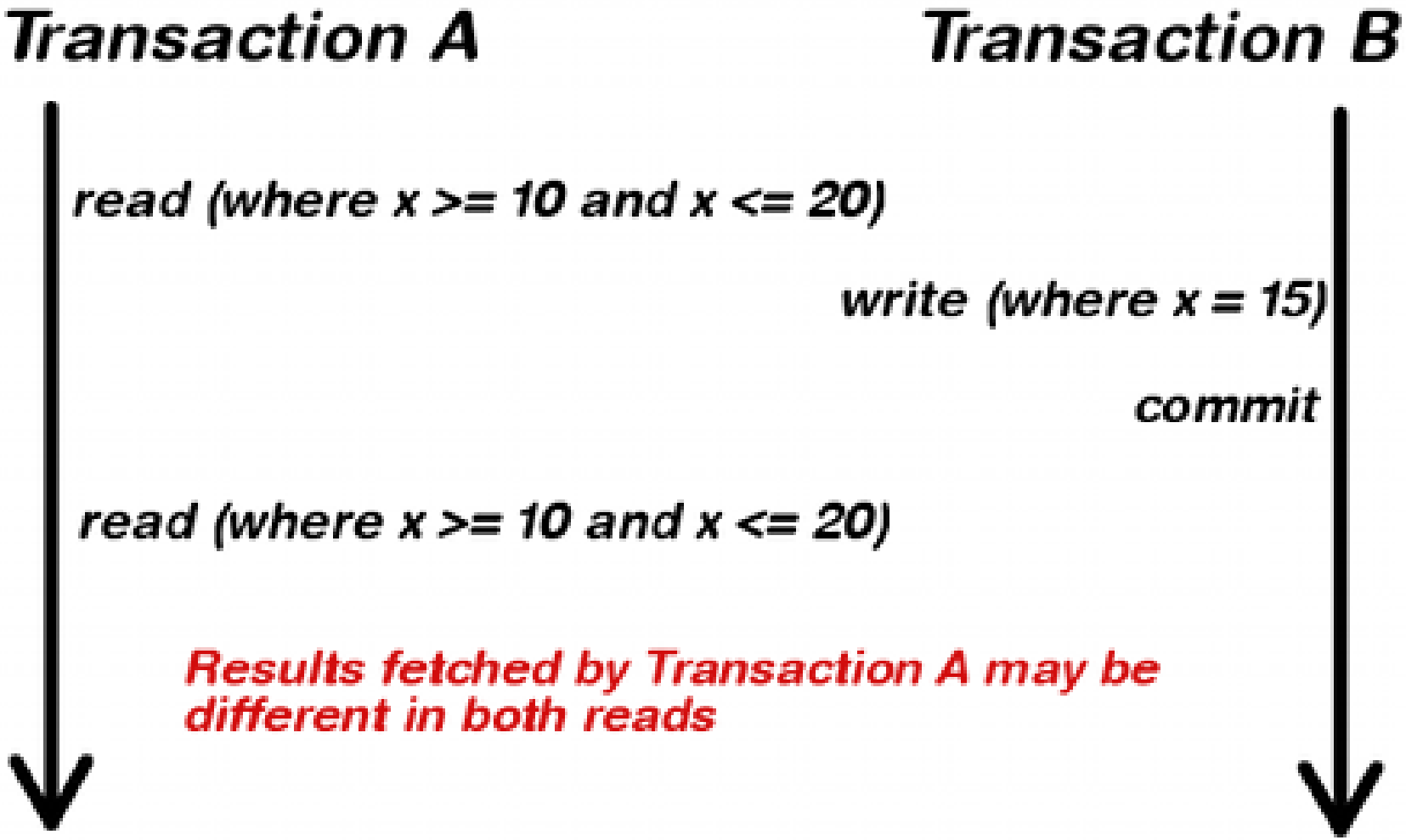
Níveis de Isolamento
Phantom read (leitura fantasma)

Uma transação “A” pode ler um conjunto de linhas de uma tabela com base em alguma condição WHERE SQL.

Suponhamos que a transação “B” insira uma nova linha que também satisfaz a clausula WHERE na tabela utilizada pela transação A.

Se a transação “A” for repetida ela verá um fantasma, ou seja, uma linha que não existia na primeira leitura utilizando a clausula WHERE.

Níveis de Isolamento
Phantom read (leitura fantasma)



Níveis de Isolamento
Repeatable Read

Este nível garante que a mesma leitura de um dado através do SELECT se repita, tendo o mesmo resultado para diferentes execuções na mesma transação.

Se neste nível a leitura não fosse repetida, ela estaria aberta a leitura fantasma, que acontece entre um SELECT e caso ocorra uma atualização nos dados neste espaço de tempo.

Níveis de Isolamento
Repeatable Read

É possível acontecer a leitura fantasma com este nível de bloqueio utilizando o Engine InnoDB.

O Repeatable Read é o nível de Isolamento Default do MySQL.

Níveis de Isolamento
Read Commited

Permite que a transação leia manipule os dados já “commitados“ por outras transações.

Caso alguma transação tenha alterado algum dado porém sem efetuar um *commit*, estes não serão vistos.

Com este nível de isolamento, você evita as leituras sujas, porém esta aberto para as leituras fantasmas e a Leitura não repetitiva.

Níveis de Isolamento
Read Uncommited

Este nível permite que uma transação possa ver e manipular valores não “commitados” por outras transações, ficando aberto para leituras sujas e Leitura não repetitiva, facilitando também casos de Leituras fantasmas.

Níveis de Isolamento
Serializable

Semelhante ao Repeatable Read, porém com a restrição adicional de que as linhas selecionadas por uma transação não podem ser alteradas ou lidas por outra transação, até que a primeira transação seja concluída.

Este nível isola completamente uma transação da outra, onde a segunda transação aguarda a finalização da primeira, e assim por diante “uma de cada vez”, evitando os fenômenos indesejados.

Níveis de Isolamento

Por default o MySQL trabalha com o nível de isolamento **Repeatable Read**, sendo possível a sua alteração.

Níveis de Isolamento

Para consultar o nível atual na instância:

```
SHOW variables LIKE '%isolation%';
+---------------+-----------------+
| Variable_name | Value           |
+---------------+-----------------+
| tx_isolation  | REPEATABLE-READ |
+---------------+-----------------+


SELECT @@tx_isolation;
+-----------------+
| @@tx_isolation  |
+-----------------+
| REPEATABLE-READ |
+-----------------+
```

Níveis de Isolamento

Para alterar o nível de isolamento na instância, você pode usar a opção **`–transaction-isolation`** com o comando **`mysqld`**, ou alterar diretamente no **`my.cnf`**:

```
[mysqld]
transaction-isolation = 'isolation_level'
```

Níveis de Isolamento

Tabela com níveis de isolamento da transação no SQL

Nível de isolamento	Dirty Read	Nonrepeatable Read	Phantom Read
Read uncommitted	Possível	Possível	Possível
Read committed	Impossível	Possível	Possível
Repeatable read	Impossível	Impossível	Possível
Serializable	Impossível	Impossível	Impossível

Fonte: http://sqlparatodos.com.br/niveis-de-isolamento-mysql/

Verificar o autocommit

```
SELECT @@autocommit;
```

0 - Desativado
1 - Ativado

Habilitar o autocommit

```
SET autocommit = 1;
SET autocommit = ON;
SET @@autocommit = ON;
```

Desabilitar o autocommit

```sql
SET autocommit = 0;
SET autocommit = OFF;
SET @@autocommit = OFF;
```

Criando tabela para testes

```sql
CREATE TABLE ator (
  id    INT AUTO_INCREMENT,
  nome VARCHAR(45) NOT NULL,
  PRIMARY KEY (id)
);

INSERT INTO ator (nome) VALUES('Adam Sandler');
INSERT INTO ator (nome) VALUES('Hey Decio');
INSERT INTO ator (nome) VALUES('Jamie Foxx');
INSERT INTO ator (nome) VALUES('Joaquim Phoenix');
INSERT INTO ator (nome) VALUES('Jude Law');
INSERT INTO ator (nome) VALUES('Meryl Streep');
INSERT INTO ator (nome) VALUES('Michael Douglas');
INSERT INTO ator (nome) VALUES('Tom Cruise');
INSERT INTO ator (nome) VALUES('Will Smith');
```

Transação com rollback

```sql
START TRANSACTION;
  DELETE FROM ator; -- Apaga todos os registros da tabela ator
  SELECT * FROM ator; -- Observe que a tabela está vazia
  INSERT INTO ator (nome) VALUES('Will Smith');
  SELECT * FROM ator; -- Observe que agora só exibe o ator Will Smith
ROLLBACK; -- Desfaz a transação

SELECT * FROM ator; # Exibe os dados iniciais pois nada foi commitado.
```

Transação com commit

```
START TRANSACTION;
  DELETE FROM ator; -- Apaga todos os registros da tabela ator
  SELECT * FROM ator; -- Observe que a tabela está vazia
  INSERT INTO ator (nome) VALUES('Will Smith');
  SELECT * FROM ator; -- Observe que agora só exibe o ator Will Smith
COMMIT; -- Confirma a transação

SELECT * FROM ator;
/* Observe que agora os dados iniciais
foram excluídos e somente o ator
"Will Smith" é exibido (transação foi confirmada) */
```

Transação em Stored Procedure

```
DELIMITER $$
CREATE PROCEDURE insere_ator()
BEGIN
DECLARE sql_erro TINYINT DEFAULT FALSE;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET sql_erro = TRUE;
START TRANSACTION;
  INSERT INTO ator (nome) VALUES('Angelo Luz');
  INSERT INTO ator (nome) VALUES('Pablo Escobar');
  INSERT INTO ator (nome) VALUES(NULL);
  -- Na última inserção há um erro que impedirá o COMMIT e provocará o ROLLBACK.
  IF sql_erro = FALSE THEN
    COMMIT;
    SELECT 'Transação OK' AS Situação;
  ELSE
    ROLLBACK;
    SELECT 'Erro na transação' AS Situação;
  END IF;
END$$
DELIMITER ;
```

Transação em Stored Procedure

```sql
CALL insere_ator();

SELECT * FROM ator;
-- Nenhum dado foi inserido, pois há um erro no último insert.
```

Habilitando o autocommit

```
SET @@autocommit = ON;
```

Script 1

O arquivo `BD2-A12-Parte1.SQL` contém o script com as instruções dos slides anteriores.

Script 2

Com a orientação do professor Rimidalg, execute passo-a-passo as instruções contidas no arquivo `BD2-A12-Parte2.SQL`

Ainda faltam 2(dois) slides (não te apressa)

MySQLDUMP

Ferramenta para copiar bancos de dados do SGBD MySQL.

Programa cliente que coloca o conteúdo de tabelas em arquivos texto, chamados de DUMP.

Permite realizar cópias de segurança de bancos de dados.

Produzir arquivos no formato SQL que contenham declarações CREATE TABLE e INSERT.

Exemplo:

```
mysqldump -h localhost -u root -p aula12 > "d:/temp/backup_aula.sql"
```

SOURCE

É possível colocar seus comandos SQL em um arquivo e dizer ao MySQL para ler a entrada a partir deste arquivo.

```
mysql -u root -p aula12 < d:\temp\backup_aula.sql
```

Se já estiver executando o MySQL, você pode executar o comando source:

```
SOURCE d:/temp/backup_aula.sql;
```