Modelo Físico de Banco de Dados e Linguagem SQL para Definição de Dados.

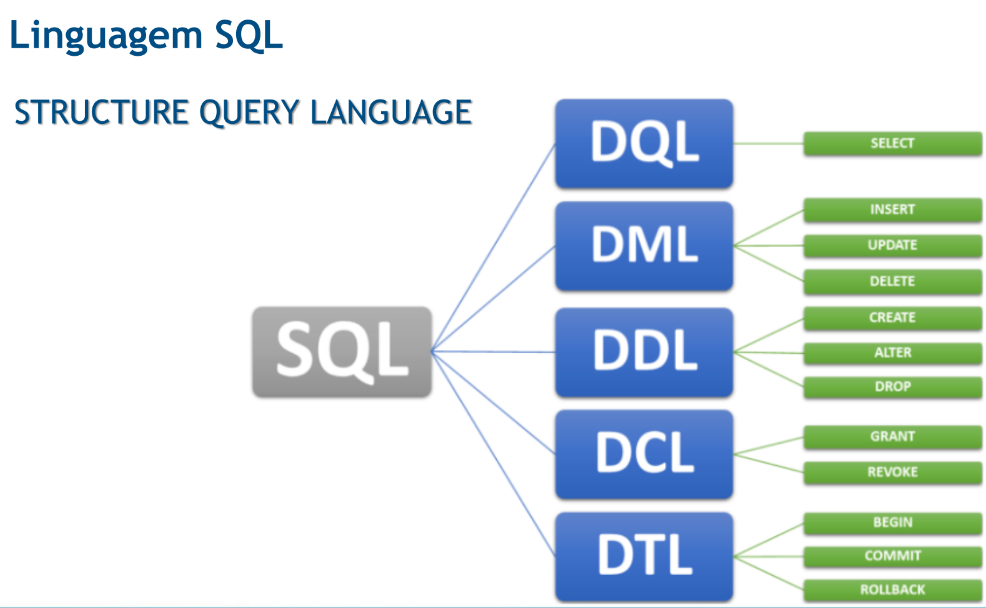
A linguagem SQL é utilizada para criarmos os scripts SQL que referência a criação do esquema de banco de dados e a manipulação de suas instâncias. É, portanto, através da Structure Query Language (SQL) que criamos o modelo físico de banco de dados.

A partir do Modelo Relacional de Banco de Dados, busca-se criar o Modelo Físico de Banco de Dados. O Modelo Físico de Banco de Dados, refere-se diretamente a criação do esquema de banco de dados criado em base de dados universal, ou seja, criado em SQL.

A Linguagem SQL (Structured Query Language) é uma linguagem de banco de dados que permite acessar banco de dados no modelo relacional. Essa linguagem é usada para acessar os principais banco de dados existentes hoje, SGBD, como o MySQL, SQL Server Microsoft, Oracle etc.

## Principais características do MySQL

* O servidor de banco de dados MySQL é extremamente rápido, confiável, e fácil de usar;
* O Servidor MySQL também tem um conjunto de recursos muito práticos desenvolvidos com a cooperação dos próprios usuários;
* O Servidor MySQL foi desenvolvido originalmente para lidar com bancos de dados muito grandes de maneira muito mais rápida que as soluções existentes;
* Apesar de estar em constante desenvolvimento, o Servidor MySQL oferece hoje um rico e proveitoso conjunto de funções;
* A conectividade, velocidade, e segurança fazem com que o MySQL seja altamente adaptável para acessar bancos de dados na Internet;
* MySQL é um Sistema de Gerenciamento de Bancos de Dados relacional;
* Um banco de dados relacional armazena dados em tabelas separadas em vez de colocar todos os dados em um só local, proporcionando mais velocidade e flexibilidade.



### **DDL – Data Definition Language**

* É utilizada para a definição da estrutura do banco de dados e apresenta um conjunto de instruções próprias para isto, ou seja, os ESQUEMAS de banco de dados.
* Criando Banco de Dados:
  + CREATE DATABASE <Nome\_Banco>;

Permite criar um banco de dados no SGBD.

Exemplo: Create Database Empresa;

* Mostrando Banco de Dados:
  + SHOW DATABASES;

Permite visualizar todos os bancos de dados existentes no servidor do SGBD.

Exemplo: Show Databases;

Observação: Todos os bancos de dados que estiverem contidos naquele diretório o qual esteja trabalhando, serão automaticamente exibidos.

* Colocando um Banco de Dados em Uso
  + USE <Nome\_Banco>;

Antes de realizar qualquer operação com um banco de dados já criado, é necessário selecionar o banco de dados que será utilizado.

Exemplo: Use Empresa;

Observação: Precisamos ter um banco de dados em uso para que possamos criar esquemas e manipular instâncias com ele.

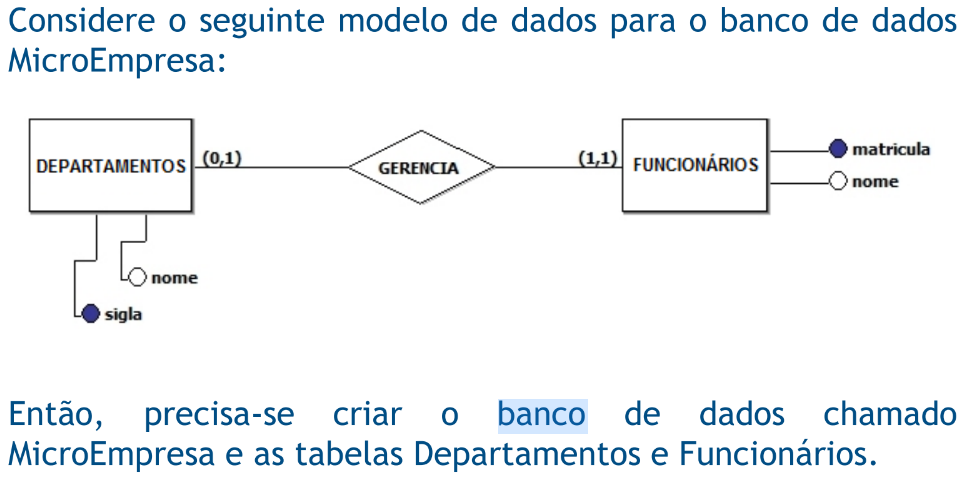
* Criando Tabelas
  + CREATE TABLE <Nome\_Tabela>;

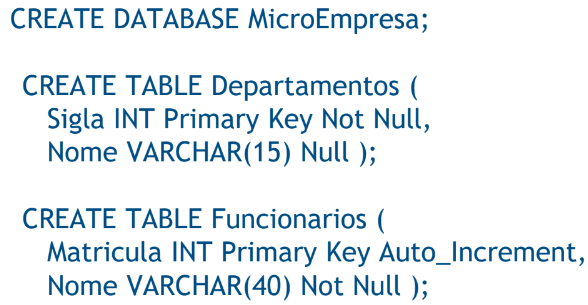
Permite ao usuário criar uma tabela no SGBD MySQL.

Exemplo:

Create Table Teste (

Id INT Primary Key Not Null Auto\_Increment,

Nome VARCHAR(40) Null );

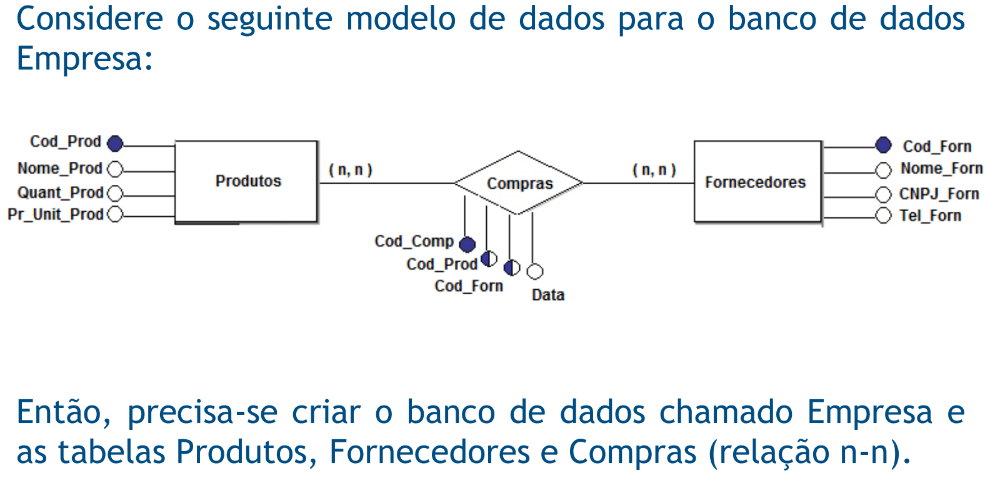


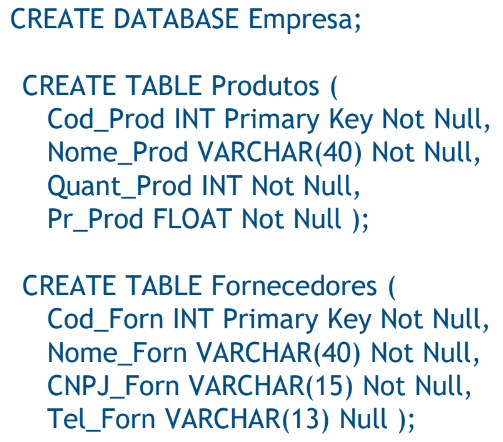
* Exibindo o Conteúdo da Estrutura da Tabela
  + DESCRIBE <Nome\_Tabela>;

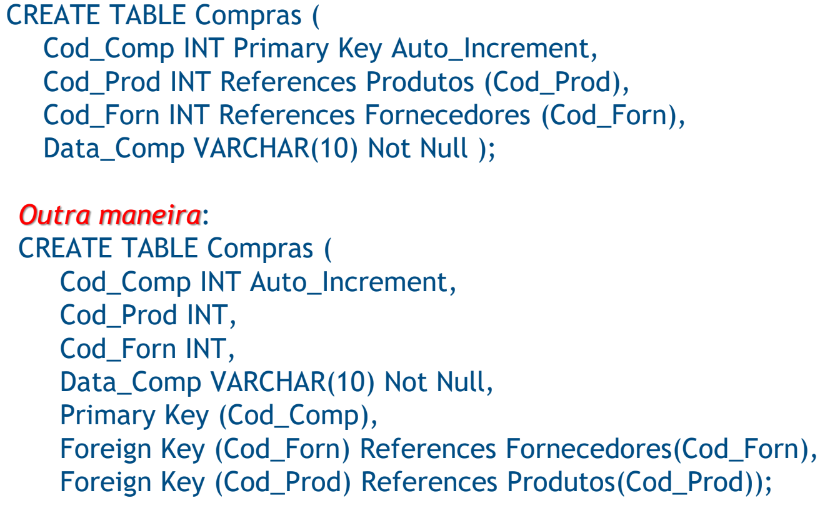
Permite ao usuário visualizar toda a estrutura (campos) da tabela em questão.

Exemplo:

DESCRIBE Departamentos;

- Mostra a estrutura da tabela Departamentos. 





* Alterando a Estrutura da Tabela
  + ALTER TABLE <Nome\_Tabela>;

Permite alterar/adicionar/remover campos da tabela.

Exemplo: Admitindo a necessidade de adicionar mais um campo na tabela MEDICO, celular VARCHAR(15), neste caso. A instrução ficaria assim:

ALTER TABLE MEDICO add celular VARCHAR(15);

Agora deseja-se, na mesma tabela MEDICO, excluir o campo idade. A instrução ficaria assim:

ALTER TABLE MEDICO drop idade;

O DROP também pode ser utilizado para apagar um banco de dados. Veja exemplo:

DROP DATABASE NOMEBANCO;

* Alterando o nome da Tabela
  + RENAME <TABELA>;

Instrução utilizada para renomear tabelas de um banco de dados.

Exemplo:

RENAME TABLE MEDICO TO MEDICOS;

* Apagando Todos os Registros da Tabela
  + TRUNCATE <TABELA>;

Instrução utilizada apagar todas as tuplas de uma tabela.

Exemplo:

TRUNCATE MEDICOS;

Para definir cláusulas padrões para aceitação de acentuação de palavras [utf8] segundo default Português-Brasil:

Exemplo :

CREATE DATABASE Banco

default character set utf8

default collate utf8\_general\_ci;

* Inserindo Registros na Tabela:
  + INSERT INTO <Tabela>

(campo1, campo2, ..., campon)

VALUES

(‘valor1’, ‘valor2’, ...,‘valorn’);

Exemplo:

INSERT INTO Clientes

(cod\_cli, nome, cpf, dt\_nasc, nacional, sexo, email, tel, cel)

VALUES

(‘11’, ‘Ana Lee’, ‘123234101’, ‘1995-06-22’, ‘Brasileira’, ‘F’, ‘lee.ana@gmail.com’, ‘(21) 98526-0221’, ‘(21) 3506-2009’);

Obs: Um campo AUTO\_INCREMENT não requer preenchimento de dados.

Exemplo:

INSERT INTO Contas

(tipo, categ, saldo)

VALUES

(‘Corrente’, ‘Especial’, ‘2598.55’);

Você pode inserir mais de um registro ao mesmo tempo, ou seja, num mesmo comando de inserção. Veja:

INSERT INTO Contas

(tipo, categ, saldo)

VALUES

(‘Corrente’, ‘Especial’, ‘2598.55’),

(‘Poupança’, ‘Comum’, ‘1050.38’),

(‘Poupança’, ‘Especial’, ‘2000.99’),

(‘Corrente’, ‘Comum’, ‘680.59’);

* Consultando Registros na Tabela:
  + SELECT \* FROM <Tabela>; (campo1, campo2, ..., campon)

Exemplo:

SELECT \* FROM Contas;

Nota: Podemos realizar diversos tipos de consultas, apresentando pequenas variações na instrução.

1. Consultar todas as tuplas da relação Contas, cuja categoria seja ‘Especial’:

SELECT \* FROM Contas WHERE categ=‘Especial’;

1. Consultar todas as tuplas da relação Contas, cujos saldos sejam maior que 1000 e menor que 3000.

SELECT \* FROM Contas WHERE saldo > 1000 and saldo < 3000;

1. Consultar todas as tuplas da relação Clientes, cujo sexo seja feminino ou cuja nacionalidade seja diferente de Brasileira.

SELECT \* FROM Clientes WHERE sexo=‘F’ or nacional<>‘Brasileira’

1. Selecionar nome, telefone e celular de todas as tuplas da relação Clientes, cujo sexo seja Masculino.

SELECT nome, tel, cel FROM Clientes WHERE sexo=‘M’;

1. Selecionar número da conta e o saldo das tuplas da relação Contas, cujo tipo seja Corrente e cuja categoria seja igual a Especial.

SELECT num\_conta, saldo FROM Contas WHERE tipo=‘Corrente’ and categ=‘Especial’;

1. Consultar todas as tuplas da relação Clientes, cujo nome inicie com a letra “A”:

SELECT \* FROM Clientes WHERE nome like ‘A%’;

1. Selecionar nome e telefone de todas as tuplas da relação Clientes ordenado ascendentemente pelo nome.

SELECT nome, tel FROM Clientes ORDER BY nome;

1. Selecionar nome, telefone e celular de todas as tuplas da relação Clientes, cujo sexo seja Feminino, ordenado descendentemente pelo nome.

SELECT nome, tel, cel FROM Clientes WHERE sexo=‘F’ ORDER BY nome DESC;

Mais Consultas SQL:

Situação: Selecionar código e nome das tuplas da tabela CLIENTES, data e valor das tuplas da tabela PEDIDOS onde o código da tabela CLIENTES seja igual ao COD\_CLI da tabela PEDIDOS.

SELECT A.codigo, A.nome, B.data, B.valor FROM CLIENTES A, PEDIDOS B WHERE A.codigo = B.cod\_cli;

Situação: Selecione código da tabela Produtos e o código da tabela Componentes, comparando linha a linha, tomando como base código do produto e código do Componente.

SELECT A.codigo, B.codigo FROM PRODUTOS as A INNER JOIN COMPONENTES as B ON (A.codigo = B.codigo);

Situação: Selecionar código e nome das tuplas da tabela CLIENTES, código e nome das tuplas da tabela FUNCIONARIOS buscando unificar o resultado.

SELECT codigo, nome FROM CLIENTES UNION SELECT codigo, nome FROM FUNCIONARIOS;

Situação: Verificar o número de tuplas da tabela CLIENTES tomando como base/referência o campo código.

SELECT COUNT(codigo) FROM CLIENTES;

* Alterando Dados do Registro:
  + Update <Nome\_Tabela>

Set col1=valor1, col2=valor2, ..., coln=valorn

Where <Condições>;

**Exemplo situação1**: Atualizar a cidade para “Niterói” e uf para “RJ”, do

registro cujo código seja 20, da tabela CLIENTES.

UPDATE CLIENTES SET cidade=‘Niteroi’, uf=‘RJ’ WHERE cod\_ref=20;

**Exemplo Situação2**: Atualizar o valor com 0.00, para os registros da tabela ITENS, cujo valor esteja NULO.

UPDATE ITENS SET valor=0.00 WHERE valor Is NULL;

**Exemplo Situação3**: Atualizar o campo ativo com “N”, da tabela VENDAS, onde a data de venda esteja compreendida entre 01/01/2023 e 30/04/2023.

UPDATE VENDAS SET ativo=‘N’ WHERE dt\_venda BETWEEN ‘01.01.2023’ And ‘30.04.2023’;

**Exemplo Situação4**: Situação: Atualizar o tipo para “H”, ativo para “S” e código do fornecedor para 100, onde o código de referência for 1, 5, 10 e 15.

UPDATE PRODUTOS SET tipo=‘H’, ativo=‘S’, cod\_forn=100 WHERE cod\_ref In (1, 5, 10, 15);

* Removendo Registros da Tabela:
  + Delete From <Nome\_Tabela> Where <Condição>;

**Exemplo Situação1:** Apagar da tabela CLIENTES todos os registros cujo

sexo seja igual a “M”.

DELETE FROM CLIENTES WHERE sexo=‘M’;

**Exemplo Situação2:** Apagar os registros da tabela PRODUTOS onde o tipo

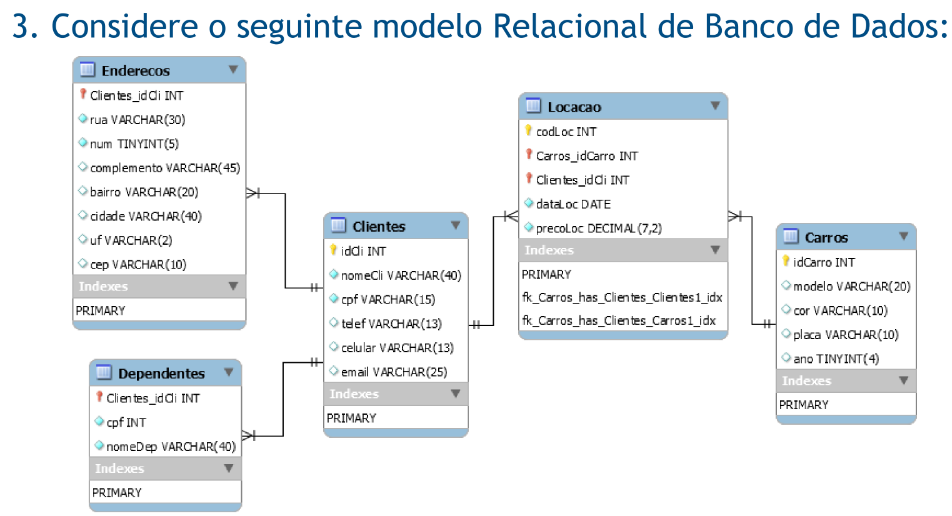
seja igual a 1.

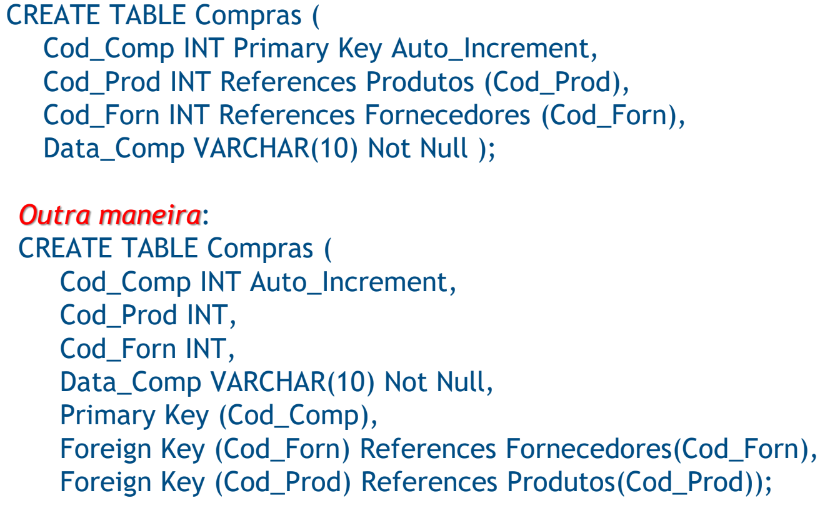
DELETE FROM PRODUTOS WHERE tipo=1;

**Exemplo Situação3**: Apagar os registros da tabela VENDAS onde a data de venda estiver compreendida entre 01/01/2021 e 30/06/2021.

DELETE FROM VENDAS WHERE data\_venda BETWEEN ‘01.01.2023’ And ‘30.04.2023’;

**Exemplo Situação4**: Apagar todos os registros da tabela DEVEDORES.

DELETE FROM DEVEDORES; 



Índices são utilizados para dinamizar as pesquisas SQL.

Também existem instruções avançadas da linguagem de consulta estruturada, ligadas, por exemplo, a linguagem de controle de dados, vinculadas a SQL DCL (Structure Query Language Data Control Language), que referem-se a comandos utilizados para controlar a parte de segurança do banco de dados.

Também, existem as instruções voltadas a SQL DTL (Structure Query Language Data Transaction Language), que referem-se as transações de bancos de dados em base universal, padrão SQL.

Neste contexto, transições SQL, considerando dependências funcionais, são melhores gerenciadas pelo DBA (Data Base Administrator), através de comandos SQL-DCL e SQL-DTL. Esses comandos são:

**SQL-DCL**: GRANT e REVOKE

**SQL-DTL**: BEGIN, COMMIT e ROLLBACK

* Atribuição de Privilégios:
  + GRANT [lista\_privilégios];

**Exemplo-Situação1**: Garantir acesso a um usuário de nome boente, sem privilégios.

GRANT USAGE ON \*.\*

TO boente@localhost

IDENTIFIED BY ‘1234’;

* + Para verificar:

SHOW GRANTS FOR boente@localhost;

**Exemplo-Situação2**: Dar privilégios globais a um usuário de nome pires.

GRANT ALL ON \*.\*

TO pires IDENTIFIED BY ‘1234’

WITH GRANT OPTION;

* + Para verificar:

SHOW GRANTS FOR pires;

**Exemplo-Situação3**: Dar privilégios específicos para a execução de

comandos DML em todas as tabelas do banco bd\_lib ao usuário

renata.

GRANT SELECT, INSERT, UPDATE, DELETE

ON bd\_lib

TO renata@localhost;

* + Para verificar:

SHOW GRANTS FOR renata@localhost;

**Exemplo-Situação4**: Dar privilégios de inserção e consulta para o usuário

juan, de todas as tabelas do banco db\_data.

GRANT INSERT, SELECT

ON db\_data

TO juan@localhost;

* + Para verificar:

SHOW GRANTS FOR juan@localhost;

* Revogação de Privilégios:
  + REVOKE [lista\_privilégios];

**Exemplo-Situação1**: Revogar os privilégios de exclusão no banco bc\_lib ao

usuário alfredo.

REVOKE DELETE

ON bc\_lib FROM alfredo@localhost;

* + Para verificar:

SHOW GRANTS FOR alfredo@localhost;

**Exemplo-Situação1**: Revogar todos os privilégios do usuário giullia, para todos os bancos de dados.

REVOKE ALL, GRANT OPTION

FROM giullia@localhost;

* + Para verificar:

SHOW GRANTS FOR giullia@localhost;

#### O que é uma Transação?

É uma unidade lógica de processamento que tem por objetivo preservar a integridade e a consistência dos dados, a partir de operações de transições, considerando suas eventuais dependências funcionais. Esse processamento pode ser executado todo ou não garantindo a atomicidade das informações.

A sintaxe básica de uma transação é:

Begin Transaction

-- Corpo de comando --

Commit ou Rollback

Nota: Commit (confirmação de um conjunto de comandos ou instruções) ou Rollback (desfaz todo o processo executado no corpo de comando caso tenha ocorrido algum evento indesejado), dependendo do caso.

1. Exemplo: @@ERROR:

BEGIN TRANSACTION

UPDATE FROM TbContas

SET NuSaldo= 10.000

WHERE NuSaldo < 50

IF @@ERROR = 0

COMMIT

ELSE

ROLLBACK

END

No exemplo 1, estamos iniciando uma transação para executar a mudança de saldo em determinadas contas onde, em caso de um ERRO, executará o comando ‘rollback’ para finalizar a transação e retornar os valores dos saldos, caso tudo ocorra sem erro, irá executar a transação e confirmando a alteração.

As vezes as mensagens de erro não são muito claras para os usuários e pensando nisso, procurando ajudar os desenvolvedores e administradores de sistema foi introduzido no SQL a função ‘RAISERROR’, que permite com que você personalize as mensagens de erro facilitando na hora de algum suporte. Vejamos o exemplo a seguir:

1. Exemplo RAISERROR;

BEGIN TRANSACTION

UPDATE FROM TbContas

SET NuSaldo= 10.000

WHERE NuSaldo < 50

IF @@ERROR = 0

COMMIT

ELSE

ROLLBACK

RAISERROR(5001,16,1,’ERRO NA ALTERACAO DO SALDO DA TBL TBCONTA’)

END

No exemplo 3, estamos criando uma transação que irá mudar todos os saldos de quem possui o valor menor que 50 para 10.000, onde estamos criando um SAVE POINT para garantir que as informações antigas sejam inseridas na tabela de log do sistema para garantir a integridade da informações, caso futuramente tenha sido feita alguma alteração que não deveria ocorrer.

Alguns autores chamam esse procedimento de “Oportunidade de Arrependimento”. Em outras palavras: mesmo após a operação ser realizada, você terá a possibilidade de acessar a tabela de log para visualizar o último registro antes da transação.