COMANDOS PARA CONSULTA VIA SQL NO ACCESS

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)

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 Tabela 2 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.	Contrainst

Tabela 2 – 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 Tabela 3 mostra os vários serviços que a DML deve prover, juntamente com os comandos MJ SQL que os implementam.

Tabela 3 – 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, Inser 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:

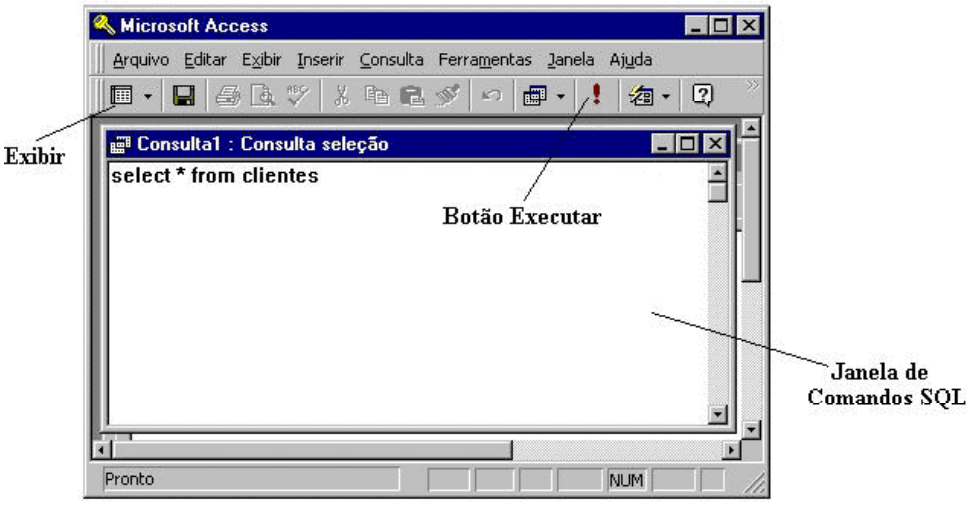
✓ **Instrução SELECT**
Objetivo: Selecionar registros de acordo com os critérios estabelecidos.
Sintaxe:
SELECT <campos> FROM <tabela> [WHERE = critério(s)]
[ORDER BY campo(s)] [GROUP BY campo(s)]

Acessando uma Tabela pelo VisData para Consulta SQL:

1. Dentro do Visual Basic acesse o **VisData** indo no menu **Add-Ins / Visual Data Manager...**
2. Depois de aberto o VisData, será necessário abrir uma tabela qualquer de um Banco de Dados. Para isso vá no menu **File / Open DataBase... / Microsoft Access...** , escolha o Banco de Dados que deseja abrir.
3. Depois, na janela **Database Window**, dê um duplo click com o mouse na tabela que deseja abrir.
4. Agora já se pode digitar os comandos na janela **SQL Statement**. Para visualizar o resultado, click em **Execute**, o Visdata poderá lhe fazer a seguinte pergunta: *Is This SQLPass Through Query?* , mas, por enquanto, seu significado não interessa, click no botão **No** para prosseguir.

Acessando uma Tabela pelo Microsoft Access para Consulta SQL:

1. Inicie o Microsoft Access e abra o banco de dados o qual deseja efetuar a consulta.
2. Com o Banco de Dados aberto, click na guia **Consultas** e em seguida no botão **Novo**. Aparecerá uma tela pedindo que se informe o tipo de consulta a ser criada, escolha **Modo Estrutura** e pressione o botão **OK.**
3. Agora aparecerá uma tela pedindo que se informe as tabelas a serem utilizadas na consulta. Não será necessário escolher as tabelas pois isto será feito depois com os comandos SQL. Click no botão **Fechar**.
4. Selecione o menu **Exibir / Modo SQL**. Aparecerá uma tela onde devem ser digitados os comandos em SQL.
5. Para verificar o resultado da consulta, acesse o menu **Exibir / Modo folha de dados** ou pressione o botão **Executar** da Barra de Ferramentas. Para retornar à tela de comandos, acesse novamente o Modo SQL. Pode-se utilizar também o botão **Exibir** da Barra de Ferramentas para escolher o modo de exibição.



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, CLENDERE, 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, CLENDERE, CLCEP from CLIENTES where CLCEP = "13330-000" ➔ Traduzindo a declaração: *Selecione as colunas CLNOME, CLENDERE, 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 todos os 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 < = #30/01/1980# ➔ Serão mostradas as colonas CLNOME e CLDATNAS cujo valor das linhas da coluna CLDATNAS seja anterior a 30/01/1980.

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 #31/12/1945# → 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 1980

Select * from CLIENTES where CLDATNAS Not Between #01/01/1943# And #31/12/1945# →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 "JOSE*" → 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 "JOSE", não importando o que venha depois.*

Exercício: -Mostrar todos os nomes de clientes que comecem com JOSÉ e terminem com SILVA

OBS 1: *Se ao invés de utilizarmos o MJ SQL (SQL do Access ou VB) estivermos utilizando o SQL Server, o * (asterisco) deve ser substituido pelo % (porcento). Isso decorre do fato de existirem vários dialetos do SQL. (Veja a tabela de caracteres curinga no final).*

OBS 2: *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 "Marisa Basile", um usuário poderia digitar " Marisa Basile" (note que há um espaço em branco antes do nome). Se tal fato ocorrer, a consulta*

Select * From CLIENTES Where CLNOME Like "Ma*"

Não incluirá o nome " Marisa Basile", pois esse nome inicia-se agora com um " M" (espaço seguido da letra "M") e não com as letras "Ma". Para evitarmos isso, devemos utilizar a função TRIM() com Access, ou as funções LTRIM() e RTRIM() com o SQL Server. A função TRIM() remove os espaços iniciais e finais de uma string. LTRIM() remove apenas os espaços iniciais (espaços à esquerda) e RTRIM() remove os espaços finais (espaços à direita). No Access, a consulta anterior pode ser refeita de modo a ignorar os espaços iniciais e finais de uma string:

Select * From CLIENTES Where Trim(CLNOME) Like "Ma"

Agora, mesmo que o usuário inserisse acidentalmente espaços durante a digitação, a consulta retornaria o resultado esperado, ou seja, incluiria no resultado a string " Marisa Basile". (Para melhor compreensão veja a tabela de Funções LTRIM(), RTRIM(), TRIM() no final).

Select * from CLIENTES where CLCIDADE Like "?.*" → O caracter **?** (interrogação) eqüivale 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.

OBS: *Se ao invés de utilizarmos o MJ SQL (SQL do Access ou VB) estivermos utilizando o SQL Server, ? (interrogação) deve ser substituído pelo _ (underline). (Veja a tabela de caracteres curinga no final).*

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 quantos filmes diferentes 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, será listado os nomes de todas os clientes agrupados por cidade.

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

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. Note que logo após a execução do comando, o nome da coluna apresentada estará com o nome: Expr1000 que representa a expressão utilizada para a geração do resultado, no caso a média. Este nome pode ser substituído, veja um outro exemplo abaixo.

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 Count(*) As Total_Registros from CLIENTES → Permite saber o número de registros de acordo com o argumento. Especificamos os campos a serem utilizados em **Count(**Nome_Campo**)** e os argumentos a serem utilizados, com a cláusula **Where** logo após o nome da tabela como já é de conhecimento. Ex: **Where CLCODIGO = 1000**.

Select Count(DIVALPAG) As Total_Registros from DISCOS Where DIVALPAG > 18 → Mostrará o número de registros onde o valor dos discos forem maiores que 18.

Select First(CLCODIGO) from CLIENTES → Retorna o primeiro registro da coluna CLCODIGO da tabela CLIENTES

Select Last(CLCODIGO) from CLIENTES → Retorna o último registro da coluna CODIGO da tabela CLIENTES.

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 * 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). No caso, seriam 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.*

Select Top 10 Percent * from DISCOS Order By DICODIGO Desc → Retornará os registros superiores em um conjunto de resultados de acordo com a sua porcentagem dos registros totais na tabela. No caso, retornará 10% superiores dos registros 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:

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 = #21/01/1995# and HICODFIL = "02300 " → Essa consulta retornará o código do cliente que locou o filme de código "02300" no

dia 21/01/1995.É disso que você precisa como valor para executar a 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 = #21/01/1995# and HICODFIL = "02300 ")

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. Já é de conhecimento o uso do operador **NOT,** mas onde usar? Veja o exemplo:
Select CLNOME from CLIENTES where Not CLCODIGO = (Select HICODSOC from HISTORICO where HIDATLOC = #21/01/1995# and HICODFIL = "02300 ")

OBS: *Quando uma subconsulta resultar em mais de um valor, deve-se utilizar **IN** em vez de **WHERE**. 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 = #21/01/1995# and HICODFIL <> "02300 ")

Exercícios:

- Mostrar apenas os nomes dos filmes locados entre os anos de 1995 e 1997 ordenados pelo nome.
- Mostrar nome e gênero dos Filmes do gênero ACAO locados em 1998.
- Mostrar o Histórico de locações do filme "02500".

Uso do INNER JOIN para mesclar dados de múltiplas tabelas:

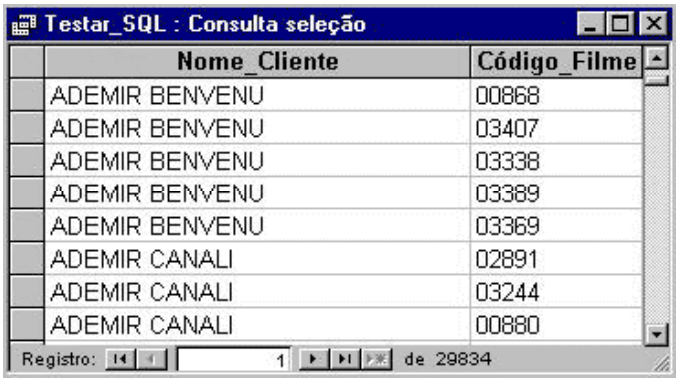
Freqüentemente é necessário mesclar dados de múltiplas tabelas em uma única visualização para propósitos de análise. Isto é referido como junção de tabelas e é realizado utilizando a operação ***INNER JOIN*** na cláusula **From** de uma consulta **Select**. Um ***INNER JOIN*** mescla os registros de duas ou mais tabelas testando a correspondência com valores em um campo que é comum para ambas as tabelas. Veja sua sintaxe:

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

Exemplo:

Select * From CLIENTES Inner Join HISTORICO On CLIENTES.CLCODIGO = HISTORICO.HICODSOC Order By CLNOME → Mostrará todos os campos tanto da tabela CLIENTES como da tabela HISTÓRICO ordenados pelo nome do cliente, ou seja, esta consulta mostrará por exemplo todos os códigos dos filmes locados por cada cliente cadastrado.
Caso queira melhorar um pouco mais a visualização desta consulta, utilize o seguinte comando abaixo:

Select CLNOME As Nome_Cliente, HICODFIL As Código_Filme From CLIENTES Inner Join HISTORICO On CLIENTES.CLCODIGO = HISTORICO.HICODSOC Order By CLNOME, HICODFIL



Obs: Note que podemos selecionar qualquer um dos campos da tabela CLIENTES como da tabela HISTORICO para serem mostrados na consulta pois as duas tabelas estão sendo utilizadas.
- Para mostrar por exemplo todos os códigos de filmes locados por um determinado cliente de código igual a **1642**, usaríamos: **Select HICODFIL As Codigo_Filme From CLIENTES Inner Join HISTORICO On CLIENTES.CLCODIGO = HISTORICO.HICODSOC Where CLIENTES.CLCODIGO = '1642'** lembrando que isto também poderia se feito com uma subconsulta simples e os resultados seriam os mesmos:

Select HICODFIL As Codigo_Filme From HISTORICO Where HICODSOC In (Select CLCODIGO From CLIENTES Where CLCODIGO = '1642')

Exercício: Crie uma subconsulta juntamente com a cláusula INNER JOIN para retornar o código e o nome de todos os filmes locados por um cliente. Utilize **1642** como código do cliente que se deseja consultar e ordene 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 * (asterisco) 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.
?	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 *santa Inês* quanto *santa Ynês*. Este padrão é muito

	útil para localizar possíveis palavras escritas incorretamente em uma única passada.
[!]	Coincide com qualquer caractere diferente dos que estão entre colchetes. O padrão **q[!u]** encontrará palavras que contém o caractere *q* não seguido pelo caractere *u*.
[-]	Coincide com qualquer caractere de um intervalo de caracteres. Os caracteres devem ser consecutivos no alfabeto e ordenados em uma ordem específica (A até Z) ou (1 até 9).
#	Corresponde exatamente a um caractere numérico. O padrão D1## encontrará D100 e D139, mas não encontrará D1000 ou D10.

Tabela de Funções TRIM(), LTRIM e RTRIM :

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