Linguagem SQL

Aula 02 – Arquitetura – Ambiente – Prévia DQL

Gustavo Bianchi Maia
gustavo.maia@faculdadeimpacta.com.br

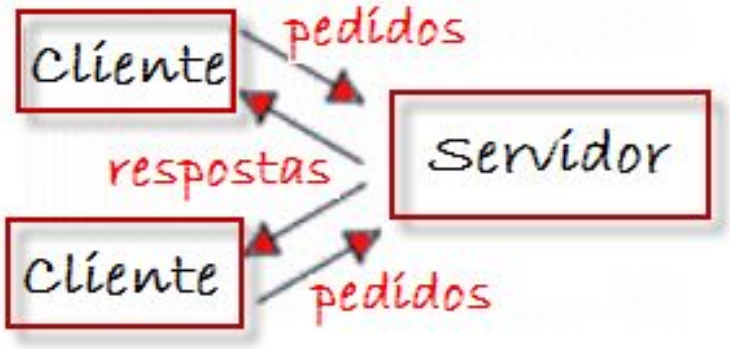
Agenda

- Arquitetura Cliente / Servidor
  - Definição, Camadas, Características, Vantagens e Desvantagens
- Overview do SQL Server Management Studio – SSMS
  - Conexão com o SGBD, tipos de serviços suportados, Object Explorer, Solution Explorer, Query, Informações Adicionais
- Prévia - Data Query Language – DQL
  - Estrutura do comando SELECT
  - Exemplos de SELECT básico
- Prática no SSMS

Arquitetura Client / Server

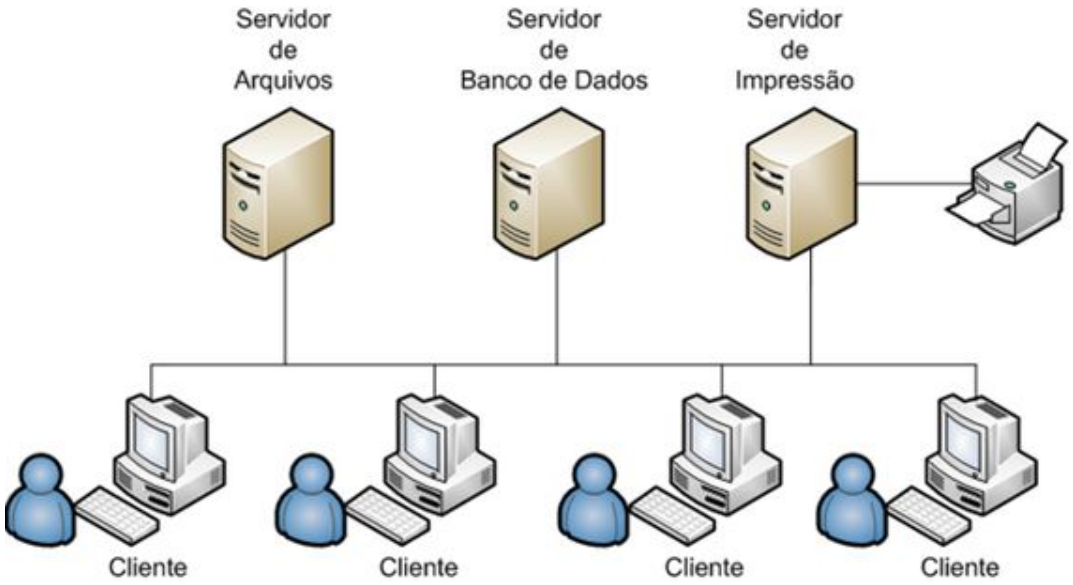
O modelo cliente-servidor foi desenvolvido na Xerox PARC durante os anos 70 e atualmente é o modelo predominante nas redes informáticas.

A arquitetura cliente-servidor, em computação, é uma estrutura de aplicação distribuída que atribui as tarefas e cargas de trabalho entre os fornecedores de um recurso ou serviço, designados como servidores, e os requerentes dos serviços, designados como clientes.



Arquitetura Client / Server

Serviços como *Banco de Dados*, *Email*, *World Wide Web*, *Redes de Impressão* entre milhares de outros serviços, são exemplos comuns deste modelo.



Geralmente os clientes e servidores comunicam através de uma rede de computadores em computadores distintos, mas tanto o cliente quanto o servidor podem residir no mesmo computador.

Arquitetura Client / Server

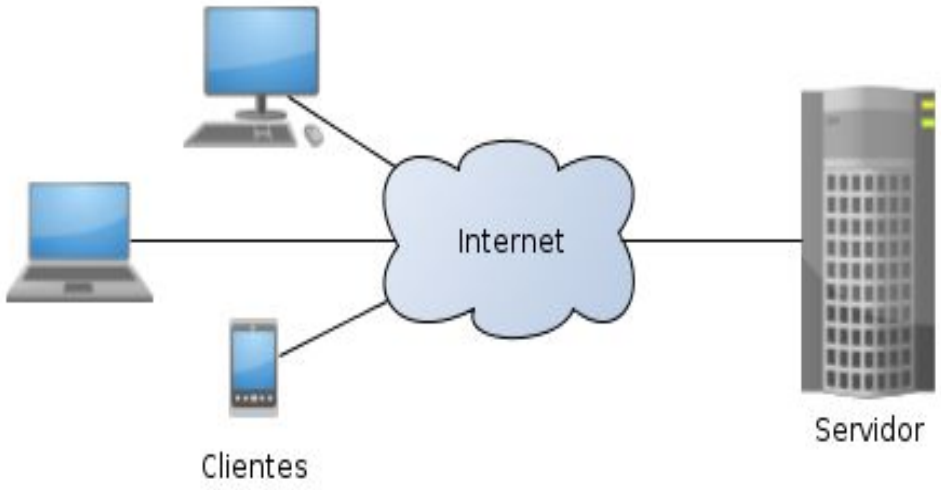
Os clientes iniciam sessões de comunicação com os servidores, que aguardam requisições de entrada, ou seja, a característica deste modelo descreve a relação de programas numa aplicação onde o componente de servidor fornece uma função ou serviço a um ou mais clientes, que iniciam os pedidos de serviço.

Tornou-se uma das ideias centrais de computação de rede e muitos aplicativos de negócios, escritos hoje, utilizam o modelo cliente-servidor. O termo também tem sido utilizado para distinguir a computação distribuída por computadores dispersos da computação monolítica centralizada em mainframe.

Mainframe Architecture

Arquitetura Client / Server

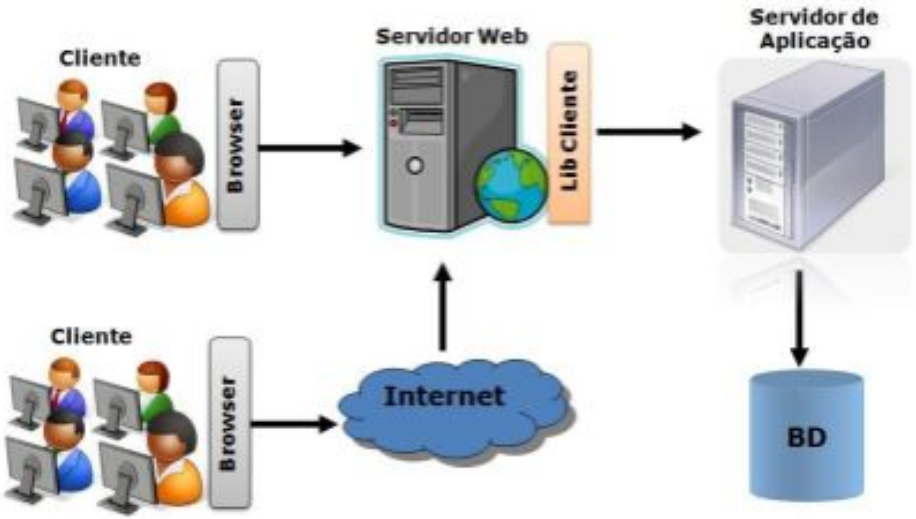
Por exemplo, um navegador web é um programa cliente, em execução no computador do usuário, que recebe informações armazenadas num servidor web na internet.



Um servidor é um host que está executando um ou mais serviços ou programas que compartilham recursos com os clientes.

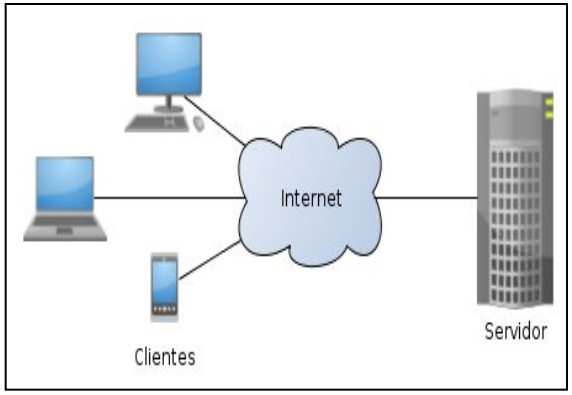
Arquitetura Client / Server

Usuários de serviços bancários usam um cliente web, de seu computador, para enviar uma solicitação para um servidor web num banco. Esse programa pode, por sua vez, encaminhar o pedido para o seu próprio programa de banco de dados do cliente que envia uma solicitação para um servidor de banco de dados noutro computador do banco para recuperar as informações da conta. O saldo é devolvido ao cliente de banco de dados do banco, que por sua vez, serve de volta ao cliente navegador exibindo os resultados para o usuário.

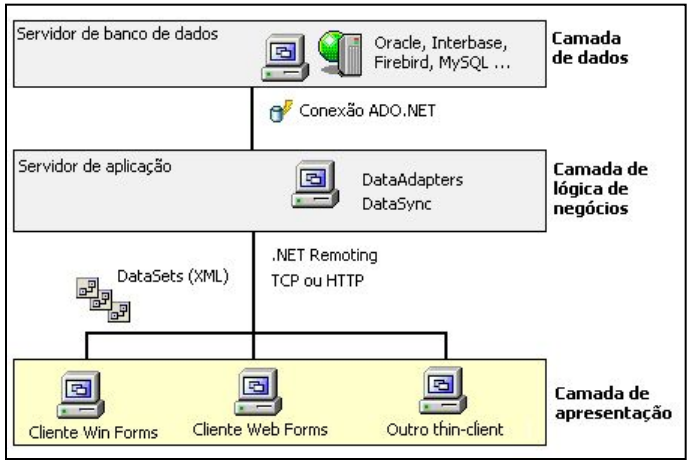


Arquitetura Client / Server

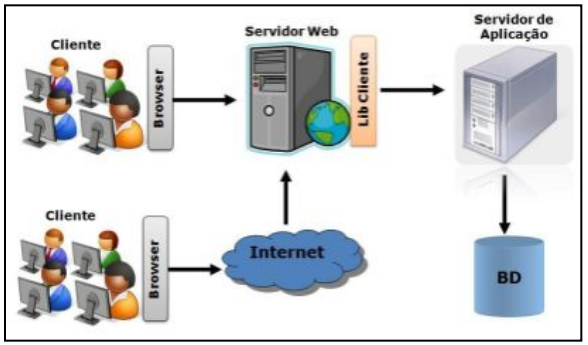
Após vários modelos estudados de cliente-servidor caracterizou-se chamar tecnicamente de arquitetura multicamada, inspirado nas camadas no *Modelo OSI*, o processo de dividir a arquitetura de cliente-servidor em várias camadas lógicas facilitando o processo de programação distribuída. Existe desde o modelo mais simples de duas camadas ou mais camadas. O mais utilizado atualmente que é o modelo de três camadas que é paralelo ao modelo de arquitetura de software denominado MVC.



Arquitetura de duas camadas



Arquitetura de três camadas



Arquitetura de quatro camadas

Arquitetura Client / Server

Características do Cliente:

- Inicia pedidos para servidores;
- Espera por respostas;
- Recebe respostas;
- Pode conectar-se vários servidores de uma só vez;
- Normalmente interage diretamente com os usuários finais através de qualquer interface com o usuário;
- Utiliza recursos da rede.

Características do Servidor:

- Sempre espera por um pedido de um cliente;
- Atende os pedidos e, em seguida, responde aos clientes com os dados solicitados;
- Pode se comunicar com outros servidores para atender uma solicitação específica do cliente;
- Fornece recursos de rede.
- Estrutura o sistema.

Arquitetura Client / Server

Vantagens:

- Permite que os papéis e responsabilidades de um sistema de computação possam ser distribuídos entre vários computadores independentes. Isso possibilita uma maior facilidade de manutenção. Por exemplo, é possível substituir, reparar, atualizar ou mesmo realocar um servidor, enquanto os clientes não são afetados;
- Todos os dados são armazenados nos servidores, que geralmente possuem controles de segurança muito maiores que a maioria dos clientes. Os servidores podem controlar melhor o acesso a recursos, para garantir que apenas os clientes com credenciais válidas possam receber e alterar os dados;
- Como o armazenamento de dados é centralizado , as atualizações dos dados são muito mais fáceis de administrar, em comparação com o paradigma P2P, onde atualizações de dados podem precisar ser distribuída e aplicada a cada ponto na rede. Esse time-consuming é passível de erro quando temos milhares ou mesmo milhões de pares;
- Muitas tecnologias avançadas estão disponíveis e foram projetadas para garantir a segurança, facilidade de interface do usuário e facilidade de uso;
- Funciona com vários clientes diferentes de capacidades diferentes.

Arquitetura Client / Server

Desvantagens:

- Clientes podem solicitar serviços, mas não podem oferecê-los para outros clientes, sobrecarregando o servidor, pois quanto mais clientes, mais informações que irão demandar mais banda;
- Um servidor poderá ficar sobrecarregado caso receba mais solicitações simultâneas dos clientes do que pode suportar;
- Este modelo não possui a robustez de uma rede baseada em P2P. Na arquitetura cliente-servidor, se um servidor crítico falha, os pedidos dos clientes não poderão ser cumpridos. Já nas redes P2P, os recursos são normalmente distribuídos entre vários nós. Mesmo se uma ou mais máquinas falharem no momento de download de um arquivo, por exemplo, as demais ainda terão os dados necessários para completar a referida operação.

Overview do SSMS

Podemos conectar com bancos de dados utilizando bibliotecas (*drivers*) de conexão, específicos de cada banco. Para estabelecer a conexão, precisamos também fornecer uma série de informações requeridas por esses *drivers*, como por exemplo, nomes do *host, user, password* e algumas vezes o *database* e object que queremos acessar.

A conexão com o banco feita desta forma é muito utilizada em linguagens de programação, onde precisamos interagir com o servidor de banco de dados, recebendo dados e disponibilizando em tela.

Outra forma de conectar com o banco é utilizar as ferramentas client fornecidas por cada fornecedor de banco de dados. Estes componentes facilitam o acesso e visualização dos objetos do servidor, permite várias atividades administrativas e facilita a utilização através de recursos gráficos.

Overview do SSMS

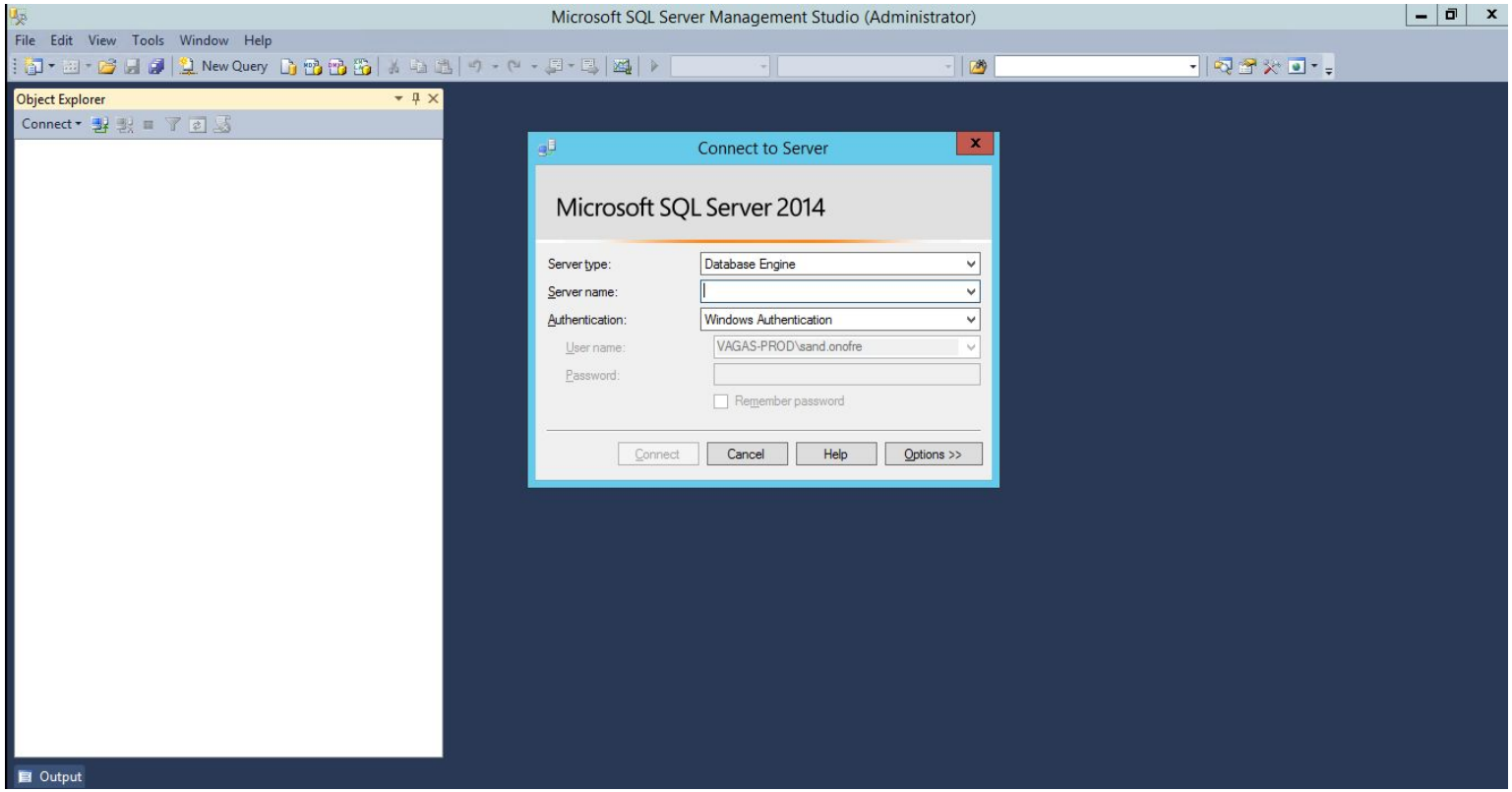
Cada fornecedor de banco de dados, pode oferecer vários componentes cliente para sua plataforma, além de podermos adquirir outras ferramentas de terceiros que fazem o mesmo.

Na plataforma que iremos utilizar, *Microsoft SQL Server,* temos vários componentes *client* como o *SQLCMD, SSMS, Power Shell, ...* A ferramenta mais intuitiva e de fácil utilização é o *SSMS*.

O *SQL Server Managment Studio* (*SSMS*) é o componente cliente que utilizamos para se conectar com o banco de dados SQL Server. Através desta ferramenta, conseguimos acesso aos serviços do servidor e várias outras funcionalidades.

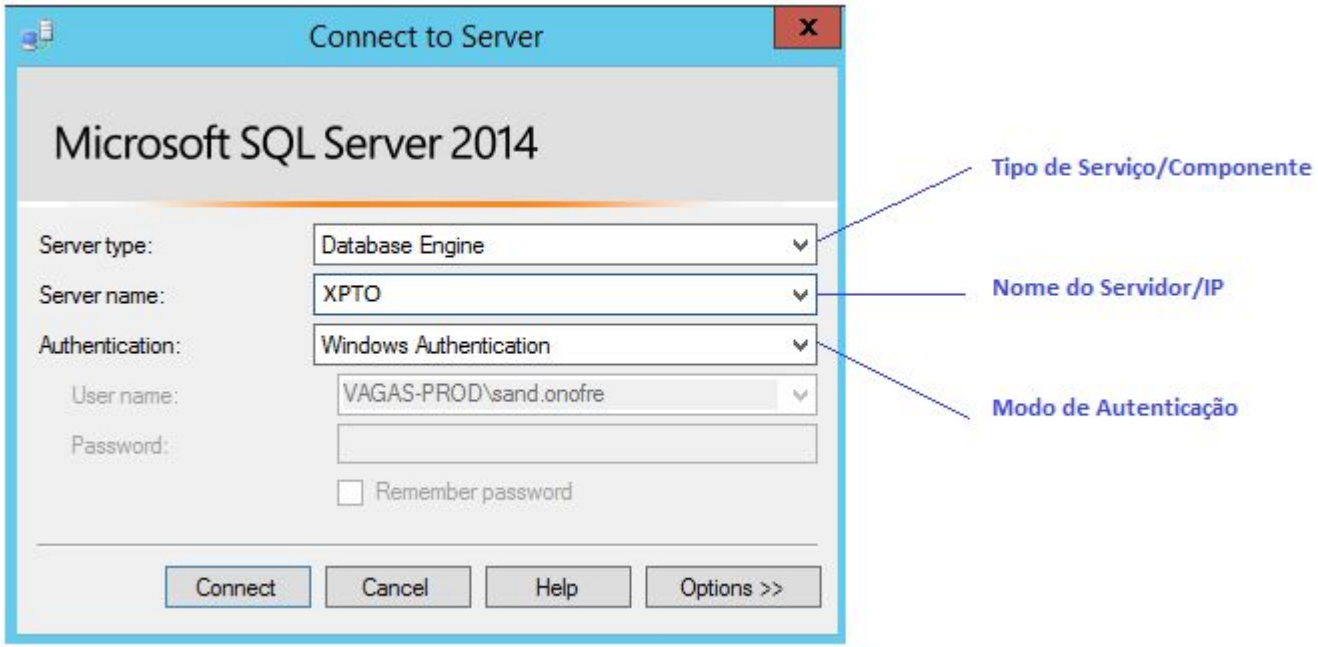
Overview do SSMS

Através do caminho *Iniciar, Microsoft SQL Server, SQL Server Management Studio*, iniciamos a ferramenta *Client*.



Overview do SSMS

Após a inicialização, precisamos fornecer informações em que componente queremos nos conectar e a forma de autenticação com o servidor.



Overview do SSMS

A maioria dos SGBDs possuem uma verificação irá validar se as credenciais do usuário que está tentando se conectar são reconhecidas pelo servidor.

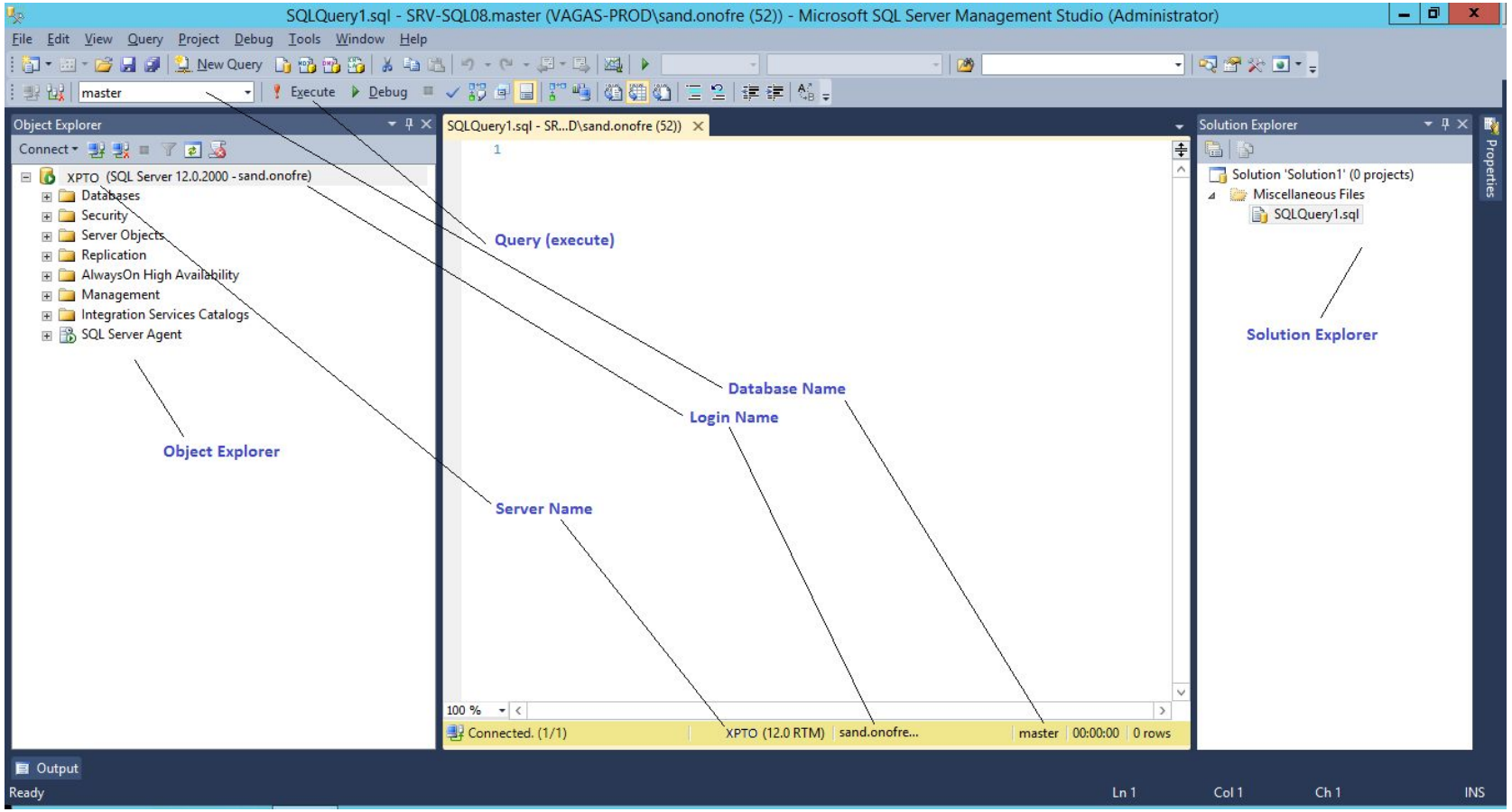
A princípio, nenhum usuário (além do administrador) possui permissão para se conectar ao servidor. O administrador do banco é o responsável por incluir, alterar ou excluir usuários que irão se conectar ao servidor.

Esse processo de verificação damos o nome de Autenticação. No SQL Server, temos dois modos possíveis de autenticação:

- Windows Authentication
- SQL Authentication

Overview do SSMS

A maioria dos SGBDs possuem uma verificação irá validar se as credenciais do usuário que está tentando se conectar são reconhecidas pelo servidor.



Data Query Language - DQL

Categoria de subcomando da linguagem SQL que envolve a declaração de recuperação de dados (SELECT).

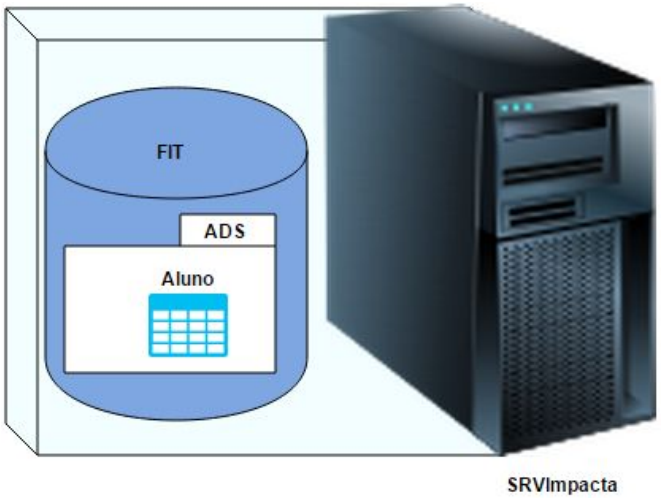
SELECT é uma declaração SQL que retorna um conjunto de resultados de registros de uma ou mais tabelas. Ela recupera zero ou mais linhas de uma ou mais tabelas-base, tabelas temporárias, funções ou visões em um banco de dados. Também retorna valores únicos de configurações do sistema de banco de dados ou de variáveis de usuários ou do sistema.

Na maioria das aplicações, SELECT é o comando mais utilizado. Como SQL é uma linguagem não procedural, consultas SELECT especificam um conjunto de resultados, mas não especificam como calculá-los, ou seja, a consulta em um "plano de consulta" é deixada para o sistema de banco de dados, mais especificamente para o otimizador de consulta, escolher a melhor maneira de retorno das informações que foram solicitadas.

DQL – Four Part Naming

Objetos (tables, views, functions, ...) no banco de dados possuem seu nome formado por 4 partes:

SELECT * FROM Server . Database . Schema . Object



SELECT * FROM SRVImpacta . FIT . ADS . Aluno

Caso de uso / Testes

Nos diversos ambientes, vamos tentar criar a seguinte estrutura.

Matricula	Nome
500	José
501	Pedro
502	Mario

idProva	Matricula	Nota
1	500	9
2	500	8
3	502	7
4	502	3
5	502	1

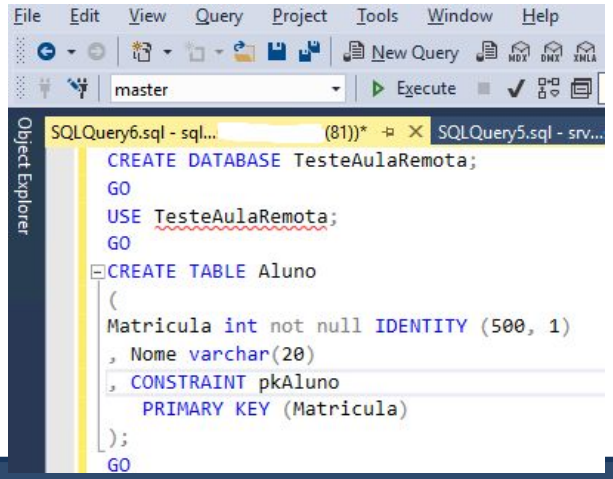
Atenção às alterações no código dependendo do banco / estrutura utilizada, ex:

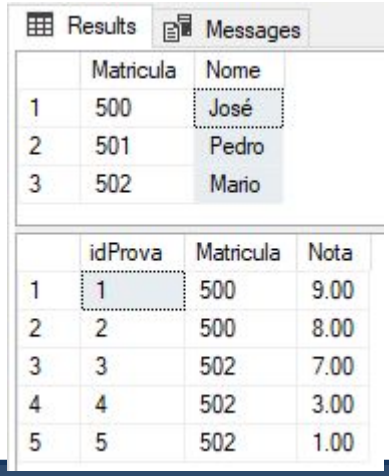
- Separador ‘GO‘ (só no MSSQL), ‘;’ é obrigatório no Mysql e Postgres
- Instrução para campos autoincrementais:
  - identity (MSSQL) = auto_increment (Mysql) = SERIAL (Postgres)
  - Alguns aplicativos 100% online usam SQLite e usam outra abordagem.

Caso de uso / Testes

```
CREATE DATABASE TesteAulaRemota;
GO
USE TesteAulaRemota;
GO
CREATE TABLE Aluno
        (
        Matricula int not null IDENTITY (500, 1)
        , Nome varchar(20)
        , CONSTRAINT pkAluno
          PRIMARY KEY (Matricula)
        );
GO
CREATE TABLE Prova
        (
        idProva int NOT NULL  IDENTITY (1, 1)
        , Matricula int  NOT NULL
        , Nota decimal(4,2) NOT NULL
        , CONSTRAINT pkProva PRIMARY KEY (idProva)
        , CONSTRAINT fkProva FOREIGN KEY (Matricula)
          REFERENCES Aluno(Matricula)
        );
```

```
INSERT INTO Aluno ( Nome ) values ( 'José' ), ( 'Pedro' ), ( 'Mario' );
GO
INSERT INTO Prova ( Matricula, Nota )
SELECT (SELECT Matricula FROM Aluno WHERE nome = 'José' ), 9
UNION ALL
SELECT (SELECT Matricula FROM Aluno WHERE nome = 'José' ), 8
UNION ALL
SELECT (SELECT Matricula FROM Aluno WHERE nome = 'Mario' ), 7
UNION ALL
SELECT (SELECT Matricula FROM Aluno WHERE nome = 'Mario' ), 3
UNION ALL
SELECT (SELECT Matricula FROM Aluno WHERE nome = 'Mario' ), 1;
GO
SELECT * FROM aluno;
SELECT * FROM Prova;
```





	Matricula	Nome
1	500	José
2	501	Pedro
3	502	Mario

	idProva	Matricula	Nota
1	1	500	9.00
2	2	500	8.00
3	3	502	7.00
4	4	502	3.00
5	5	502	1.00

DQL – Schema

O Schema é como se fosse um repositório onde colocamos objetos como tabela, visão, função, procedimento, ... Todo objeto possui um schema e quando não escolhemos algum, ele se encarrega de colocar o schema DBO como padrão (*default*).

Quando não escrevemos explicitamente os 4 nomes, o banco tenta preenche-los automaticamente:

```
SELECT * FROM ALUNO
```

Na consulta acima o banco de dados tenta encontrar o objeto Aluno. Como não foi fornecido o SERVER, ele usará o servidor a que está conectado no momento. O mesmo procedimento é feito para o DATABASE, como não foi fornecido, tenta utilizar a base que está conectada. Para o SCHEMA, se não foi mencionado, tentará o DBO.

O SELECT só irá funcionar se com todos os *defaults* tentados pelo banco, contenham o objeto ALUNO.

Data Query Language - DQL

Existem vários elementos na declaração SELECT, mas os principais são:

Elemento	Expressão	Descrição
SELECT	<lista de seleção>	Define quais as colunas que serão retornadas
FROM	<tabela de origem>	Define a(s) tabela(s) envolvidas na consulta
WHERE	<condição de pesquisa>	Filtra as linhas requeridas
GROUP BY	<agrupar a seleção>	Agrupa a lista requerida (utiliza colunas)
HAVING	<condição de agrupamento>	Filtra as linhas requeridas, pelo agrupamento
ORDER BY	<ordem da lista>	Ordena o retorno da lista

Data Query Language - DQL

A ordem como a consulta (query) é escrita, não significa que será a mesma ordem que o banco de dados utilizará para executar o processamento:

5: SELECT <select list>

1: FROM <table source>

2: WHERE <search condition>

3: GROUP BY <group by list>

4: HAVING <search condition>

6: ORDER BY <order by list>

Data Query Language - DQL

A forma mais simples da declaração SELECT é a utilização junto ao elemento FROM, conforme mostrado abaixo.

Note que no <select list> faz uma filtragem vertical, ou seja, retorna uma ou mais colunas de tabelas, mencionadas pela cláusula FROM.

Elemento	Expressão
SELECT	**<select list>**
FROM	**<table source>**

```
SELECT Nome, Sobrenome
FROM   Cliente;
```

Data Query Language - DQL

Outros exemplos para SELECT simples

(*) - Retorna todas as colunas da tabela *exemploSQL*

```
SELECT  * FROM exemploSQL
```

(coluna) - Retorna a coluna *texto_curto_naonulo* da tabela *exemploSQL*

```
SELECT  texto_curto_naonulo FROM exemploSQL
```

(coluna 1, coluna 2, ...) - Retorna as colunas texto_curto_naonulo e numero_check da tabela exemploSQL

```
SELECT  texto_curto_naonulo, numero_check FROM exemploSQL
```

Data Query Language - DQL

Podemos fazer utilização de diversos operadores matemáticos para cálculo de valores, abaixo mostramos os principais operadores.

Operator	Description
+	Add or concatenate
-	Subtract
*	Multiply
/	Divide
%	Modulo

```
SELECT preco, qtd, (preco * qtd)
FROM   DetalhesDoPedido;
```

OBS: Operadores possuem precedência entre si.

Data Query Language - DQL

Exemplos para SELECT simples e operadores

Retorna o resultado das operações abaixo

```
SELECT  20 + 20 / 5 FROM exemploSQL
SELECT  (20 + 20) / 5 FROM exemploSQL
SELECT  20 + (20 / 5) FROM exemploSQL

SELECT  ( (10+2) / 2 ) * 0.3 ) % 2

SELECT  Nome, Salario * 1.07 FROM Funcionario
```

Nota: O operador + se transforma em concatenador quando lidamos com string:

```
SELECT  'Hoje' + ' ' + 'é' + ' terça-feira ' + 'ou' + ' quinta-feira '
```

Data Query Language - DQL

Pode ser necessário darmos apelidos (Aliases) a colunas para facilitar o entendimento no retorno dos dados:

- **Apelidos na coluna utilizando a cláusula AS**

```
SELECT idPedido, preco, qtd AS Quantidade
FROM   DetalhesDoPedido;
```

- **Também é possível realizar a mesma operação com =**

```
SELECT idPedido, preco, Quantidade = qtd
FROM   DetalhesDoPedido;
```

- **Ou mesmo sem a necessidade do AS**

```
SELECT idPedido, preco ValorProduto
FROM   DetalhesDoPedido;
```

Data Query Language - DQL

Também pode ser necessário darmos apelidos em tabelas, principalmente quando formos realizar joins:

- **Apelidos em tabelas com a cláusula AS**

```
SELECT  idPedido, dataPedido
FROM     Pedido AS SO;
```

- **Table aliases without AS**

```
SELECT  idPedido, dataPedido
FROM     Pedido  SO;
```

- **Usando os apelidos no SELECT**

```
SELECT  SO.idPedido, SO.dataPedido
FROM     Pedido AS SO;
```

DQL – Cláusula WHERE

Operadores são utilizados para avaliar uma ou mais expressões que retornam os valores possíveis: TRUE, FALSE ou UNKNOWN.

O retorno de dados se dará em todas as tuplas onde a combinação das expressões retornarem TRUE.

Operadores de Comparação Escalar

=, <>, >, >=, <, <=, !=

```
SELECT FirstName, LastName, MiddleName
FROM Person.Person
WHERE ModifiedDate >= '20040101'
```

FirstName	LastName	MiddleName
Ken	Sánchez	J
Terri	Duffy	Lee
Roberto	Tamburello	NULL
Rob	Walters	NULL
Gail	Erickson	A

DQL – Cláusula WHERE

- **Operadores Lógicos são usados para combinar condições na declaração**

Retorna somente registros onde o primeiro nome for 'John' **E** o sobrenome for 'Smith'

```
WHERE FirstName = 'John' AND LastName = 'Smith'
```

Retorna todos as linhas onde o primeiro nome for 'John' **OU** todos onde o sobrenome for 'Smith'

```
WHERE FirstName = 'John' OR LastName = 'Smith'
```

Retorna todos as tuplas onde o primeiro nome for 'John' e o sobrenome **NÃO** for 'Smith'

```
WHERE FirstName = 'John' AND NOT LastName = 'Smith'
```

Data Query Language - DQL

Muitas vezes queremos visualizar apenas o retorno de algumas linhas e não necessariamente todos os registros de uma tabela. Podemos utilizar a cláusula TOP para isso:

TOP (N) / TOP (N) PERCENT

Retorna uma certa quantidade de linhas (ou percentual de linhas) definido.

```
SELECT top 10 * FROM exemploSQL
SELECT top 10 percent * FROM exemploSQL
```

Prática no SSMS - Seguir as instruções do professor.

Obtendo informações sobre uma tabela:

SP_HELP <nome da tabela>

OU selecione o nome da tabela e utilize a tecla de atalho ALT+F1

Obrigado !

Gustavo Bianchi Maia
gustavo.maia@faculdadeimpacta.com.br