COMANDOS PARA CONSULTA SQL NO SQL

1 – Introdução ao SQL:

SQL (Structured Query Language – Linguagem de Consulta Estruturada) é uma linguagem desenvolvida para permitir que qualquer pessoa, mesmo não sendo um programador, realize uma consulta (pesquisa) a um banco de dados, para isso, essa linguagem se aproxima muito da linguagem natural dos seres humanos. Como teremos oportunidade de ver, as instruções SQL permitem a realização de consultas complexas por meio de comandos que são praticamente frases ditas no idioma inglês.

2 - Breve Histórico:

Como já sabemos, os bancos de dados relacionais surgiram a partir de um artigo publicado em 1970 pelo matemático da IBM E. F. Codd. A partir deste trabalho, diversos outros pesquisadores empreenderam estudos com o objetivo de desenvolver sistemas com base no modelo relacional proposto por Codd. Foi então que em 1974 outros dois pesquisadores da IBM, D. D. Chamberlin e Boyce , publicaram o artigo intitulado "SEQUEL: A Structured English Query Language" -*Uma Linguagem de Consulta Estruturada em Inglês*- ao mesmo tempo em que apresentaram um protótipo da IBM denominado SEQUEL-XRM; nasce então a linguagem SEQUEL. No início da década de 80, por motivos jurídicos, a linguagem SEQUEL passa a se chamar SQL. Devido a esse fato histórico, o termo *SQL* pode ser pronunciado tanto soletrando-se suas letras quanto pronunciando-se a palavra SEQUEL.

3 - SQL como padrão para acesso a Banco de Dados Relacionais:

Desde suas primeiras implementações, a SQL alcançou enorme sucesso devido ao equilíbrio entre facilidade de uso e poder de recursos. Com isso, uma grande quantidade de produtos comerciais destinados ao gerenciamento de Banco de Dados (SGDB) foi tendo a SQL como linguagem básica de acesso aos dados. Dentre esses produtos, podemos citar o *Oracle* da Oracle Corporation, o *DB2* da IBM, o *RDB* da Digital, o *SYBASE* da Sybase INC, e o *MS SQL Server* da Microsoft, entre outros. Desse modo, a SQL tornou-se um padrão de fato no ambiente de Banco de Dados Relacional; faltava ainda torna-la um padrão de direito. Foi então que em 1986, a linguagem SQL foi definida como norma ANSI - X3.135. No ano seguinte, o padrão ANSI foi aceito como padrão internacional pela ISO. Isso significa que desde 1987, qualquer Sistema Gerenciador de Banco de Dados Relacional (SGDBR) deve incorporar, pelo menos, a linguagem SQL como meio de acesso aos dados, daí a enorme importância do aprendizado dessa linguagem.

Glossário1:(Baseado no original http://guiadohardware.tcinet.com.br/dicionario/index.asp)

ANSI - American National Standards Institute, uma associação voluntária, formada por mais de 1.300 membros, entre eles várias grandes companhias. A ANSI se encarrega de estabelecer padrões para a indústria, compatibilizando linguagens de programação, protocolos de rede, especificações elétricas de vários componentes, etc. Para fins de comparação, a ANSI é para os EUA o mesmo que a ABNT é para o Brasil.

4 – ANSI SQL versus SQL dos produtos comerciais:

Como sabemos, o objetivo dos padrões é servirem de referência para que produtos criados por empresas diferentes sejam compatíveis entre si. Assim, se todos os fabricantes de SGDB's (Oracle, IBM, MS, etc) seguissem integralmente as diretrizes indicadas no padrão ANSI SQL (mesmos recursos e mesma sintaxe dos comandos), teríamos uma compatibilidade total entre os vários produtos comerciais existentes, ou seja, uma mesma instrução SQL poderia ser igualmente executada em qualquer SGDB. Entretanto, o que ocorre é que cada fabricante incorpora, a seu modo, vários outros recursos além dos especificados pelo padrão ANSI. Se por um lado isso gera produtos mais poderosos do que se poderia obter pelo padrão ANSI, por outro lado faz com que existam vários dialetos da linguagem SQL, não havendo, portanto, uma compatibilidade total entre os vários SGDB's comerciais. Além de incorporarem recursos que não estão descritos no padrão ANSI SQL, também pode ocorrer de um determinado produto não seguir as regras sintáticas do padrão. Isso contribui ainda mais para diminuir a compatibilidade entre os produtos. Como exemplo, façamos uma rápida comparação entre o ANSI SQL e o Microsoft Jet SQL (SQL do Access). A tabela abaixo mostra os caracteres curinga do operador *Like* segundo o padrão ANSI SQL e o que foi implementado no produto Access da Microsoft.

Tabela 1 – Comparação entre ANSI SQL e o MJ SQL

Máscara para	Microsoft Jet SQL	ANSI SQL
Quaisquer caracteres simples	?	_ (underscore)
Nenhum ou vários caracteres	*	% (Porcentagem)

Não obstante as pequenas diferenças entre os vários dialetos da SQL, o domínio dessa linguagem em quaisquer de suas implementações permite migrar muito facilmente de um dialeto para o outro.

5 - A Linguagem SQL:

Como já foi dito anteriormente, o objetivo da linguagem SQL é permitir que qualquer pessoa, mesmo não sendo um programador, seja capaz de realizar CONSULTAS em um banco de dados. Entretanto, isso não é toda a verdade, de fato, a SQL possui muitas outras capacidades além da consulta a banco de dados. Mais exatamente, podemos subdividir os comandos da SQL de acordo com o quadro abaixo:

5.1 Linguagem de Definição de Dados (DDL – Data Definition Language):

É formado pelos comandos que permitem a implementação do banco de dados, isto é, criação e eliminação de tabelas e índices, estabelecimento de relacionamentos e alteração da estrutura de uma tabela já existente. A Tab.**Erro! Indicador não definido.** mostra os vários serviços que a DDL deve prover e o respectivo comando do MJ SQL que executa o serviço em questão.

Tabela 2 – Serviços da DDL versus comandos SQL Access

Serviço	Comando SQL Access
Criar/deletar tabelas e índices	Create Table/Drop Table
Alterar a estrutura de uma tabela já existente.	Alt Table
Definir os relacionamentos entre as tabelas.	Constraint

Tabela 3 – Serviços da DDL versus comandos SQL Access – Continuação

Aplicar restrições sobre as tabelas (definição de chaves primárias, especificar se os valores de um campo devem ser únicos e especificar se um campo pode ter ou não um valor null).	Constraint

5.2 Linguagem de Manipulação de Dados (DML –Data Manipulation Languagem):

Também chamada de *Linguagem de Consulta*, a DML é formada pelos comandos utilizados para inclusão, exclusão, consulta e alteração das informações do Banco de Dados. A Tab.**Erro! Indicador não definido.** mostra os vários serviços que a DML deve prover, juntamente com os comandos MJ SQL que os implementam.

Tabela 4 – Serviços da DML versus Comandos SQL Access

Serviço	Comando SQL Access
Consultas (permite a realização de pesquisas na base de dados)	Select, Select into, Inner Join, Left Join, Right Join
Inserir novo registro	Insert, Insert into
Alterar um registro já existente	Update
Deletar um registro	Delete
Consultas parametrizadas (permite incluir variáveis dentro da cláusula Where. Com isso, pode-se facilmente mudar o critério de consulta bastando alterar a variável em questão)	Parameters
União de várias consultas independentes	Union

5.3 Controle de Acesso:

Protege os dados de manipulações não autorizadas.

6 - Integridade dos Dados:

A SQL possui recursos que permitem garantir a integridade dos dados, protegendo-os contra corrupções, inconsistências e falhas do sistema de computação.

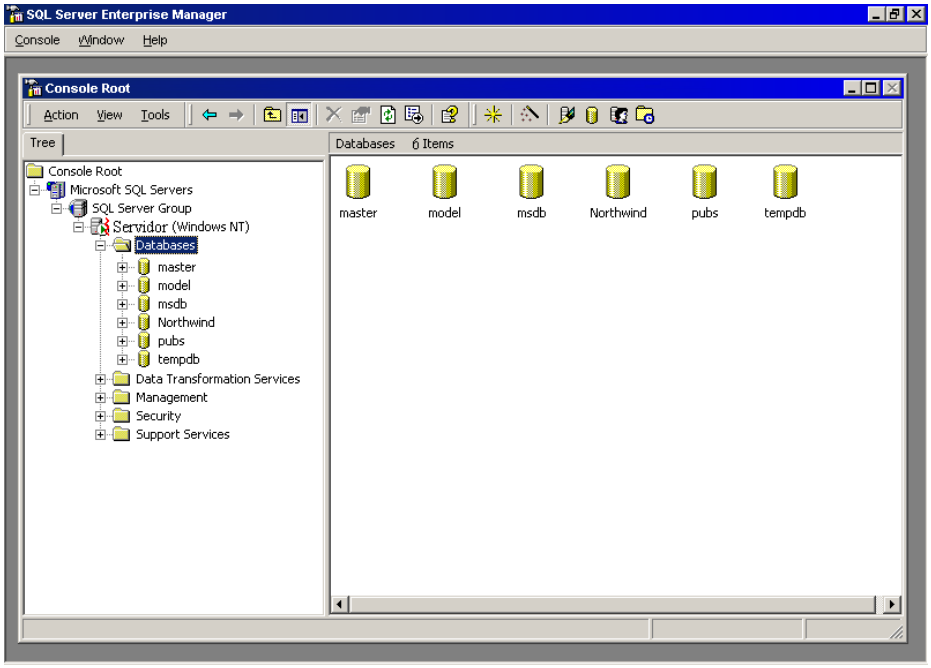
COMANDOS SQL

Sintaxe de um comando SQL:

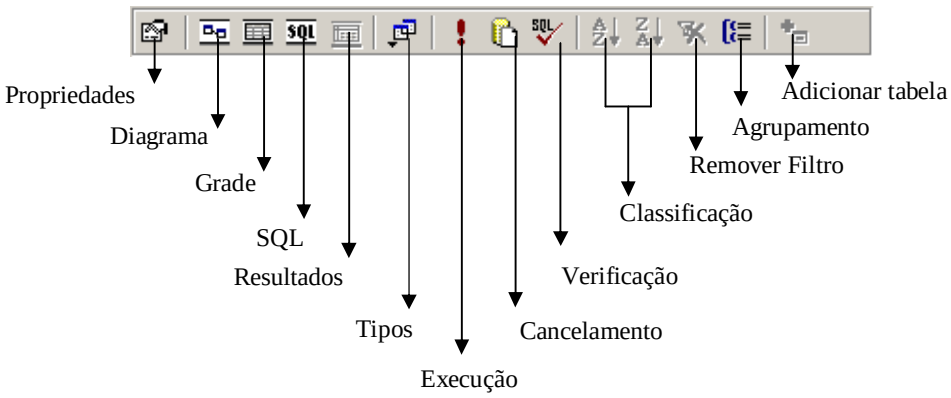
✓ **Instrução SELECT**
Objetivo: Selecionar registros de acordo com os critérios estabelecidos.
Sintaxe:
SELECT <campos> FROM <tabela> [WHERE = critério(s)]
[ORDER BY campo(s)] [GROUP BY campo(s)]

Acessando uma Tabela pelo SQL Server para Consulta SQL:

1. Acesse o **Enterprise Manager** do SQL Server;
2. Acesse a pasta **Databases** onde contém as bases de dados conforme a seguinte figura:

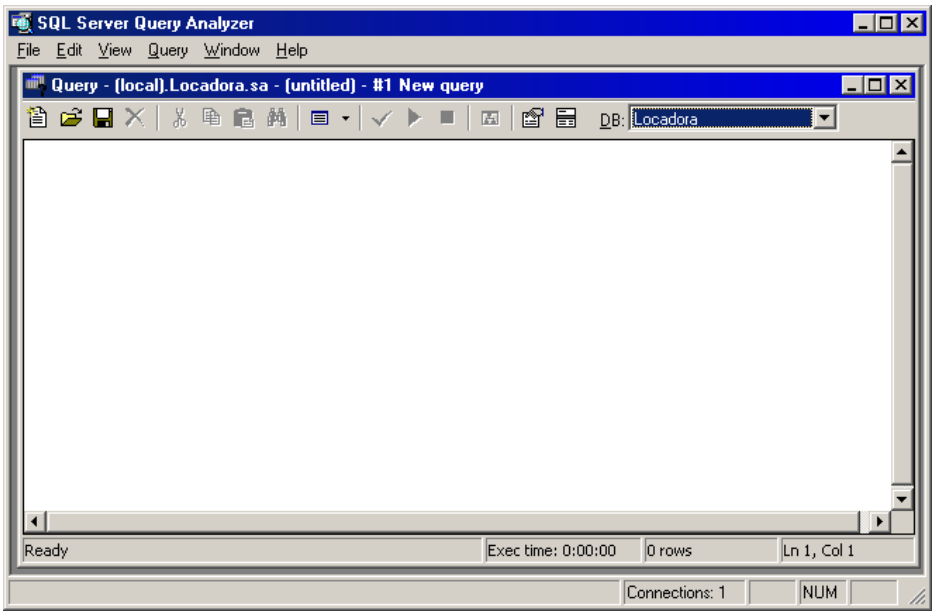


3. Selecione a base de dados **Locadora**, expanda a árvore e selecione **Tables**.
4. Click com o botão direito do mouse sobre qualquer uma das tabelas: **CLIENTES, DISCOS, FILMES, GAMES** ou **HISTORICO**, selecione a opção **Open Table / Return all rows** e uma nova janela será aberta. Conheça um pouco da sua barra de ferramentas:



5. Click agora no botão **SQL** da barra de ferramentas para que apareça a tela onde se digita os comandos SQL. Para executar o comando SQL pressione o botão **Execução**.

Uma outra maneira e bem mais apropriada de se realizar uma consulta SQL no SQL Server é através do **Query Analyzer**. Para acessa-lo vá em **Iniciar / Programas / Microsoft SQL Server 7.0** / **Query Analyzer.** Uma tela para que se escolha o computador (Servidor SQL) será mostrada. Escolha o nome do computador em que se encontra a base de dados **Locadora** e depois click em **Ok**. A seguinte tela será mostrada:



Selecione em **DB** a base de dados **Locadora**. Depois que construir a instrução SQL pressione a tecla **F5** para executar o comando.

Comandos mais utilizados em uma Consulta SQL:

Select * from CLIENTES ➔ A declaração em si : **SELECT**; as colunas a serem retornadas: ***** (asterisco) – significa todas as colunas; a cláusula **FROM** e o nome da tabela de origem dos dados, no caso, **CLIENTES**

Select CLCODIGO, CLNOME, CLBAIRRO from CLIENTES ➔ O resultado trará o código na primeira coluna, o nome na segunda, e assim por diante. Se voce quiser visualizar apenas o código, coloque apenas CLCODIGO nas colunas a serem selecionadas. As demais colunas existentes serão omitidas e não eliminadas da tabela durante a consulta.

Select Distinct CLCIDADE from CLIENTES ➔ A Tabela Clientes possui registros com Cidades Iguais. Se você solicitar apenas a coluna CLCIDADE, as linhas serão iguais. Para evitar essa repetição, basta incluir a cláusula DISTINCT logo após a declaração SELECT.

Select CLNOME, CLCEP from CLIENTES where CLCEP = '13330-000' ➔ Traduzindo a declaração: *Selecione as colunas CLNOME, CLCEP da tabela CLIENTES em que o valor de CLCEP seja igual a 13330-000.*

Exercício: Mostrar todos os clientes que moram no bairro Centro e que não tem telefone.

Select * from DISCOS where DIVALPAG > = 18 ➔ Serão mostrados todos os discos onde o valor correspondente seja maior ou igual a 18.

Exercício: Mostrar apenas os nomes dos discos cujo valor seja maior que 18 e menor e igual a 20

Select CLNOME, CLDATNAS from CLIENTES where CLDATNAS < = '01/30/1980' ➔ Serão mostradas as colonas CLNOME e CLDATNAS cujo valor das linhas da coluna CLDATNAS seja anterior a 30/01/1980. Repare que a data para essa consulta é informada da seguinte maneira: **mês/dia/ano** devido que o SQL Server utiliza este padrão de data.

Select CLDATNAS from CLIENTES Where Year(CLDATNAS) > 1983 ➔ Mostrará todas as linhas do campo CLDATNAS onde o ano seja maior que 1983. A função **Year** serve para retornar o ano de uma determinada data.

Select * from CLIENTES Where Month(CLDATNAS) = 06 ➔Mostrará todos os clientes que fazem aniversário no mês 6 por exemplo, ou seja, em Junho. A função **Month** retorna o mês de uma determinada data.

Select * from CLIENTES Where Day(CLDATNAS) = 10 ➔Mostrará todas as linhas da coluna CLDATNAS onde o dia for igual a 10. A função **Day** retorna o dia de uma determinada Data.

Exercício: -Mostrar quais são os clientes menores de 18 anos de idade.
- Mostrar todas as datas posteriores a 30/01/1990

Exercício: Mostrar o nome e data de nascimento dos clientes que nasceram no mês de Outubro de 1950

Select * from CLIENTES where CLCIDADE < > 'Indaiatuba' ➔ Serão mostradas todas as linhas cujo valor da coluna CLCIDADE seja diferente de *Indaiatuba.*

Select * from DISCOS where DIVALPAG Between 10 And 12 ➔ Será solicitada todas as linha cujo valor da coluna DIVALPAG esteja compreendido entre 10 e 12. Poderíamos utilizar também: **Select * from DISCOS where DIVALPAG >= 10 And DIVALPAG <=12**

Select * from CLIENTES where CLDATNAS Between '01/01/1943' And '12/31/1945' ➔ Mostrará todas as linhas cujo valor da coluna CLDATNAS esteja compreendido entre 01/01/1943 e 31/12/1945. Poderíamos utilizar também: **Select * from CLIENTES where Year (CLDATNAS) Between 1943 And 1945**

Exercício: -Mostrar os clientes nascidos no último bimestre de 1943

Select * from CLIENTES where CLDATNAS Not Between '01/01/1943' And '12/31/1945' ➔Selecionará todas as colunas da tabela CLIENTES em que o valor da coluna CLDATNAS **Não** esteja entre 01/01/1943 e 31/12/1945.

Exercício: -Mostrar os clientes que não nasceram entre 1970 e 1990

Select * from FILMES where FITEMDUR Is Null → O operador ***Is Null*** permite obter todas as linhas cujo valor de uma determinada coluna seja nulo (desconhecido).

Select * from FILMES where FITEMDUR Is Not Null → Obtendo o resultado contrário da declaração anterior: selecionará todas as linhas cujo valor da coluna FITEMDUR não seja nulo.

Exercício: -Mostrar todos os filmes cujo tempo de duração seja nulo e que o ano de produção não seja nulo.

Select * from FILMES where FIGENERO Like 'AVENTURA' → O operador ***Like*** é o mais apropriado para comparar valores de colunas do tipo texto, permitindo não especificar cadeias de texto idênticas, mas também semelhantes, pois existem alguns caracteres especiais que funcionam como uma espécie de máscara. Traduzindo a declaração: *Selecione todas as colunas da tabela FILMES em que o valor da coluna FIGENERO pareça com AVENTURA.* Poderíamos utilizar também: **Select * from FILMES where FIGENERO = 'AVENTURA'**

Exercício: -Mostrar quais são os filmes do gênero DESENHO e INFANTIL

Select * from CLIENTES where CLNOME Like 'JO%' → O caracter % equivale a seguinte condição: *qualquer que seja a seqüência*. Traduzindo a declaração: *Selecione todas as linhas da tabela CLIENTES em que o valor da coluna CLNOME comece com "JO", não importando o que venha depois.*

Exercício: -Mostrar todos os nomes de clientes que comecem com JO e terminem com A

OBS: *Um problema em potencial quando se trata de comparações envolvendo strings é o fato de o usuário acidentalmente digitar espaços antes ou depois dos dados de entrada. Por exemplo, ao invés de digitar "Penélope Charmosa", um usuário poderia digitar " Penélope Charmosa" (note que há um espaço em branco antes do nome). Se tal fato ocorrer, a consulta*

Select * From CLIENTES Where CLNOME Like Pe%'

Não incluirá o nome " Penélope Charmosa", pois esse nome inicia-se agora com um " P" (espaço seguido da letra "P") e não com as letras "Pe". Para evitarmos isso, devemos utilizar as funções LTRIM() e RTRIM(). A função LTRIM() remove os espaços iniciais (espaços à esquerda) e RTRIM() remove os espaços finais (espaços à direita). A consulta anterior pode ser refeita de modo a ignorar os espaços iniciais:

Select * From CLIENTES Where LTrim(CLNOME) Like 'Pe%'

Agora, mesmo que o usuário inserisse acidentalmente espaços durante a digitação, a consulta retornaria o resultado esperado, ou seja, incluiria no resultado a string " Penélope Charmosa".

Select * from CLIENTES where CLCIDADE Like '_.%' → O caracter "**_**" (underline) equivale a seguinte condição: *qualquer caractere único*. No caso, a declaração pede todas as linhas cuja a coluna CLCIDADE do CLIENTE tenha um ponto como segundo caracter no nome da cidade, não importando qual seja o primeiro e nem o que vem depois, por exemplo: S. Paulo. Caso não haja nenhum registro equivalente para a consulta, experimente incluir um em que o campo CLCIDADE possua o nome S. Paulo ou R. Janeiro

Exercício: -Mostrar todos os nomes de filmes que possuem Day.

Select * from FILMES where FIDATCOM > '01/01/1998' And FITEMDUR = '090' → Mostrará todos os filmes com a data de compra superior a 01/01/1998 e que o tempo de duração do filme seja igual a *090*.

Select * from CLIENTES where CLNOME Like 'M%' Or CLNOME Like 'F%' →Mostrará todas as linhas da tabela CLIENTES em que o valor da coluna CLNOME comece com *M* ou com *F*.

Select FICODIGO, FINOME from FILMES Order By FINOME → Selecionará todos as linhas das colunas FICODIGO e FINOME da tabela FILMES definindo a coluna FINOME como ordem de exibição.

Exercício: -Ordenar todos os filmes pelo nome, menos os que tem nome em branco

Exercício: -Mostrar os filmes diferentes que existem.

Select * from CLIENTES Order By CLNOME Desc → A palavra DESC logo após a coluna a ser ordenada serve para inverter a ordem da coluna CLNOME, ou seja, a coluna CLNOME será mostrada em ordem decrescente.

Select * from CLIENTES Order By CLCIDADE Desc, CLNOME → Selecionará todas as colunas da tabela CLIENTES ordenado pela CIDADE e pelo NOME sendo que a coluna CLCIDADE estará em ordem decrescente.

OBS: *A cláusula **ORDER BY** deve ser sempre a última de uma consulta e você não precisa necessariamente selecionar a coluna pela qual deseja ordenar os registros. Por exemplo, mesmo selecionando apenas os registros da coluna CLCODIGO com o comando: **Select CLCODIGO from CLIENTES** , poderíamos ordená-los por nome mesmo que os registros da coluna CLNOME não sejam selecionados. Veja o um exemplo: **Select CLCODIGO from CLIENTES Order By CLNOME**.*

Select CLCIDADE, CLNOME from CLIENTES Group By CLCIDADE, CLNOME → A cláusula *Group By* permite um agrupamento. No caso, serão listados os nomes de todos os clientes agrupados por cidade.

OBS: *A cláusula **Group By** permite fazer um agrupamento somente se as colunas solicitadas forem incluídas também na cláusula, caso contrário, a execução da consulta retorna um erro. Entretanto, não é necessário agrupar as colunas na mesma ordem em que as selecionar.*

Exercício: Mostrar o nome de todos os filmes agrupados pelo ano de produção e pelo gênero, desde que o ano de produção seja diferente de nulo:

Desta forma aparecerão, por exemplo, todos os filmes de ação produzidos em 1990, seguidos por todos os filmes de ação produzidos em 1991 e assim por diante.

Select Avg(DIVALPAG) from DISCOS → A função **AVG** retorna a média dos valores, sendo que estes devem ser numéricos ou datas. No caso será mostrada a média dos registros da coluna DIVALPAG da tabela DISCOS.

Select avg(DIVALPAG) as Media from DISCOS → retorna o mesmo resultado da função anterior, exceto o nome da coluna que será nomeada como **Media**.

Select Convert(decimal(10,2), avg(DIVALPAG)) as Media from DISCOS → mostra a média formatada com 10 casas sendo duas decimais. ***Obs****. Todas as formas de conversão encontram-se no Books on-line do SQL-Server.*

Select Min (CLDATNAS) As Mais_Velho, Max (CLDATNAS) As Mais_Novo From CLIENTES → As funções Max e Min retornam, respectivamente, o maior e o menor valor do argumento. No caso, mostrará a pessoa mais nova e a pessoa mais velha cadastrada na tabela. Para melhorar o resultado, foram adicionados títulos às colunas com a cláusula **AS**.

Select Sum (DIVALPAG) As Soma from DISCOS → A função **SUM** retorna o somatório dos valores correspondentes aos argumentos, como o exemplo, a somatória dos valores de DIVALPAG da tabela DISCOS.

Select Upper(CLNOME) From CLIENTES → A função **Upper** é utilizada para alterar uma string para letras maiúsculas, ou seja, o resultado será todos os nomes de clientes em letras maiúsculas. Pode-se utilizar também a função **Lower( )** para fornecer o resultado em letras minúsculas.

Select CLNOME, DATEDIFF(yy, CLDATNAS, GETDATE( )) As Idade From CLIENTES → A função **DATEDIFF** é utilizada para mostrar a diferença entre duas datas. Os argumentos são: o tipo de retorno desejado (veja tabela seguinte), a data mais antiga e a data mais recente. O script apresentado mostrará o nome e a idade de cada cliente. Essa diferença entre duas datas pode ser retornada em vários formatos, no caso a diferença é mostrada em anos (**yy**). **GETDATE( )** é utilizado para retornar a data atual. Os possíveis formatos de retorno entre duas datas são:

Retorno	Valor a Usar
Dia	Dd
dia de ano	Dy
Hora	Hh
milissegundo	Ms
minuto	Mi
mês	Mm
quarto	Oq
segundo	Siga os procedimentos seguintes:
semana	Wk
Dia da semana	Dw
ano	Yy

Exercício: Apresentar a média de idade de todos os clientes.

Select SUBSTRING(CLNOME, 1, 10) From CLIENTES → A função SUBSTRING é utilizada para retornar uma determinada faixa especificada de uma string. Os parâmetros usados são: a string que será manipulada, um índice para o caractere inicial e a quantidade de caracteres desejados. No

script apresentado, a função é utilizada para retornar os dez primeiros caracteres do nome dos clientes, a partir da posição do caracter 1 retorne os próximos 10 caracteres.

Exercício: Apresentar o nome dos clientes e as três primeiras letras da cidade onde ele mora.

Select CLNOME, len(CLNOME) From CLIENTES → a função len retorna a quantidade de caracteres de uma String. Esse script retorna o nome do cliente e a quantidade de caracteres existentes em cada nome.

Exercício: Apresentar todos os clientes classificados pelo comprimento do nome e pelo nome.

Select CLNOME as Nome, RIGHT(CLNOME, 5) as Cinco_Últimas From CLIENTES → A função RIGHT pega um determinado numero de letras da direita para a esquerda de uma determinada string. Pode-se usar também a função **LEFT** que tem o mesmo princípio, porém retorna os caracteres da esquerda para a direita. Neste caso a função foi utilizada para mostrar as cinco últimas letras do nome dos clientes.

Select REVERSE(CLNOME) FROM CLIENTES → A função REVERSE é utilizada para inverter uma determinada string.

Select CLNOME, REPLACE(CLCIDADE, 'INDAIATUBA', 'INDAIATUBANO') From CLIENTES → A função REPLACE é utilizada para substituir uma determinada string por outra. Este exemplo diz o seguinte: substitua em CLNOME todas as ocorrências “INDAIATUBA” por “INDAIATUBANO”. A função replace em conjunto com a função Select atua apenas em memória.

Select DINOME, DIVALPAG, FLOOR(DIVALPAG) From DISCOS → A função FLOOR é utilizada para arredondar um determinado valor para o seu inteiro anterior, ou seja, 12,8 fica 12.0.

Select DINOME, DIVALPAG, CEILING(DIVALPAG) From DISCOS → A função CEILING é utilizada para arredondar um determinado valor para o seu inteiro posterior, ou seja, 12,8 fica 13.0.

Select DINOME, POWER(FLOOR(DIVALPAG),2) From DISCOS → A função POWER é utilizada para calcular a potência de um determinado valor. Note que logo após DIVALPAG é informado o número **2**, ou seja, DIVALPAG será elevado a potência **2** ou elevado ao quadrado. A função FLOOR foi utilizada apenas para que o valor de retorno seja inteiro.

Select DINOME, SQRT(DIVALPAG) From DISCOS → A função SQRT é utilizada para retornar a raiz quadrada de um determinado número. No caso serão mostradas as raízes quadradas de todos os valores de discos.

Select DIFORNEC, DINOME, DIVALPAG From DISCOS Order By DIFORNEC Compute Sum(DIVALPAG) BY DIFORNEC → A função COMPUTE é utilizada para produzir uma nova linha para um certo agrupamento de dados. Esta linha pode mostrar, por exemplo, a soma dos valores de um determinado grupo. Este exemplo faz um agrupamento pelo fornecedor e ao final de cada grupo de fornecedor informa o valor da somatória dos discos.

Select Count(FINOME) From FILMES → conta todos os filmes que possuem um nome, isto é, conta todos os que forem diferentes de nulo.
Select Count(FIGENERO) From FILMES→ conta todos os filmes que possuem um gênero.

Select Count(FIANOPRO) From FILMES→ conta todos os filmes que possuem um ano de produção.
Select Count(DISTINCT FINOME) From FILMES → Conta todos os filmes sem repetição.

Select Count(*) As Total_Registros from CLIENTES → Conta todos os registros existentes, desprezando se existem valores nulos.

Exercício: Apresentar a quantidade de discos com valor superior a 18.

Select CLCODIGO, CLNOME from CLIENTES Union Select FICODIGO,FINOME from FILMES → A instrução **UNION** serve para se unir duas tabelas. No caso, serão mostrados dois campos sendo que um terá os códigos das pessoas e dos filmes, e outro os nomes das pessoas e dos filmes.

Exercício: - Quais os nomes de filmes e games que começam com a letra "x"?

Select Top 3 DINOME,DIVALPAG From DISCOS Order By DIVALPAG Desc → O comando **TOP** é utilizado para exibir apenas alguns registros superiores ou inferiores de um conjunto de registros. Nas consultas, a palavra TOP é combinada com a ordem de classificação (Order By). Nesse script são mostrados os três preços superiores da tabela de DISCOS. Se desejar retornar os últimos registros de uma tabela, basta classificar na ordem ascendente (do menor para o maior, ou seja, retirando o comando Desc).

OBS: *Por que as vezes a consulta retorna quatro registros quando você pediu especificamente 3 ? Não há garantias de que apenas 3 registros serão retornados em uma consulta TOP 3. É possível que nenhum, um ou dois registros sejam retornados se sua tabela tiver apenas esses registros. E se dois ou mais registros forem iguais no último lugar em sua lista de resultados, é possível que quatro ou mais registros sejam retornados.*

Exercício: Apresentar os 10 discos mais caros sem realizar a repetição de nomes.

Select Top 5 Percent DINOME, DIVALPAG from DISCOS Order By DIVALPAG Desc → retornará 5% dos discos mais caros da tabela de DISCOS.

OPERADORES ARITMÉTICOS:

Podemos utilizar qualquer um dos operadores aritméticos em declarações SQL: soma (+), subtração (-), multiplicação (*), e divisão (/).

Select HIDATLOC + 30 from HISTORICO → Esse é um cálculo envolvendo data, onde serão acrescentados 30 dias nas linhas da coluna HIDATLOC (Data de Locação) da tabela HISTORICO.

Select DIVALPAG * 1.15 from DISCOS → As linhas da coluna DIVALPAG (Valor do Disco) da tabela DISCOS serão multiplicadas por 1.15 ,ou seja, haverá um acréscimo de 15% sobre seus valores.

OBS: *Se desejar substituir o título de uma coluna por um mais apropriado, basta utilizar a cláusula **AS** como abaixo:*
Select DIVALPAG * 1.15 As Acréscimo from DISCOS

SUBCONSULTAS:

As subconsultas geralmente são usadas quando o resultado de uma consulta é proveniente de mais de uma tabela. Imagine a seguinte necessidade: Uma consulta que retorne o nome de um cliente que locou um filme num determinado dia, mas você só sabe apenas o código do filme e a data de locação, não sabendo então nenhum dado do cliente. Vamos supor que o código do filme locado pelo cliente desconhecido é "02300" e este foi locado no dia 21/01/1995. A tabela do histórico das locações efetuadas não possui os dados referentes ao cliente que locou o filme, mas possui o seu código. Analise e execute a seguinte declaração:

Select HICODSOC from HISTORICO where HIDATLOC = '01/21/1995' and HICODFIL = '02300' → Essa consulta retornará o código do cliente que locou o filme de código "02300" no dia 21/01/1995. Sobre esse resultado é possível realizar uma nova consulta que lhe retorne o nome do cliente:

Select CLNOME from CLIENTES where CLCODIGO = ?(não se sabe o código do cliente que se deseja consultar)

Agora, utilizando a subconsulta:
Select CLNOME from CLIENTES where CLCODIGO = (Select HICODSOC from HISTORICO where HIDATLOC = '01/21/1995' and HICODFIL = '02300')

Obs.: É importante observar que o Select mais interno, dentro dos parênteses, é executado toda vez que o Select mais externo muda de linha.

Agora apenas um exemplo caso queira utilizar o comando **Not** em uma Subconsulta: Suponha que se queira o contrário, ou seja, o nome de todos os clientes que não locaram esse filme neste dia. Veja o exemplo:
Select CLNOME from CLIENTES where Not CLCODIGO = (Select HICODSOC from HISTORICO where HIDATLOC = '01/21/1995')

OBS: *Quando uma subconsulta resultar em mais de um valor, deve-se utilizar **IN** em vez do sinal de =. O operador **IN** permite selecionar os registros que estejam contidos em uma lista de valores. No caso, quem fornece tal lista, é uma subconsulta. Exemplo:*
Select * from CLIENTES where CLCODIGO IN (Select HICODSOC from HISTORICO where HIDATLOC = '01/21/1995' and HICODFIL <> '02300')

Exercício: - Mostrar apenas os nomes dos filmes locados entre os anos de 1995 e 1997 ordenados pelo nome.

Exercício: - Mostrar o nome de todos os clientes que locaram o filme '01369'**.**

Outros exemplos de subconsultas:

Uma subconsulta executada sobre uma mesma tabela:

Select DINOME, Convert(money, DIVALPAG) From DISCOS Where DIVALPAG < (Select Avg(DIVALPAG) From DISCOS) And DIVALPAG <> 0 Order By DIVALPAG => Mostrará os discos com valores inferiores a média e diferentes de zero, ordenados pelo valor do disco.

Select CLCODIGO, CLNOME, CLTELEFO From CLIENTES Where (Select Count(*) From HISTORICO Where HICODSOC=CLCODIGO) >=300 Order By CLNOME => mostrará quais os clientes que locaram mais de 300 vezes. Essa consulta consome um tempo considerável de processamento, uma vez que para cada cliente é verificada a quantidade de filmes que ele locou.

Select CLCODIGO, CLNOME, CLTELEFO, (Select Count(*) From HISTORICO Where HICODSOC=CLCODIGO) As Quantidade From CLIENTES Where (Select count(*) From HISTORICO Where HICODSOC=CLCODIGO) >=300 Order By CLNOME => o mesmo que o anterior, porém mostrando a quantidade de locações de cada cliente.

Exercício: Mostrar o nome de todos os clientes que locaram o filme “Jurassic Park”.

Select CLCODIGO, CLNOME, CLTELEFO From CLIENTES Where Exists (Select * From HISTORICO Where HICODSOC=CLCODIGO) Order By CLNOME => mostra todos os clientes que já locaram pelo menos uma vez, isto é, se a consulta depois do comando Exists (interna) retornar pelo menos uma linha, a consulta mais externa apresentará o cliente correspondente.

Exercício: Mostrar o código, o nome, o telefone e a quantidade de locações de todos os clientes que já locaram, ordenados pela quantidade de locações (do maior para o menor).
Obs. Se não for colocado o **Exists** mostra todos os clientes, inclusive os que nunca locaram.

Exercício: Mostrar apenas o nome de todos os clientes que já locaram pelo menos uma vez.

Exercício: Mostrar todos os códigos dos clientes que locaram fitas, porém que não se encontram cadastrados (os clientes foram excluídos, porém seus códigos continuam no histórico de locações).

O exemplo seguinte apresenta uma maneira de usar a mesma tabela duas vezes, uma na consulta mais externa (F1) e outra na consulta mais externa (F2).
Select Distinct F1.FIGENERO, (Select Count(*) From FILMES F2 Where F2.FIGENERO=F1.FIGENERO) As Quant From FILMES F1 Order By Quant => Mostra quantos filmes de cada gênero existem, ordenados pela quantidade.

Exercício: Apresentar o código e o nome dos clientes que possuem o mesmo nome.

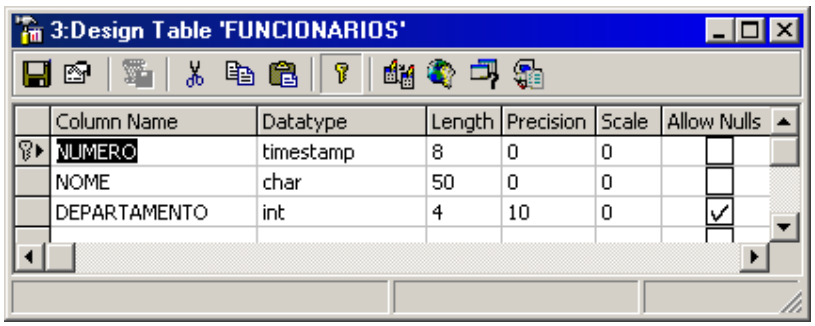
Exercício: Mostrar o nome dos filmes e games que possuem o mesmo nome.

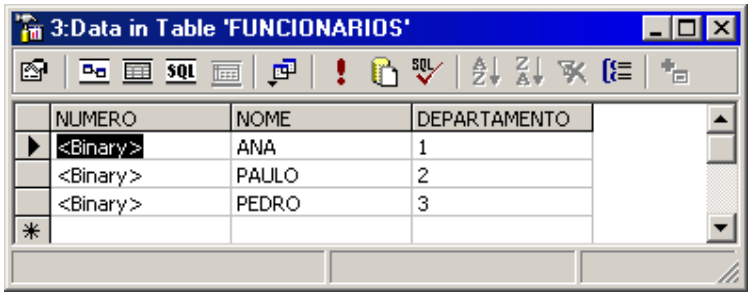
Uso do INNER JOIN para mesclar dados de múltiplas tabelas:

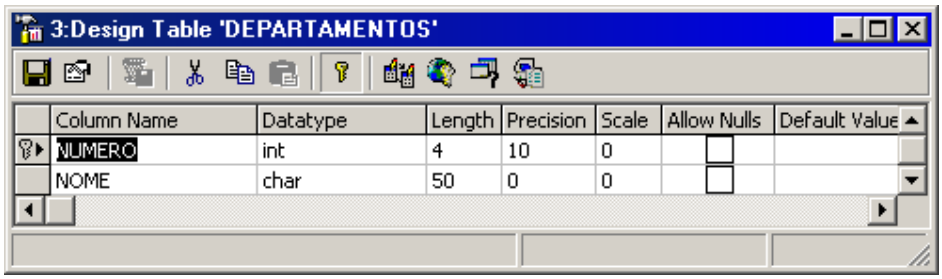
Freqüentemente é necessário mesclar dados de múltiplas tabelas dependendo das necessidades de consulta. Isto é referido como junção de tabelas e é realizado normalmente através da operação ***INNER JOIN*** na cláusula **From** de uma consulta **Select**. Um ***INNER JOIN*** mescla os registros de duas ou mais tabelas testando a correspondência com valores em um campo que é comum para ambas às tabelas.

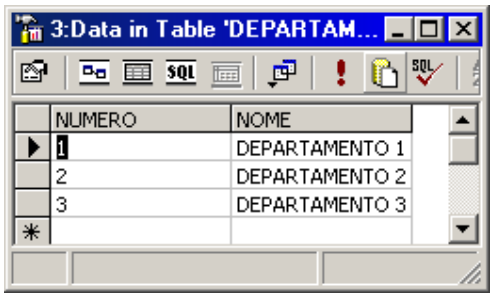
Como descobrir quais são os clientes que possuem o mesmo nome e estão cadastrados com códigos diferentes?

A forma mais simples de se mesclar uma tabela se refere ao produto cartesiano que pode ser realizado apenas com uma instrução Select, conforme apresentado a seguir. Antes de testar os comandos, crie as tabelas seguintes de acordo com as figuras.









Select FUNCIONARIOS.NOME,DEPARTAMENTOS.NOME from FUNCIONARIOS, DEPARTAMENTOS → Essa linha de comando realiza o produto cartesiano entre as tabelas funcionários e departamentos. Observe o nome da tabela inserido antes do nome do campo, uma vez que existe o mesmo nome de campo (NOME) nas duas tabelas.

Tipos de Junções:

1 - Junção Idêntica (ou equivalente): Equi Join
Realiza a junção de duas tabelas quando as colunas das duas tabelas (chave primária e chave estrangeira) possuírem o mesmo conteúdo.

Sintaxe:
Select * From *Tabela1* **INNER JOIN** *Tabela2* **On** *Tabela1.Campo* **=** *Tabela2.Campo*

Exemplo:
```
Select FUNCIONARIOS.NOME, DEPARTAMENTOS.NOME
from FUNCIONARIOS
inner join DEPARTAMENTOS
on FUNCIONARIOS.DEPARTAMENTO = DEPARTAMENTOS.NUMERO
```

NOME	NOME
ANA	DEPARTAMENTO 1
PAULO	DEPARTAMENTO 2
PEDRO	DEPARTAMENTO 3

O mesmo resultado é alcançado com:
```
Select FUNCIONARIOS.NOME, DEPARTAMENTOS.NOME
from FUNCIONARIOS, DEPARTAMENTOS
where FUNCIONARIOS.DEPARTAMENTO = DEPARTAMENTOS.NUMERO
```

Uso de apelidos: quando o nome da tabela é repetido diversas vezes, tornando demorado o processo de digitação das linhas de comando, torna-se possível utilizar apelidos, reduzindo a digitação. O script seguinte fornece o mesmo resultado do script anterior.
```
Select F.NOME, D.NOME
from FUNCIONARIOS F
inner join DEPARTAMENTOS D
on F.DEPARTAMENTO = D.NUMERO
```

Junções entre 3 tabelas:
A consulta seguinte apresenta a data de locação (Histórico), o nome do filme (filmes) e o nome do cliente (clientes) que alugou o filme. A primeira tabela a ser lida será a Histórico, uma vez que aparece em primeiro lugar no comando Select.

```
Select H.HIDATLOC, C.CLNOME, F.FINOME
from HISTORICO H, CLIENTES C,FILMES F
where H.HICODSOC = C.CLCODIGO
and H.HICODFIL = F.FICODIGO
and year(H.HIDATLOC)=1995
order by H.HIDATLOC
```

ou

```
Select H.HIDATLOC, C.CLNOME, F.FINOME
from HISTORICO H
inner join CLIENTES C
on H.HICODSOC = C.CLCODIGO
inner join FILMES F
```

on H.HICODFIL = F.FICODIGO
and year(H.HIDATLOC)=1995
order by H.HIDATLOC

Exercício: Mostrar o nome de todos os clientes e o nome de todos os filmes que cada um deles locou e a data de locação, classificado pelo nome do cliente. Considerar apenas os filmes locados em 1993.

Exercício: mostrar o nome dos filmes locados pelo cliente de código igual a **1642.**

Exercício: Mostrar o nome dos últimos 5 clientes que locaram o filme "02500".

2 - Junção Externa: Outer Join
Fornecem um resultado parecido a junção equivalente, porém, acrescentando também os registros que não possuem correspondência nas duas tabelas.
Para que o exemplo funcione corretamente é necessário excluir o registro correspondente ao departamento número 3 (tabela de departamentos).

Select * from FUNCIONARIOS
left outer join DEPARTAMENTOS
on FUNCIONARIOS.DEPARTAMENTO = DEPARTAMENTOS.NUMERO

NUMERO	NOME	DEPARTA...	NUMERO	NOME
0x0000000000000269	ANA	1	1	DEPARTAMENTO 1
0x000000000000026A	PAULO	2	2	DEPARTAMENTO 2
0x000000000000026B	PEDRO	3	NULL	NULL

Como o departamento 3 foi deletado, o funcionário Pedro não está alocado em nenhum departamento, aparecendo o conteúdo NULL.
Obs. Caso as tabelas Funcionários e Departamentos apareçam na ordem trocada o resultado será diferente. Uma junção externa não pode usar o operador IN ou estar vinculada a outra condição pelo operador OR.

O mesmo resultado pode ser alcançado com:
Select * from FUNCIONARIOS,DEPARTAMENTOS
where FUNCIONARIOS.DEPARTAMENTO *= DEPARTAMENTOS.NUMERO
O sinal * ao lado esquerdo do sinal de igual tem a mesma função de **left outer join**.

Select * from FUNCIONARIOS
right outer join DEPARTAMENTOS
on FUNCIONARIOS.DEPARTAMENTO = DEPARTAMENTOS.NUMERO
O comando **right** indica que a pesquisa começará pela tabela a direita da palavra join, portanto, sentido contrário em relação ao left outer join, conforme a seguir:

NUMERO	NOME	DEPARTA...	NUMERO	NOME
0x000000000000...	ANA	1	1	DEPARTAMENTO 1
0x000000000000...	PAULO	2	2	DEPARTAMENTO 2

O mesmo resultado pode ser alcançado com o script:
Select * from FUNCIONARIOS,DEPARTAMENTOS

where FUNCIONARIOS.DEPARTAMENTO =* DEPARTAMENTOS.NUMERO
O sinal * ao lado direito do sinal de igual tem a mesma função de **right outer join**.

3 - AutoJunção: Self Join
A autojunção é usada quando uma tabela se relaciona com ela mesmo, isto é, existe um auto-relacionamento na tabela.

Para verificar o funcionamento da AutoJunção é necessário criar a tabela seguinte:

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls	Default Value	Ident
🔑▶	codigo	char	2	0	0	☐		☐
	descricao	char	30	0	0	☐		☐
	produtofinal	char	2	0	0	☑		☐
	quantidade	numeric	9	18	0	☐		☐
						☐		☐

	codigo	descricao	produtofinal	quantidade
▶	01	A		0
	02	B	01	2
	03	C	01	3
	04	D	01	4
*				

Exemplo:
Select P1.DESCRICAO as PRODUTO_FINAL, P2.DESCRICAO as MATERIA_PRIMA, P2.QUANTIDADE from Produtos P1,PRODUTOS P2
where P1.CODIGO = P2.PRODUTOFINAL

Para a mesma tabela são fornecidos dois apelidos diferentes (P1 e P2) para realizar a autojunção.
O resultado da execução do script anterior é mostrado a seguir:

PRODUTO_FINAL	MATERIA_PRIMA	QUANTIDADE
A	B	2
A	C	3
A	D	4

Pode ser observado que o produto final A é composto por 2 produtos B, 3 produtos C e 4 produtos D.

Dada a tabela seguinte:

	Column Name	Datatype	Length	Precision	Scale	Allow Nulls	Default Value	Identi
🔑▶	codigo	char	2	0	0	☐		☐
	nome	char	30	0	0	☐		☐
	pai	char	2	0	0	☑		☐
	mae	char	2	0	0	☑		☐
						☐		☐

	codigo	nome	pai	mae
▶	01	PEDRO	<NULL>	<NULL>
	02	ANA	<NULL>	<NULL>
	03	THIAGO	01	<NULL>
	04	FELIPE	05	<NULL>
	05	PAULO	<NULL>	<NULL>
	06	CARLOS	<NULL>	02
	07	SANDRA	<NULL>	<NULL>
	08	TOBIAS	06	07
*				

Crie uma consulta que apresente os pais e seus filhos correspondentes:

```
Select P1.NOME as PAI,P2.NOME as FILHO
from PESSOAS P1,PESSOAS P2
where P1.CODIGO = P2.PAI
```

Crie uma consulta que apresente as mães e seus filhos correspondentes:

```
Select P1.NOME as PAI,P2.NOME as FILHO
from PESSOAS P1,PESSOAS P2
where P1.CODIGO = P2.MAE
```

Crie uma consulta que apresente pais e mães e seus filhos correspondentes:

```
Select P1.NOME as PAIS,P2.NOME AS FILHO
from PESSOAS P1,PESSOAS P2
where P1.CODIGO = P2.PAI
union
Select P1.NOME as PAIS,P2.NOME AS FILHO
from PESSOAS P1,PESSOAS P2
where P1.CODIGO = P2.MAE
```

PAIS	FILHO
ANA	CARLOS
SANDRA	TOBIAS
PAULO	FELIPE
CARLOS	TOBIAS
PEDRO	THIAGO

Crie uma consulta que apresente os filhos que possuem pai e mãe:

```
Select P1.NOME as PAI,P2.NOME AS MAE, P3.NOME AS FILHO
from PESSOAS P1,PESSOAS P2,PESSOAS P3
where P1.CODIGO = P3.PAI
and P2.CODIGO = P3.MAE
```

PAI	MAE	FILHO
CARLOS	SANDRA	TOBIAS

Exercícios finais de consultas em SQL

1 - Todos os clientes com nomes que iniciam com J e que possuem mais de 50 anos:

2 - Todos os filmes e discos que foram comprados em Julho de 1997.

3 - Os discos com valor de compra superior a média.

4 - O nome e preço dos discos com valor superior a media e comprados na Canta Brasil.

5 - Os clientes que fizeram cadastro no mesmo mês que fazem aniversário.

6 - A quantidade de filmes alugados em dezembro de 1994.

7 - Mostrar a quantidade filmes locados pelo cliente de código 1642.

8 - Quais os nomes do filme de código: 02880 e do game de código: 93520

9 - Qual é a data do primeiro cadastro de cliente depois do dia 10/10/1990

10 - Qual o resultado inteiro da soma dos valores dos discos

11- Mostre os nomes de games e filmes que coincidem (os nomes são iguais)

12 - Mostrar o código, o nome e a quantidade de locações de todos os filmes classificados pela quantidade de locações (do mais locado para o menos locado).

13 - Fazer uma subconsulta que retorne o nome de todos os filmes comprados entre os meses de Maio e Outubro a partir do ano de 1997 ordenados pelo nome do filme.

14 - Mostrar a data de nascimento do cliente mais velho nascido depois da data de 10/05/1980

15 - Mostrar os filmes comprados depois do filme ANACONDA, classificados pelo nome do filme.

Tabela de caracteres curinga padrão Access e VB:

Caractere curinga	Descrição
%	Coincide com qualquer número de caracteres. O padrão *program%* encontrará Program, programming, programa, programação, programador, etc. Note que o % (porcentagem) pode ser usado como o primeiro ou último caractere da string, ou ambos (primeiro e último). O padrão %Program% localizará strings que contêm as palavras program, programming, nonprogrammer, reprogramar, etc.
_	O caractere "_" (underline) coincide com um e somente um caractere alfanumérico. O padrão *m_u* encontrará *mau* e *meu* mas não *miau*.
[]	Coincide com qualquer caractere único entre colchetes. O padrão *Santa [YI]nês* encontrará tanto S*anta Inês* quanto S*anta Ynês*. Este padrão é muito útil para localizar possíveis palavras escritas incorretamente em uma única passada. Por exemplo: **SELECT * FROM clientes WHERE clnome LIKE '[WV]AGNER%'**

Tabela de Funções LTRIM e RTRIM :

Função	Exemplo	Valor de retorno	Observação
LTRIM()	LTRIM(“ Marisa Basile ”)	“Marisa Basile ”	Retira apenas os espaços iniciais.
RTRIM()	RTRIM(“ Marisa Basile ”)	“ Marisa Basile”	Retira apenas os espaços Finais.