TRANSAÇÕES

Uma transação é a execução de um programa, vista pelo SGBD como uma série de leituras e escritas, efetuadas por múltiplos utilizadores, sem que haja interferência nas operações de cada um, não havendo, portanto, colisões.

AS PROPRIEDADES ACID

Um SGBD deve garantir quatro propriedades de transação para manter os dados protegidos de acesso concorrente e de falhas de sistema:

- **Atomicidade**: O resultado das operações não deve ser parcialmente executado; ou todas as ações são executadas ou nenhuma delas o é;
- **Consistência**: Cada transação deve preservar a consistência do banco de dados, isto é, devem ser respeitadas as regras de integridade e não deve haver falha de um estado para o outro;
- **Isolamento**: Toda transação é isolada ou protegida das ações de outras transações concorrentes (se um utilizador acede a um registo e o outro o modifica, a transação deve funcionar sem interferência);
- **Durabilidade**: Os efeitos da transação devem permanecer mesmo em caso de falhas.

Problema: Suponhamos que o Sr. X pretende transferir 10 000 € da sua conta bancária para a conta do Sr. K.
Uma transferência consiste num depósito numa conta e num levantamento, do mesmo montante, noutra conta.

O conjunto de comandos SQL a executar para realizar a referida transação poderia ser substituída pelos 2 comandos UPDATE que se seguem:

```
UPDATE tabela_conta
SET Saldo= Saldo +10000
WHERE Conta="conta K"
```

Creditar a conta do Sr. K

```
UPDATE tabela_conta
SET Saldo= Saldo - 10000
WHERE Conta="conta X"
```

Debitar a conta do Sr. X

E se existisse algum problema e apenas fosse realizado o 1ª UPDATE? Ou apenas fosse realizado o 2ª UPDATE?

Para evitar estas questões surge o **conceito de transação** que consiste numa unidade lógica de trabalho em que todas as operações são realizadas ou em que nenhuma das operações é realizada, pode-se dizer que é um caso de "tudo ou nada".

As transações surgem, assim, como resposta a situações muito comum no dia-a-dia, que é o facto de haver operações que só podem ser realizadas se o forem por inteiro. Caso contrário têm que ser anuladas, até ao ponto de partida.

No exemplo anterior, independentemente da ordem pela qual os comandos são realizados, se existir alguma falha, as contas bancárias deverão permanecer inalteradas. Caso não haja erros de execução todas as operações serão efetuadas e as contas movimentadas com sucesso.
Outro exemplo: imaginemos que temos um voo e que, nesse aeroporto, o check in está a ser feito em 4 balcões diferentes.

Os funcionários apenas sabem os lugares existentes de acordo com o que o sistema lhe transmite, mas a questão é que todo este processo é dinâmico e ocorre em simultâneo. Assim, torna-se necessário assegurar que não sejam marcados dois passageiros para o mesmo lugar.

Caraterísticas das transações

- Uma vez iniciada uma transação, todas as alterações feitas aos dados por um utilizador só são visíveis por ele, todos os outros utilizadores da base de dados acedem aos dados como se a transação não tivesse sequer sido iniciada;

- No espaço de tempo em que uma transação é realizada, os dados alterados são colocados num repositório temporário (registo log) até ser aplicado o comando COMMIT, passando a partir daí as alterações realizadas a definitivas, ou se aplicado o comando ROLLBACK, estas são ignoradas.

- Depois de realizar um comando COMMIT não se pode realizar um ROLLBACK sobre a mesma transação, pois a mesma deixou de existir. Os dados já foram, fisicamente, colocados na base de dados.

- O conceito de transação só se aplica aos comandos que fazem a manipulação de dados, isto é, INSERT, UPDATE e DELETE. Se, uma tabela é apagada com o comando DROP TABLE não há ROLLBACK que lhe valha!

- Uma transação inicia-se com o comando START TRANSACTION OU BEGIN TRANSACTION e termina com o comando COMMIT ou ROLLBACK.

    - **COMMIT** – Faz com que todas as alterações efetuadas se tornem definitivas;

    - **ROLLBACK** – Elimina as alterações realizadas, ficando a BD no estado em que se encontrava antes de a transação ser iniciada.

Sintaxe

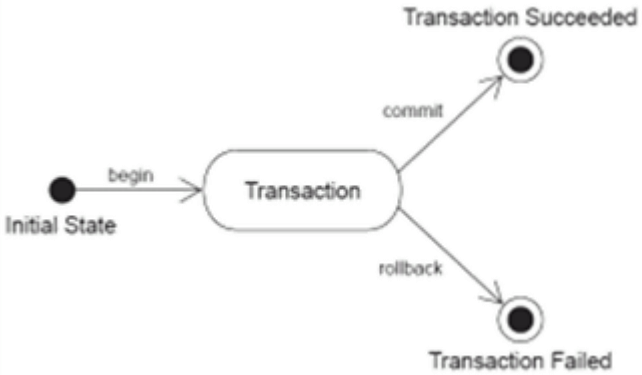
Iniciar uma nova transação.

BEGIN TRANSACTION

Concluir uma transação consolidando todo o trabalho efetuado durante a transação.

COMMIT [TRANSACTION | WORK]

ROLLBACK [TRANSACTION | WORK]



Observações

As transações não são iniciadas automaticamente, esta só é iniciada utilizando BEGIN/START TRANSACTION.

As transações podem estar aninhadas até cinco níveis de profundidade. Para iniciar uma transação aninhada, utilize BEGIN TRANSACTION no contexto de uma transação existente.

No mysql o modo COMMIT, é por defeito, automático, contudo é possível "desligar" este modo de COMMIT automático (autocommit), utilizando a instrução seguinte (0—desligado e 1— ligado. Tal permite reverter o processo, através do comando ROLLBACK.):

```
SET global autocommit = {0|1}
```

Exercício:

No o PhpMyAdmin abre a base de dados VendasCD.

Executa as seguintes instruções em SQL:

a) Em primeiro lugar irás alterar as definições do autocommit

```
SET global autocommit=0
```

b) Verifica se realmente ouve alterações nas configurações do autocommit (o resultado deve ser 0) executando:

```
select @@autocommit
```

c) Altera o nome da loja Adega para Cave, mas aplicando o ROLLBACK:

```
START TRANSACTION;
UPDATE lojas
SET nome='Cave' WHERE nome='Adega';
ROLLBACK;
```

Nota: é obrigatório colocar um ponto e vírgula (;) para separar as instruções

Visualiza o resultado

d) Altera o nome da loja Adega para Cave, mas agora aplicando o COMMIT:

```
START TRANSACTION;
UPDATE lojas
SET nome='Cave' WHERE nome='Adega';
COMMIT;
```

Visualiza o resultado

e) Aplica o comando SET global autocommit=1, certificando-te que o valor realmente passou a 1!

f) Volta a executar o update, mas agora repondo o nome inicial Adega.

Visualiza o resultado

Níveis/Graus de isolamento de uma transação – isolation degree

Quando se trabalha com transações é possível definir o grau de isolamento que corresponde à forma como decorrem as transações de leitura e escrita, que vai desde o grau mais rápido ao grau mais seguro. É possível controlar a variação das caraterísticas da transação, que pode ser do tipo:

GLOBAL – aplica-se o grau de isolamento a todas as transações que ocorram nas demais sessões de forma global, não sendo as atuais afetadas por esta alteração;

SESSION – aplica-se o grau de isolamento a todas as transações subsequentes realizadas dentro da sessão, não sendo as atuais afetadas por esta alteração;

Sem SESSION ou GLOBAL – aplica-se à transação seguinte que venha a ocorrer, mas ainda não tenha sido iniciada, realizada dentro da sessão atual.

Se for tentada a alteração destas caraterísticas enquanto decorrem as transações ocorrerá o erro 1568 (25001): `Transaction characterisitics can´t be changed while a transaction is in progress.`
`Nestes casos SERÁ NECESSÁRIO FORÇÃO O COMMIT.`

GRAUS DE ISOLAMENTO

READ UNCOMMITTED (leitura não confirmada) – é o que permite maior rapidez (menor segurança), e o SELECT de um utilizador pode ocorrer em simultâneo com outras operações de SELECT em transações não terminadas. Maximização da eficiência em detrimento da integridade e segurança do sistema. Permite que numa transação sejam visualizados dados manipulados por outras transações e que não sofreram um COMMIT– as chamadas "leituras sujas"

READ COMMITTED (leitura confirmada) – tem em conta as operações de SELECT que vão decorrendo, ou seja, o mesmo comando SELECT pode devolver resultados diferentes se for executado durante ou depois do COMMIT, uma vez que há uma atualização de dados. Dados atualizados, mas que ainda não receberam um COMMIT explícito não serão vistos, só permitirá que a transação atual leia e manipule apenas os dados que já receberam um COMMIT por outras transações;

REPEATEABLE READ (leitura repetida) – é o nível padrão (para InnoDB) e tem em conta apenas as transações finalizadas com o COMMIT. Tal significa que se forem feitas várias consultas, estas serão sempre feitas em relação ao mesmo conjunto de dados. Garante que os dados não serão afetados por quaisquer outras transações durante o período em que estiverem a ser lidos. Reduz o nível de concorrência da base de dados e minimiza a eficiência da aplicação, mas garante a integridade e consistência absoluta da informação.

SERIALIZABLE – é semelhante á opção anterior, apenas com a diferença de que o SELECT é realizado em LOCK IN SHARE MODE *[1]. Este nível de isolamento, isola completamente uma transação de outra, isto é, quando uma está em execução a outra aguarda a sua conclusão. Semelhante ao REPEATEABLE READ, mas em que as linhas selecionadas por uma transação, nem são lidas nem atualizadas por outra transação.

Para verificar qual é o isolation degree, inserir a instrução:

```
SELECT @@tx_isolation ou SELECT @@transaction_isolation ou
```

```
SHOW GLOBAL VARIABLES LIKE 'TRANSACTION_isolation';
```

No caso de estar em uso o isolation degree padrão do MySQL é devolvido o resultado:

```
REPEATEABLE READ
```

Para alterar o isolation degree para, por exemplo, READ UNCOMMITED, dentro de uma sessão, deve ser feito com a seguinte instrução:

```
SET SESSION TRANSACTION_ISOLATION = 'READ UNCOMMITTED'
```

Sintaxe geral:

```
SET [GLOBAL|SESSION] TRANSACTION_ISOLATION = 'REPEATABLE READ|READ COMMITTED|READ
UNCOMMITTED|SERIALIZABLE'
```

Para verificar qual o modo do isolation degree (só de leitura ou leitura e escrita):

```
SELECT @@transaction_read_only;
```

(Se 1 verdade se 0 falso)

Alterar o modo de acesso na transação:

```
SET global TRANSACTION_READ_ONLY = {0|1}
```

SAVEPOINT

```
SAVEPOINT nome_da_marca

ROLLBACK [WORK] TO [SAVEPOINT] nome_da_marca

RELEASE SAVEPOINT nome_da_marca
```

Se COMMIT ou ROLLBACK for emitido e nenhuma transação tiver sido iniciada, nenhum erro será relatado.

Exercícios:

Considera a base de dados Vendas CD.

1. Para inserir "marcas" a fim de possibilitar um rollback de apenas partes da transação podemos utilizar o comando:

   ```
   SAVEPOINT <nome_da_marca>;
   ```

2. Executa as instruções seguintes todas de uma vez:
   a. Inicia uma transação:
   `START TRANSACTION;`

   b. Insere na tabela editoras os dados:

   ```
   INSERT INTO Editoras (CodEditora, NomeEditora)
   VALUES (4,'Vira o disco');
   ```

   c. Estabelece um SAVEPOINT:

   ```
   SAVEPOINT incluir_ok;
   ```

   d. Inclui outro registo na mesma tabela:

   INSERT INTO Editoras VALUES (5,'Toca Outra');

   g. Utiliza o comando
   rollback TO incluir_ok;
   e de seguida o comando
   COMMIT
   Executa as instruções SQL
3. Executa o seguinte SQL: `SELECT * FROM Editoras;`
   Observa o resultado e interpreta-o.

4. Altera o modo da sessão para READ ONLY, verificando, após a execução da instrução o modo de acesso na transação. E inicia uma transação.
5. Insere novos registos:
   ```
   INSERT INTO Editoras VALUES (5,'Toca Outra'), (6, 'DiscoSound');
   ```
   Seguido de COMMIT.
   Observa o resultado e explica-o.

6. Altera o modo da sessão de modo a poderes executar o SQL do exercício 5.

7. Altera o isolation degree para SERIALIZABLE, verificando, após a execução da instrução o grau de isolamento definido.

Bibiografia

Damas, L. (2005). SQL. Lisboa: FCA
Tavares, F. (2015). MySql. Lisboa: FCA
https://dev.mysql.com/doc/refman/5.6/en/commit.html
https://dev.mysql.com/doc/refman/8.0/en/set-transaction.html
https://mariadb.com/kb/pt-br/isolation/
https://mariadb.com/docs/reference/mdb/system-variables/tx_isolation/#dynamically-setting-the-global-transaction-isolation
https://pt.slideshare.net/Wagnerbianchi/7-my-sql-56-transacoes
http://www.mysqltutorial.org/mysql-table-locking/
https://dev.mysql.com/doc/refman/8.0/en/set-transaction.html