SQL
Structured Query Language

Oracle 10g

Tipos de dados:

number(p,e)	precisão e escala definidos pelo usuário
varchar2(n)	string de tamanho variável
date	data (aaaa/mm/dd)
timestamp	data e hora (aaaa/mm/dd hh:mm:ss)
char(n)	string de tamanho fixo
clob	character large object - documentos ou memorandos
blob	binary large object - fotos, vídeos, áudio
nvarchar2(n)	string de tamanho variável - multilingua (tamanho max. n)
binary_float	ponto flutuante de precisão simples - científico
binary_double	ponto flutuante de precisão dupla - científico

1. CREATE TABLE
Use a instrução CREATE TABLE para definir uma nova tabela e seus campos e restrições de campo

1.1. Sintaxe
CREATE TABLE tabela (campo1 tipo [(tamanho)] [NOT NULL] [índice1] [, campo2 tipo [(tamanho)]
[NOT NULL] [índice2] [, ...]] [, CONSTRAINT índicedemulticampos [, ...]])

1.2. A instrução CREATE TABLE tem estas partes:

Parte	Descrição
tabela	O nome da tabela a ser criada.
campo1, campo2	O nome do campo ou campos a serem criados na nova tabela. Uma tabela deve ter pelo menos um campo.
tipo	O tipo de dados de campo na nova tabela.
tamanho	O tamanho do campo em caracteres (se o tipo exigir).
índice1, índice2	Uma cláusula CONSTRAINT que define um índice de campo único.
Índicedemulticampos	Uma cláusula CONSTRAINT que define um índice de campos múltiplos.

1.2. Exemplos

a) Criar a tabela: DEPARTAMENTO

```
CREATE TABLE DEPARTAMENTOS
(Codigo Varchar2(3) NOT NULL,
Nome Varchar2(30) NOT NULL,
CONSTRAINT pk_Codigo PRIMARY KEY (Codigo))
```

b) Criar a tabela: CURSOS

```
CREATE TABLE CURSOS
(Codigo Varchar2(3) NOT NULL,
Nome Varchar2(30) NOT NULL,
CodigoDepartamento Varchar2(3) NOT NULL,
CONSTRAINT pk_Codigo PRIMARY KEY (Codigo),
CONSTRAINT fk_Depto FOREIGN KEY (CodigoDepartamento)
REFERENCES DEPARTAMENTOS (Codigo))
```

c) Cria a tabela: DISCIPLINAS

```
CREATE TABLE DISCIPLINAS
(CodigoCurso Varchar2(3) NOT NULL,
Numero Number(5) NOT NULL,
Nome Varchar2(30) NOT NULL,
Creditos Number(1) NOT NULL,
Laboratorio Number(1) NOT NULL,
Prelecao Number(1) NOT NULL,
CodigoDepartamento Varchar2(3) NOT NULL,
CONSTRAINT pk_Disciplina PRIMARY KEY (CodigoCurso, Numero),
CONSTRAINT fk_CursoDisciplina FOREIGN KEY (CodigoCurso)
REFERENCES CURSOS (Codigo))
```

d) Criar a Tabela: SEMESTRES

```
CREATE TABLE SEMESTRES
(Ano Number(4) NOT NULL,
Numero Number(1) NOT NULL,
CONSTRAINT pk_Semestre PRIMARY KEY (Ano,Numero))
```

e) Criar a Tabela: TURMAS

```
CREATE TABLE TURMAS
(CodigoCurso Varchar2(3) NOT NULL,
NumeroDisciplina Number(3) NOT NULL,
Numero Varchar2(3) NOT NULL,
NumeroSubturma Varchar2(1) NOT NULL,
AnoSemestre Number(4) NOT NULL,
NumeroSemestre Number(1) NOT NULL,
CONSTRAINT pkTurma PRIMARY KEY (CodigoCurso, NumeroDisciplina,
Numero, NumeroSubTurma, AnoSemestre,NumeroSemestre),
```

```
CONSTRAINT fk_DisciplinaTurma
FOREIGN KEY (CodigoCurso,NumeroDisciplina)
REFERENCES DISCIPLINAS (CodigoCurso,Numero),
CONSTRAINT fk_DisciplinaSemestre
FOREIGN KEY (AnoSemestre,NumeroSemestre)
REFERENCES SEMESTRES (Ano,Numero))
```

1.3. Comentários

Uma cláusula CONSTRAINT estabelece várias restrições em um campo e pode ser utilizada para estabelecer a chave primária.

2. ALTER TABLE

Modifica a estrutura de uma tabela depois de ter sido criada com a instrução CREATE TABLE.

2.1. Sintaxe

```
ALTER TABLE tabela {ADD {COLUMN campo tipo[(tamanho)] [NOT NULL]
[CONSTRAINT índice] |
CONSTRAINT índicedemulticampos} | DROP {COLUMN campo I CONSTRAINT
nomedoíndice} }
```

2.2. A instrução ALTER TABLE tem estas partes:

Parte	Descrição
Tabela	O nome da tabela a ser alterada.
Campo	O nome do campo a ser adicionado ou excluído da tabela.
Tipo	O tipo de dados de campo.
Tamanho	O tamanho do campo em caracteres (somente os campos Varchar2, Char, Nvarchar2 e Number).
Índice	O índice para campo. Consulte o tópico da cláusula CONSTRAINT para maiores informações sobre como construir este índice.
Índicedemulticampos	A definição de um índice de campos múltiplos a ser adicionado à tabela.
Nomedoíndice	O nome do índice de campo múltiplo a ser removido.

2.3. Exemplos

a) Acrescentar na tabela DEPARTAMENTOS, o atributo DataDeCriacao de preenchimento opcional e do tipo data.

```
ALTER TABLE DEPARTAMENTOS ADD COLUMN DataDeCriacao Date
```

b) Excluir da tabela DEPARTAMENTOS, o atributo DataDeCriacao

ALTER TABLE DEPARTAMENTOS DROP COLUMN DataDeCriacao

2.4. Comentários

Através da instrução ALTER TABLE, você pode alterar uma tabela existente de diversas maneiras. Você pode:

Utilizar ADD COLUMN para adicionar um novo campo à tabela. Você especifica o nome do campo, tipo de dados e se o tipo de campo exigir, um tamanho.

Utilizar ADD CONSTRAINT para adicionar um índice de campos múltiplos.

Utilizar DROP COLUMN para excluir um campo. Você especifica somente o nome do campo.

Utilizar DROP CONSTRAINT para excluir um índice de campos múltiplos. Você especifica somente o nome do índice após a palavra reservada CONSTRAINT.

Você não pode adicionar ou excluir mais de um campo ou índice de cada vez.

Você pode utilizar NOT NULL em um campo único ou dentro de uma cláusula CONSTRAINT nomeada que se aplica tanto a uma CONSTRAINT de campo único ou campos múltiplos. Contudo, você pode aplicar a restrição NOT NULL somente uma vez a um campo pois, senão, ocorrerá um erro em tempo de execução.

3. DROP TABLE

Exclui uma tabela existente de um banco de dados ou exclui um índice existente de uma tabela.

3.1. Sintaxe
DROP {TABLE tabela | INDEX índice ON tabela}

3.2. A instrução DROP tem estas partes:

Parte	Descrição
Tabela	O nome da tabela a ser excluída ou a tabela a partir da qual um índice deve ser excluído.
Índice	O nome do índice a ser excluído da tabela.

3.3. Exemplos

a) Excluir a Tabela: TABEXEMPLO criada pelo comando abaixo

Criando a tabela

CREATE TABLE TABEXEMPLO
(Codigo Varchar2(3), Nome Varchar2(30), DataNascimento Date,
CONSTRAINT pk_Codigo PRIMARY KEY (Codigo))

Excluindo a tabela

DROP TABLE TABEXEMPLO

3.4. Comentários

Você deve fechar a tabela para poder excluí-la ou remover um índice dela. Você também pode utilizar ALTER TABLE para excluir um índice de uma tabela. Você pode utilizar CREATE TABLE para criar uma tabela e CREATE INDEX ou ALTER TABLE para criar um índice. Para modificar uma tabela, use ALTER TABLE.

4. CREATE INDEX
Cria um novo índice em uma tabela existente.

4.1. Sintaxe
CREATE [UNIQUE] INDEX índice ON tabela (campo [ASC|DESC][, campo [ASC|DESC], ...])
[WITH { PRIMARY | DISALLOW NULL | IGNORE NULL }]

4.2. A instrução CREATE INDEX tem estas partes:

Parte	Descrição
índice	O nome do índice a ser criado.
tabela	O nome da tabela existente que conterá o índice.
campo	O nome do campo ou campos a serem indexados. Para criar um índice de campo único, liste o nome do campo entre parênteses após o nome da tabela. Para criar um índice de campos múltiplos, liste o nome de cada campo a ser incluído no índice. Para criar índices descendentes, use a palavra reservada DESC; caso ontrário, assume-se que os índices são ascendentes.

4.3. Exemplos

a) Criar um Índice único para o campo nome da tabela: DEPARTAMENTOS

CREATE UNIQUE INDEX idx_Nome ON DEPARTAMENTOS (Nome ASC)

A cláusula UNIQUE indica que o índice criado não pode Ter valores duplicados.

b) Criar um índice para o campo nome da tabela DEPARTAMENTOS proibindo a entrada de dados nulos.

CREATE INDEX idx_Nome ON DEPARTAMENTOS (Nome ASC) WITH DISALLOW NULL

A cláusula WITH DISALLOW NULL proibe a entrada de dados nulos no campo nome.
c) Criar um índice para o campo nome da tabela DEPARTAMENTOS que não contenha valores nulos:

CREATE INDEX idx_Nome ON DEPARTAMENTOS (Nome ASC) WITH IGNORE NULL

A cláusula WITH IGNORE NULL proibe valores nulos no índice.

4.4. Comentários

Para proibir valores duplicados no campo ou campos indexados de diferentes registros, use a palavra reservada UNIQUE.

Na cláusula WITH opcional, você pode impor regras de validação de dados. Você pode:

Proibir entradas Null no campo ou campos indexados dos novos registros, utilizando a opção DISALLOW NULL

Impedir que registros com valores Null no campo ou campos indexados sejam incluídos no índice utilizando a opção IGNORE NULL.

Designar o campo ou campos indexados como a chave primária utilizando a palavra reservada PRIMARY.
Isto significa que a chave é exclusiva e, portanto, você pode omitir a palavra reservada UNIQUE.

Você também pode utilizar a instrução ALTER TABLE para adicionar um índice de campo único ou de campos múltiplos a uma tabela e pode utilizar a instrução ALTER TABLE ou a instrução DROP para remover um índice criado com ALTER TABLE ou CREATE INDEX.

5. CONSTRAINT

Uma restrição é semelhante a um índice, embora também possa ser utilizada para estabelecer uma relação com uma outra tabela.

Você utiliza a cláusula CONSTRAINT nas instruções ALTER TABLE e CREATE TABLE para criar ou excluir restrições. Há dois tipos de cláusulas ONSTRAINT:

um para criar uma restrição em um campo único e outro para criar uma restrição em mais de um campo.

5.4. Sintaxe

Restrição de campo único:

CONSTRAINT nome {PRIMARY KEY | UNIQUE | NOT NULL | REFERENCES tabelaexterna [(campoexterno1, campoexterno2)]}

Restrição de campos múltiplos:

CONSTRAINT nome {PRIMARY KEY (primária1[, primária2 [, ...]]) |

UNIQUE (exclusiva1[, exclusiva2 [, ...]]) |

NOT NULL (nãonulo1[, nãonulo2 [, ...]]) |

FOREIGN KEY (ref1[, ref2 [, ...]])

REFERENCES tabelaexterna [(campoexterno1 [, campoexterno2 [, ...]])]}

5.5. A cláusula CONSTRAINT tem estas partes:

Parte	Descrição
Nome	O nome da restrição a ser criada.
primária1, primária2	O nome do campo ou campos a ser designado(s) à chave primária.
exclusiva1, exclusiva2	O nome do campo ou campos a ser designado(s) como uma chave exclusiva.
nãonulo1, nãonulo2	O nome do campo ou campos que estão restritos a valores não-Null
ref1, ref2	O nome do campo ou campos de uma chave externa que fazem referência a campos em uma outra tabela.
Tabelaexterna	O nome da tabela externa contendo o campo ou campos especificados por campoexterno.
campoexterno1, campoexterno2	O nome do campo ou campos na tabelaexterna especificados por ref1, ref2. Você pode omitir esta cláusula se o campo referenciado for a chave primária de tabelaexterna.

5.6. Comentários

Você utiliza a sintaxe para uma restrição de campo único na cláusula de definição de campo de uma instrução ALTER TABLE ou CREATE TABLE que se segue imediatamente à especificação do tipo de dados do campo.

Você utiliza a sintaxe para uma restrição de campos múltiplos sempre que utilizar a palavra reservada CONSTRAINT fora de uma cláusula de definição de campo em uma instrução ALTER TABLE ou CREATE TABLE.

Utilizando CONSTRAINT, você pode designar um campo como um dos seguintes tipos de restrições:

Você pode utilizar a palavra reservada UNIQUE para designar um campo como chave exclusiva. Isto significa que não pode haver dois registros em uma tabela que tenham o mesmo valor neste campo. Você pode restringir qualquer campo ou lista de campos como exclusivo. Se uma restrição de campos múltiplos for designada como uma chave exclusiva, os valores combinados de todos os campos no índice devem ser exclusivos, mesmo que dois ou mais registros tenham o mesmo valor em apenas um dos campos.

Você pode utilizar as palavras reservadas PRIMARY KEY para designar um campo ou conjunto de campos em uma tabela como uma chave primária. Todos os valores na chave primária devem ser exclusivos e não Nulos e só pode haver uma chave primária para uma tabela.

Você pode utilizar as palavras reservadas FOREIGN KEY para designar um campo como uma chave externa. Se a chave primária da tabela externa consistir em mais de um campo, você deverá utilizar uma definição de restrição de campos múltiplos, listando todos os campos referenciais, o nome da tabela externa e os nomes dos campos referenciados na tabela externa, na mesma ordem em que os campos referenciais são listados. Se o campo ou campos referenciados forem a chave primária da tabela externa, você não precisará especificar os campos referenciados — por padrão, o mecanismo de banco de dados se comporta como se a chave primária da tabela externa fosse os campos referenciados.

6. INSERT INTO
Adiciona um registro ou múltiplos registros a uma tabela. Isto é chamado de consulta acréscimo.

6.4. Sintaxe
Acréscimo de múltiplos registros:

INSERT INTO destino [IN bancodedadosexterno] [(campo1[, campo2[, ...]])]
SELECT [origem.]campo1[, campo2[, ...] FROM expressãodetabela

Consulta acréscimo de registro único:

INSERT INTO destino [(campo1[, campo2[, ...]])]
VALUES (valor1[, valor2[, ...])

6.5. A instrução INSERT INTO possui as partes a seguir:
Parte Descrição destino O nome da tabela ou consulta à qual acrescentar os registros. bancodedadosexterno0 O caminho até um banco de dados externo.

Parte	Descrição
destino	O nome da tabela ou consulta à qual acrescentar os registros.
bancodedadosexterno0	O caminho até um banco de dados externo.
origem	O nome da tabela ou consulta a partir da qual os registros vão ser copiados.
campo1, campo2	Nomes dos campos aos quais os dados serão acrescentados, se se seguirem a um argumento destino, ou os nomes dos campos a partir dos quais os dados serão obtidos, se se seguirem a um argumento origem.
Expressãodetabela	O nome da tabela ou das tabelas das quais os registros são inseridos. Este argumento pode ser um nome de tabela simples ou um composto resultante de uma operação INNER JOIN, LEFT JOIN ou RIGHT JOIN ou uma consulta salva.
valor1, valor2	Os valores a serem inseridos nos campos específicos do novo registro. Cada valor é inserido no campo que corresponde à posição do valor na lista: valor1 é inserido no campo1 do novo registro, valor2 no campo2, e assim por diante. Você deve separar os valores com uma vírgula e colocar os campos de texto entre aspas (' ').

6.6. Comentários

Você pode usar a instrução INSERT INTO para adicionar um único registro a uma tabela usando a sintaxe da consulta acréscimo de registro único como mostrado acima. Nesse caso, o código especifica o nome e o valor de cada campo do registro. Você deve especificar cada um dos campos do registro ao qual será atribuído um valor e um valor para aquele campo. Quando você não especifica os campos, o valor padrão ou Null é inserido para colunas ausentes. Os registros são adicionados ao fim da tabela.

Você pode também usar INSERT INTO para acrescentar um conjunto de registros de uma outra tabela ou consulta utilizando a cláusula SELECT ... FROM, como mostrado acima na sintaxe da consulta acréscimo de múltiplos registros. Nesse caso, a cláusula SELECT especifica os campos a acrescentar à tabela destino especificada.

A tabela origem ou destino pode especificar uma tabela ou uma consulta. Se for especificada uma consulta, o mecanismo de banco de dados Microsoft Jet acrescenta registros a todas as tabelas especificadas pela consulta. INSERT INTO é opcional, mas, quando incluída, precede a instrução SELECT. Se a tabela de destino contiver uma chave primária, certifique-se de acrescentar valores exclusivos não-Null ao campo ou campos da chave primária; se não o fizer, o mecanismo de banco de dados Microsoft Jet não acrescentará os registros.

Se você acrescentar os registros a uma tabela com um campo AutoNumeração e quiser renumerar os registros acrescentados, não inclua esse campo. Inclua o campo AutoNumeração na consulta se quiser conservar os valores originais do campo.

Use a cláusula IN para acrescentar os registros a uma tabela em um outro banco de dados.

Para criar uma nova tabela, use a instrução SELECT... INTO em lugar de criar uma consulta criar tabela.

Para descobrir quais registros serão acrescentados antes de executar a consulta acréscimo, primeiro execute e visualize os resultados de uma consulta seleção que use os mesmos critérios de seleção.

Uma consulta acréscimo copia registros de uma ou mais tabelas para outra. As tabelas que contêm os registros que você acrescenta não são afetadas pela consulta acréscimo.

Em vez de acrescentar registros existentes de uma outra tabela, você pode especificar o valor de cada campo em um único registro novo, usando a cláusula VALUES. Se você omitir a lista de campos, a cláusula VALUES deverá

incluir valores para todos os campos da tabela; caso contrário, a operação INSERT falhará. Use uma instrução INSERT INTO adicional com uma cláusula VALUES para cada registro adicional que quiser criar.

7. DELETE
Remove os registros de uma ou mais tabelas relacionadas na cláusula FROM que satisfaçam à cláusula WHERE.

7.4. Sintaxe
DELETE FROM tabela WHERE critérios

7.5. A instrução DELETE possui as partes a seguir:

Parte	Descrição
tabela	O nome da tabela da qual são excluídos registros.
Critérios	Uma expressão que determina os registros a ser excluídos.

7.6. Comentários

DELETE é especialmente útil quando você quer excluir muitos registros. Para excluir uma tabela inteira do banco de dados, você pode usar o método Execute com uma instrução DROP. No entanto, se você excluir a tabela a estrutura será perdida. Em contrapartida, quando você usa DELETE, somente os dados são excluídos; a estrutura da tabela e todas as suas propriedades, como atributos de campo e índices, permanecem intactos.

Você pode usar DELETE para remover registros de tabelas que estão em um relacionamento um-paramuitos com outras tabelas. As operações de exclusão em cascata fazem com que os registros em tabelas que estão no lado muitos do relacionamento sejam excluídos quando o registro correspondente no lado um do relacionamento é excluído na consulta. Por exemplo, nos relacionamentos entre as tabelas Clientes e Pedidos, a tabela Clientes está no lado um e a tabela Pedidos está no lado muitos do relacionamento. A exclusão de um registro de Clientes resulta na exclusão dos registros Pedidos correspondentes se a opção de exclusão em cascata estiver especificada.

O comando DELETE exclui registros inteiros, não somente os dados em campos específicos. Se você quiser excluir valores de um campo específico, crie uma consulta atualização que altere os valores para Null.

7.7. Importante

Depois de remover os registros utilizando uma consulta exclusão, você não poderá desfazer a operação. Se quiser saber quais registros foram excluídos, examine antes os resultados de uma consulta seleção que use os mesmos critérios e, depois, execute a consulta exclusão.

8. UPDATE

Altera valores de campos em uma tabela especificada, com base em critérios especificados.

8.4. Sintaxe
UPDATE tabela SET campo = novovalor WHERE critérios;

8.5. A instrução UPDATE possui as partes a seguir:

Parte	Descrição
tabela	O nome da tabela contendo os dados que você quer modificar.
Novovalor	Uma expressão que determina o valor a ser inserido em um campo específico dos registros atualizados.
critérios	Uma expressão que determina quais registros serão atualizados. Apenas os registros que satisfazem à expressão são atualizados.

8.6. Comentários

UPDATE é especialmente útil quando você quer alterar vários registros ou quando os registros que você quer alterar estão em várias tabelas.
Você pode alterar vários campos ao mesmo tempo. O exemplo a seguir aumenta os valores de Quantia do Pedido em 10% e os valores de Frete em 3% para transportadores no Reino Unido (UK):

```
UPDATE Pedidos SET QuantiaDoPedido = QuantiaDoPedido * 1.1, Frete =
Frete * 1.03 WHERE PaísDeDestino = 'UK';
```

Importante

UPDATE não gera um conjunto de resultados. Além disso, depois de atualizar os registros usando uma consulta de atualização, você não poderá desfazer a operação. Se quiser saber quais os registros que foram atualizados, examine antes os resultados de uma consulta seleção que usem os mesmos critérios e, depois, execute a consulta de atualização. Mantenha sempre cópias de backup dos dados. Se você atualizar os registros errados, poderá recuperá-los a partir das cópias.

9. SELECT INTO

Criar uma tabela a partir de outra.

9.4. Sintaxe

SELECT campo1[, campo2[, ...]] INTO novatabela [IN bancodedadosexterno] FROM origem

9.5. A instrução SELECT...INTO possui as partes a seguir:

Parte	Descrição
campo1, campo2	Os nomes dos campos a ser copiados na nova tabela. novatabela O nome da tabela a ser criada. Deve ser compatível com às convenções de nomenclatura padrão. Se novatabela for igual ao nome de uma tabela existente, ocorrerá um erro interceptável.
Bancodedadosexterno	O caminho para um banco de dados externo. Para obter uma descrição do caminho, consulte a cláusula IN.
origem	O nome da tabela existente a partir da qual os registros são selecionados. Podem ser tabelas simples ou múltiplas ou um consulta.

9.6. Comentários

Você pode utilizar consultas criar tabela para arquivar registros, fazer cópias de backup das tabelas ou fazer
cópias para exportar para um outro banco de dados, ou para usar como base para relatórios que exibem
dados sobre um determinado período de tempo. Por exemplo, você poderia produzir um relatório de
Vendas Mensais por Região, executando a mesma consulta criar tabela todos os meses.

Convém definir uma chave primária para a nova tabela. Quando você cria a tabela, os campos na nova
tabela herdam o tipo de dados e tamanho de campo de cada campo das tabelas base da consulta, mas
nenhuma outra propriedade do campo ou da tabela é transferida.

Para adicionar dados a uma tabela existente, use a instrução INSERT INTO em vez de criar uma consulta
acréscimo.

Para descobrir quais registros serão selecionados antes de executar a consulta criar tabela, examine antes os
resultados de uma instrução SELECT que use os mesmos critérios de seleção.

10. SELECT
Retorna informações do banco de dados como um conjunto de registros.

10.4. Sintaxe

SELECT [atributo] { * | tabela.* | [tabela.]campo1 [AS alias1] [, [tabela.]campo2 [AS alias2] [, ...]]}
FROM expressãodetabela [, ...] [IN bancodedadosexterno]
[WHERE...]
[GROUP BY...]
[HAVING...]
[ORDER BY...]
[WITH OWNERACCESS OPTION]

10.5. A instrução SELECT possui as partes a seguir:

Parte	Descrição
atributo	Um dos atributos a seguir: ALL, DISTINCT, DISTINCTROW ou TOP. Você utiliza o atributo para restringir o número de registros retornados. Se nenhum for especificado, o padrão será ALL.
*	Especifica que todos os campos da tabela ou tabelas especificadas estão selecionados. tabela O nome da tabela contendo os campos a partir dos quais os registros são selecionados.
campo1, campo2	Os nomes dos campos contendo os dados que você deseja recuperar. Se você incluir mais de um campo, eles serão recuperados na ordem listada.
alias1, alias2	Os nomes a serem utilizados como cabeçalhos de coluna em lugar dos nomes originais de coluna em tabela.
expressãodetabela	O nome da tabela ou tabelas contendo os dados que você deseja recuperar.
bancodedadosexterno	nome do banco de dados que contém as tabelas em expressãodetabela se elas não estiverem no banco de dados atual.

10.6. Comentários

Para executar esta operação, o mecanismo de banco de dados Microsoft Jet procura a tabela ou tabelas especificadas, extrai as colunas escolhidas, seleciona as linhas que atendem aos critérios e classifica ou agrupa as linhas resultantes na ordem especificada.

As instruções SELECT não alteram os dados no banco de dados.

SELECT é geralmente a primeira palavra em uma instrução SQL. A maioria das instruções SQL são instruções SELECT ou SELECT...INTO.

A sintaxe mínima para uma instrução SELECT é:

SELECT campos FROM tabela

Você pode utilizar um asterisco (*) para selecionar todos os campos em uma tabela. O exemplo a seguir seleciona todos os campos na tabela Funcionários:

SELECT * FROM Funcionários;

Se um nome de campo for incluído em mais de uma tabela na cláusula FROM, coloque antes dele o nome da tabela e o operador . (ponto). No exemplo a seguir, o campo Departamento está tanto na tabela Funcionários quanto na tabela Supervisores. A instrução SQL seleciona os departamentos a partir da tabela Funcionários e os nomes de supervisores a partir da tabela upervisores:

SELECT Funcionários.Departamento, Supervisores.SupvNome
FROM Funcionários INNER JOIN Supervisores
WHERE Funcionários.Departamento = Supervisores.Departamento;

Sempre que utilizar funções agregadas ou consultas que retornam nomes de objetos Field ambíguos ou duplicados, você deve utilizar a cláusula AS para fornecer um nome alternativo para o objeto Field. O exemplo a seguir utiliza o título ContagemDePessoas para nomear o objeto Field retornado no objeto Recordset resultante:

SELECT COUNT(CódigoDoFuncionário)

AS ContagemDePessoas FROM Funcionários;

Você pode utilizar as outras cláusulas em uma instrução SELECT para ampliar a restrição e organizar os dados retornados. Para maiores informações, consulte o tópico da Ajuda para a cláusula que você está utilizando.

11. Atributos ALL, DISTINCT, DISTINCTROW e TOP
Especifica registros selecionados com consultas SQL.

11.4. Sintaxe
SELECT [ALL | DISTINCT | DISTINCTROW | [TOP n [PERCENT]]]
FROM tabela

11.5. Uma instrução SELECT contendo estes atributos possui as partes a seguir:

Parte	Descrição
ALL	Adotada quando você não inclui um dos atributos. O mecanismo de banco de dados Microsoft Jet seleciona todos os registros que atendam às condições na instrução SQL. Os dois exemplos a seguir são equivalentes e retornam todos os registros da tabela Funcionários: SELECT ALL * FROM Funcionários ORDER BY CódigoDoFuncionário; SELECT * FROM Funcionários ORDER BY CódigoDoFuncionário;
DISTINCT	Omite registros que contêm dados duplicados nos campos selecionados. Para serem incluídos nos resultados da consulta, os valores de cada campo listado na instrução SELECT devem ser exclusivos. Por exemplo, vários funcionários listados em uma tabela Funcionários podem ter o mesmo sobrenome. Se dois registros contiverem Smith no campo Sobrenome, a instrução SQL a seguir retornará somente um deles: SELECT DISTINCT Sobrenome FROM Funcionários; Se você omitir DISTINCT, esta consulta retornará os dois registros Smith.Se a cláusula SELECT contiver mais de um campo, a combinação de valores de todos os campos deverão ser exclusivos para que um dado registro seja incluído nos resultados.A saída de uma consulta que utiliza DISTINCT não é atualizável e não reflete alterações subseqüentes feitas por outros usuários.
DISTINCTROW	Omite dados baseado em registros duplicados completos, e não somente campos duplicados. Por exemplo, você poderia criar uma consulta que associasse as tabelas Cliente e Pedidos no campo CódigoDoCliente. A tabela Clientes não contém campos CódigoDoCliente duplicados, mas a tabela Pedidos contém, pois cada cliente pode fazer vários pedidos. A instrução SQL a seguir mostra como você pode utilizar DISTINCTROW para produzir uma

lista de empresas que têm pelo menos um pedido, mas sem exibir detalhes sobre esses pedidos:

```
SELECT DISTINCTROW NomeDaEmpresa
FROM Clientes INNER JOIN Pedidos
ON Clientes.CódigoDoCliente = Pedidos.CódigoDoCliente
ORDER BY NomeDaEmpresa;
```

Se você omitir DISTINCTROW, esta consulta produzirá várias linhas para cada empresa que tenha mais de um pedido.DISTINCTROW produz um efeito somente quando você seleciona campos de algumas, mas não todas, as tabelas utilizadas na consulta. DISTINCTROW será ignorado se a consulta incluir somente uma tabela ou se você obtiver saída de campos de todas as tabelas.

TOP n [PERCENT] Retorna um certo número de registros que caem no topo ou na base de um intervalo especificado por uma cláusula ORDER BY. Suponha que você deseje obter os nomes dos 25 melhores estudantes da classe de 1994:

```
SELECT TOP 25 Nome, Sobrenome
FROM Estudantes
WHERE AnoDeGraduação = 1994
ORDER BY MédiaDeNotas DESC;
```

Se você não incluir a cláusula ORDER BY, a consulta retornará um conjunto arbitrário de 25 registros da tabela Estudantes que satisfaçam à cláusula WHERE.O atributo TOP não escolhe entre valores iguais. No exemplo anterior, se a vigésima quinta e a vigésima sexta melhores médias de notas forem iguais, a consulta retornará 26 registros.Você também pode utilizar a palavra reservada PERCENT para retornar uma certa porcentagem de registros que se situem no topo ou na base de um intervalo especificado pela cláusula ORDER BY. Suponha que, em vez dos 25 melhores estudantes, você queira os 10% inferiores da classe:

```
SELECT TOP 10 PERCENT Nome, Sobrenome
FROM Estudantes
WHERE AnoDeGraduação = 1994
ORDER BY MédiaDeNotas ASC;
```

O atributo ASC especifica um retorno de valores inferiores. O valor que se segue a TOP deve ser um

	Number sem sinal.TOP não afeta o fato de a consulta ser ou não atualizável.
tabela	O nome da tabela a partir da qual os registros são recuperados.

12. Cláusula FROM

Especifica as tabelas ou consultas que contêm os campos listados na instrução SELECT.

12.1. Sintaxe

SELECT listadecampos FROM expressãodetabela [IN bancodedadosexterno]

12.2. Uma instrução SELECT contendo uma cláusula FROM possui as partes a seguir:

Parte	Descrição
listadecampos	O nome do campo ou campos a serem recuperados juntamente com quaisquer aliases de nome de campo, funções agregadas SQL, atributos de seleção (ALL, DISTINCT, DISTINCTROW ou TOP) ou outras opções da instrução SELECT.
expressãodetabela	Uma expressão que identifica uma ou mais tabelas a partir das quais os dados são recuperados. A expressão pode ser um simples nome de tabela, um nome de consulta salvo ou uma composição resultante de um INNER JOIN, LEFT JOIN, ou RIGHT JOIN.
Bancodedadosexterno	O caminho completo de um banco de dados externo contendo todas as tabelas em expressãodetabela.

12.3. Comentários

FROM é exigido e segue-se a qualquer instrução SELECT.

A ordem dos nomes de tabela em expressãodetabela não é importante.

Para um melhor desempenho e facilidade de utilização, convém utilizar uma tabela vinculada em vez de uma cláusula IN para recuperar dados de um banco de dados externo.
O exemplo a seguir mostra como você pode recuperar os dados da tabela Funcionários:

SELECT Sobrenome, Nome FROM Funcionários;

13. Cláusula GROUP BY

Combina registros com valores idênticos na lista de campos especificada em um único registro. Um valor de resumo é criado para cada registro se você incluir uma função agregada SQL, como Sum ou Count, na instrução SELECT.

13.1. Sintaxe

SELECT listadecampos FROM tabela WHERE critérios
[GROUP BY listadecamposdegrupo]

13.2. Uma instrução SELECT contendo uma cláusula GROUP BY possui as partes a seguir:

Parte	Descrição
listadecampos	O nome do campo ou dos campos a serem recuperados juntamente com qualquer alias de nome de campo, funções agregadas SQL, atributos de seleção (ALL, DISTINCT, DISTINCTROW ou TOP) ou outras opções da instrução SELECT.
tabela	O nome da tabela a partir da qual os registros são recuperados. Para obter maiores informações, consulte a cláusula FROM.
critérios	Critérios de seleção. Se a instrução incluir uma cláusula WHERE, o mecanismo de banco de dados Microsoft Jet agrupará valores depois de aplicar as condições WHERE aos registros.
Listadecamposdegrupo	Os nomes de até 10 campos utilizados para agrupar os registros. A ordem dos nomes de campo em listadecamposdegrupo determina os níveis de agrupamento do nível mais alto ao mais baixo do agrupamento.

13.3. Comentários

GROUP BY é opcional.

Os valores de resumo são omitidos se não houver uma função agregada SQL na instrução SELECT.

Valores Null nos campos GROUP BY são agrupados e não são omitidos. Contudo, valores Null não são avaliados em nenhuma função SQL agregada.

Utilize a cláusula WHERE para excluir linhas que você não deseja que permaneçam agrupadas e utilize a cláusula HAVING para filtrar os registros depois de eles terem sido agrupados.

Todos os campos na lista de campos SELECT devem estar incluídos na cláusula GROUP BY ou serem incluídos como argumentos em uma função SQL gregada.

14. Cláusula HAVING

Especifica quais registros agrupados são exibidos na instrução SELECT com uma cláusula GROUP BY. Depois de GROUP BY combinar os registros, HAVING exibirá qualquer registro agrupado pela cláusula GROUP BY que satisfaça às condições da cláusula HAVING.

14.1. Sintaxe

SELECT listadecampos FROM tabela WHERE critériosdeseleção
GROUP BY listadecamposdegrupo [HAVING critériosdegrupo]

14.2. Uma instrução SELECT contendo uma cláusula HAVING possui as partes a seguir:

Parte	Descrição
listadecampos	O nome do campo ou campos a serem recuperados juntamente com qualquer alias de nome de campo, funções SQL agregadas, atributos de seleção (ALL, DISTINCT, DISTINCTROW ou TOP) ou outras opções da instrução SELECT. tabela O nome da tabela a partir da qual os registros são recuperados. Para obter maiores informações, consulte a cláusula FROM.
critériosdeseleção	Critérios de seleção. Se a instrução inclui uma cláusula WHERE, o mecanismo de banco de dados Microsoft Jet agrupa valores depois de aplicar as condições WHERE aos registros.
Listadecamposdegrupo	Os nomes de até 10 campos utilizados para agrupar registros. A ordem dos nomes de campo em listadecamposdegrupo determina os níveis de agrupamento do nível mais alto ao mais baixo de agrupamento.
critériosdegrupo	Uma expressão que determina quais registros agrupados exibir.

14.3. Comentários

HAVING é opcional.

HAVING é semelhante a WHERE, que determina quais registros são selecionados. Depois de os registros
serem agrupados com o GROUP BY, HAVING determina os registros a serem exibidos:

```
SELECT CódigoDaCategoria, Sum(UnidadesNoEstoque)
FROM Produtos
GROUP BY CódigoDaCategoria
HAVING Sum(UnidadesNoEstoque) > 100 And Like "BOS*";
```

Uma cláusula HAVING pode conter até 40 expressões vinculadas por operadores lógicos, como And e Or.

15. Cláusula IN

Identifica tabelas em qualquer banco de dados externo ao qual o mecanismo de banco de dados Microsoft Jet pode se conectar, como um banco de dados dBASE ou Paradox ou um banco de dados Microsoft Jet externo.

15.1. Sintaxe

Para identificar uma tabela de destino:

```
[SELECT | INSERT] INTO destino IN
{caminho | ["caminho" "tipo"] | ["" [tipo; DATABASE = caminho]]}
```

Para identificar uma tabela de origem:

```
FROM expressãodetabela IN
{caminho | ["caminho" "tipo"] | ["" [tipo; DATABASE = caminho]]}
```

15.2. Uma instrução SELECT contendo uma cláusula IN possui as partes a seguir:

Parte	Descrição
destino	O nome da tabela externa à qual os dados são inseridos.
expressãodetabela	O nome da tabela ou tabelas a partir das quais os dados são recuperados. Este argumento pode ser um único nome de tabela, uma consulta salva ou uma composição resultante de um INNER JOIN, LEFT JOIN

	ou RIGHT JOIN. caminho O caminho completo para o diretório ou arquivo contendo tabela.
tipo	O nome do tipo de banco de dados utilizado para criar tabela se um banco de dados não for um banco de dados Microsoft Jet (por exemplo, dBASE III, dBASE IV, Paradox 3.x ou Paradox 4.x).

15.3. Comentários

Você pode utilizar IN para se conectar a um único banco de dados externo de cada vez.
Em alguns casos, o argumento caminho refere-se ao diretório que contém os arquivos do banco de dados.
Por exemplo, ao trabalhar com tabelas de banco de dados dBASE, FoxPro ou Paradox, o argumento caminho especifica o diretório contendo os arquivos .dbf ou .db. O nome do arquivo de tabela deriva do argumento destino ou expressãodetabela.

Para especificar um banco de dados não-Microsoft Jet, anexe um ponto-e-vírgula (;) ao nome e coloque-o entre aspas simples (' ') ou duplas (" "). Por exemplo, tanto 'dBASE IV;' quanto "dBASE IV;" são aceitos.

Você também pode utilizar a palavra reservada DATABASE para especificar o banco de dados externo.
Por exemplo, as linhas a seguir especificam a mesma tabela:

... FROM Table IN "" [dBASE IV; DATABASE=C:\DBASE\DATA\SALES;];
... FROM Table IN "C:\DBASE\DATA\SALES" "dBASE IV;"

Para um melhor desempenho e facilidade de uso, utilize uma tabela vinculada em lugar de IN.

Você também pode utilizar a palavra reservada IN como um operador de comparação em uma expressão.

16. Cláusula ORDER BY

Classifica os registros resultantes de uma consulta em um campo ou campos especificados, em ordem crescente ou decrescente.

16.1. Sintaxe

SELECT listadecampos FROM tabela WHERE critériosdeseleção
[ORDER BY campo1 [ASC | DESC][, campo2 [ASC | DESC]][, ...]]]

16.2. Uma instrução SELECT contendo uma cláusula ORDER BY possui as partes a seguir:

Parte	Descrição
Listadecampos	O nome do campo ou campos a serem recuperados juntamente com qualquer alias de nome de campo, funções SQL agregadas, atributos de seleção (ALL, DISTINCT, DISTINCTROW ou TOP) ou outras opções da instrução SELECT.
tabela	O nome da tabela a partir da qual os registros são recuperados. Para obter maiores informações, consulte a cláusula FROM.
critériosdeseleção	Critérios de seleção. Se a instrução incluir uma cláusula WHERE, o mecanismo de banco de dados Microsoft Jet ordenará os valores depois de aplicar as condições WHERE aos registros.
campo1, campo2	Os nomes dos campos nos quais os registros devem ser classificados.

16.3. Comentários

ORDER BY é opcional. Entretanto, se desejar que os dados sejam exibidos em ordem classificada, então você deverá utilizar ORDER BY.

A ordem de classificação padrão é ascendente (de A a Z, de 0 a 9). Os dois exemplos a seguir classificam os nomes dos funcionários pela ordem do sobrenome:

SELECT Sobrenome, Nome FROM Funcionários ORDER BY Sobrenome;

SELECT Sobrenome, Nome FROM Funcionários ORDER BY Sobrenome ASC;

Para classificar em ordem decrescente (de Z a A, de 9 a 0), adicione a palavra reservada DESC ao final de cada campo que você queira classificar em ordem

decrescente. O exemplo a seguir seleciona os salários e os classifica em ordem decrescente:

SELECT Sobrenome, Salário FROM Funcionários ORDER BY Salário DESC, Sobrenome;

Se você especificar um campo contendo dados de Memorando ou Objeto OLE na cláusula ORDER BY, um erro será gerado. O mecanismo de banco de dados Microsoft Jet não classifica campos desses tipos.

ORDER BY é geralmente o último item em uma instrução SQL.

Você pode incluir campos adicionais na cláusula ORDER BY. Os registros são classificados pelo primeiro campo listado após ORDER BY. Os registros que têm valores iguais naquele campo serão então classificados pelo valor no segundo campo listado e assim por diante.

17. Cláusula WHERE

Especifica quais registros das tabelas listadas na cláusula FROM são afetados por uma instrução SELECT, UPDATE ou DELETE.

Operadores que podem ser utilizados pela cláusula WHERE:

Operadores Aritméticos:

*	multiplicação	/	divisão
+	soma	-	subtração

Operadores relacionais:

>	maior	>=	maior igual
<	menor	<=	menor igual
<>	diferente	=	igual

Operadores lógicos:

and	(e)	or	(ou)	not	(negação)

17.1. Sintaxe

SELECT listadecampos FROM expressãodetabela WHERE critérios

17.2. Uma instrução SELECT contendo uma cláusula WHERE possui as partes a seguir:

Parte	Descrição
listadecampos	O nome do campo ou campos a serem recuperados juntamente com quaisquer aliases de nome de campo, atributos de seleção (ALL, DISTINCT, DISTINCTROW ou TOP) ou outras opções da instrução SELECT.
expressãodetabela	O nome da tabela ou tabelas a partir das quais os dados são recuperados.
critérios	Uma expressão que os registros devem atender para serem incluídos nos resultados da consulta.

17.3. Comentários

O mecanismo de banco de dados Microsoft Jet seleciona os registros que satisfazem às condições listadas na cláusula WHERE. Se você não especificar uma cláusula WHERE, a consulta retornará todas as linhas da tabela. Se você especificar mais de uma tabela na consulta e não tiver incluído uma cláusula WHERE ou uma cláusula JOIN, a consulta irá gerar um produto cartesiano das tabelas.

WHERE é opcional, mas quando incluído, se segue ao FROM. Por exemplo, você pode selecionar todos os funcionários no departamento de vendas (WHERE Depto = 'Vendas') ou todos os clientes com idades entre 18 e 30 anos (WHERE Idade Between 18 And 30).

Se você não utilizar uma cláusula JOIN para executar operações de associação SQL em várias tabelas, o objeto Recordset resultante não será atualizável.

WHERE é semelhante a HAVING. WHERE determina quais registros são selecionados. Da mesma forma, depois de os registros serem agrupados com GROUP BY, HAVING determina quais registros serão exibidos.

Utilize a cláusula WHERE para eliminar os registros que você não deseja que sejam agrupados por uma cláusula GROUP BY.

Utilize várias expressões para determinar quais registros a instrução SQL retorna. Por exemplo, a instrução SQL a seguir seleciona todos os funcionários cujos salários sejam maiores que R$21.000:

SELECT Sobrenome, Salário FROM Funcionários WHERE Salário > 21000;

Uma cláusula WHERE pode conter até 40 expressões vinculadas por operadores lógicos, como And e Or.

Ao digitar um nome de campo que contém um espaço ou pontuação, coloque o nome entre colchetes ([ ]).
Por exemplo, uma tabela de informações sobre o cliente poderia incluir informações sobre clientes específicos:

SELECT [Restaurante Favorito do Cliente]

Quando você especifica o argumento critérios, os literais de data deverão estar no formato dos EUA, mesmo que você não esteja utilizando a versão norte-americana do mecanismo de banco de dados Microsoft Jet. Por exemplo, 10 de maio de 1996 é escrito 10/5/96 no Brasil e 5/10/96 nos Estados Unidos.

Para encontrar registros datados de 10 de maio de 1996 em um banco de dados dos Estados Unidos, você deve utilizar a instrução SQL a seguir:

SELECT * FROM Pedidos WHERE DataDeEnvio = '5/10/96';

18.1. Sintaxe

instruçãosql WITH OWNERACCESS OPTION

18.2. Comentários

A declaração WITH OWNERACCESS OPTION é opcional.

O exemplo a seguir permite que o usuário visualize informações sobre salários (mesmo que, de outra forma, o usuário não tenha permissão para visualizar a tabela FolhaDePagamento), desde que o proprietário da consulta tenha essa permissão:

SELECT Sobrenome, Nome, Salário FROM Funcionários ORDER BY Sobrenome WITH OWNERACCESS OPTION;

Se de alguma outra forma o usuário for impedido de criar ou adicionar a uma tabela, você pode usar WITH OWNERACCESS OPTION para permitir que ele execute uma consulta criar tabela ou consulta acréscimo.

Se você deseja impor configurações de segurança de grupo de trabalho e permissões de usuários, não inclua a declaração WITH OWNERACCESS OPTION.

Essa opção requer que você tenha acesso ao arquivo System.mdw associado ao banco de dados. É realmente útil somente nas implementações de multiusuários com segurança.

19. Funções agregadas SQL

Utilizando os SQL funções agregadas, você pode determinar várias estatísticas em conjuntos de valores. Você pode utilizar estas funções em uma consulta e em expressões agregadas na propriedade SQL de um objeto QueryDef ou ao criar um objeto Recordset baseado em uma consulta SQL.

Funções Agregadadas

Função Avg
Função Count
Funções Min, Max
Funções StDev, StDevP
Função Sum
Funções Var, VarP

20. Função Avg

Calcula a média aritmética de um conjunto de valores contido em um campo especificado em uma consulta.

20.1. Sintaxe

Avg(expr)

O espaço reservado expr representa uma expressão de seqüência que identifica o campo que contém os dados numéricos dos quais você quer tirar uma média ou uma expressão que realiza um cálculo utilizando os dados naquele campo. Os operandos em expr podem incluir o nome de um campo de tabela, uma constante ou uma função (que pode ser intrínseca ou definida pelo usuário, mas nenhuma das outras funções agregadas SQL).

20.2. Comentários

A média calculada por Avg é a média aritmética (a soma dos valores dividida pelo número de valores). Você poderia utilizar o Avg, por exemplo, para calcular o custo médio do frete.

A função Avg não inclui nenhum campo Null no cálculo.

Você pode utilizar Avg em uma expressão de consulta e na propriedade SQL de um objeto QueryDef ou ao criar um objeto Recordset baseado em uma consulta SQL.

21. Função Count

Calcula o número de registros retornado por uma consulta.

21.1. Sintaxe

Count(expr)

O espaço reservado expr representa uma expressão de seqüência de caracteres que identifica o campo que contém os dados que você quer contar ou uma expressão que realiza um cálculo utilizando os dados no campo. Os operandos em expr podem incluir o nome de um campo de tabela ou função (que pode ser intrínseca ou definida pelo usuário, mas nenhuma outra função agregada SQL). Você pode contar qualquer tipo de dados, inclusive texto.

21.2. Comentários

Você pode utilizar Count para contar o número de registros em uma consulta base. Por exemplo, você poderia utilizar Count para contar o número de pedidos enviados a um determinado país. Embora expr possa realizar um cálculo em um campo, Count simplesmente computa o número de registros, independentemente dos valores armazenados nos registros.

A função Count não conta registros que tenham campos Null, exceto quando expr for o caractere curinga asterisco (*). Se você utilizar um asterisco, Count calculará o número total de registro, inclusive aqueles que contêm campos Null. Count(*) é consideravelmente mais rápido que Count([Nome da Coluna]). Não coloque o asterisco entre aspas (' '). O exemplo a seguir calcula o número de registros na tabela Pedidos:

```
SELECT Count(*) AS TotalDePedidos FROM Pedidos;
```

Se expr identificar vários campos, a função Count contará um registro somente se um dos campos não for Null. Se todos os campos especificados forem Null, o registro não será contado. Separe os nomes de campo com um e-comercial (&). O exemplo a seguir mostra como você poderia limitar a contagem aos registros nos quais DataDeEnvio ou Frete não fosse Null:

```
SELECT Count('DataDeEnvio & Frete') AS [Not Null] FROM Pedidos;
```

Você pode utilizar Count em uma expressão de consulta. Você também pode utilizar esta expressão na propriedade SQL de um objeto QueryDef ou durante a criação de um objeto Recordset baseado em uma consulta SQL.

22. Função Sum

Retorna a soma de um conjunto de valores contido em um campo especificado em uma consulta.

22.1. Sintaxe

Sum(expr)

O espaço reservado expr representa uma expressão de seqüência que identifica o campo que contém os dados numéricos que você quer adicionar ou uma expressão que realiza um cálculo utilizando os dados naquele campo. Os operandos em expr podem incluir o nome de um campo de tabela, uma constante ou uma função (que pode ser intrínseca ou definida pelo usuário, mas nenhuma das outras funções agregadas SQL).

22.2. Comentários

A função Sum totaliza os valores em um campo. Por exemplo, você poderia utilizar a função Sum para determinar o custo total dos encargos de frete.

A função Sum ignora os registros que contenham campos Null. O exemplo a seguir mostra como você pode calcular a soma dos produtos dos campos PreçoUnitário e Quantidade:

SELECT Sum(PreçoUnitário * Quantidade)AS [Receita Total] FROM [Detalhes do Pedido];

Você pode utilizar a função Sum em uma expressão de consulta. Você também pode utilizar esta expressão na propriedade SQL de um objeto QueryDef ou durante a criação de um objeto Recordset, com base em uma consulta SQL.

23. Operação INNER JOIN

Combina registros de duas tabelas sempre que houver valores correspondentes em um campo comum.

23.1. Sintaxe

FROM tabela1 INNER JOIN tabela2 ON tabela1.campo1 opercomp tabela2.campo2

23.2. A operação INNER JOIN possui as partes a seguir:

Parte	Descrição
tabela1, tabela2	Os nomes das tabelas das quais os registros são combinados.

campo1, campo2	Os nomes dos campos que são associados. Se não forem numéricos, os campos deverão ser do mesmo tipo de dados e conter o mesmo tipo de dados, mas não precisarão ter o mesmo nome.
opercomp	Qualquer operador de comparação relacional: "=," "<," ">," "<=," ">=," ou "<>."

23.3. Comentários

Você pode usar uma operação INNER JOIN em qualquer cláusula FROM. Este é o tipo mais comum de associação. As associações internas combinam registros de duas tabelas sempre que houver valores correspondentes em um campo comum a ambas.

Você pode usar INNER JOIN com as tabelas Departamentos e Funcionários para selecionar todos os funcionários em cada departamento. Em contrapartida, para selecionar todos os departamentos (mesmo que alguns não tenham funcionários designados para eles) ou todos os funcionários (mesmo que alguns não sejam designados a um departamento), você pode usar uma operação LEFT JOIN ou RIGHT JOIN para criar uma associação externa.

Se você tentar associar campos contento dados de Memorando ou Objeto OLE, ocorrerá um erro.

Você pode associar quaisquer dois campos numéricos de tipos semelhantes. Por exemplo, você pode associar em campos AutoNumeração e Longo pois são tipos semelhantes. Entretanto, você não podeassociar os tipos de campo Único e Duplo.

O exemplo a seguir mostra como você poderia associar as tabelas Categorias e Produtos no campo CódigoDaCategoria:

SELECT NomeDaCategoria, NomeDoProduto FROM Categorias INNER JOIN Produtos ON Categorias.CódigoDaCategoria = Produtos.CódigoDaCategoria;

No exemplo acima, CódigoDaCategoria é o campo associado, mas não é incluído na saída da consulta pois não está incluído na instrução SELECT. Para incluir o campo associado, inclua o nome do campo na instrução SELECT neste caso, Categorias.CódigoDaCategoria.

Você pode também vincular várias cláusulas ON em uma instrução JOIN, usando a sintaxe a seguir:

SELECT campos FROM tabela1 INNER JOIN tabela2
ON tabela1.campo1 opercomp tabela2.campo1 AND
ON tabela1.campo2 opercomp tabela2.campo2) OR

ON tabela1.campo3 opercomp tabela2.campo3)];

Você pode também aninhar instruções JOIN usando a seguinte sintaxe:

SELECT campos FROM tabela
INNER JOIN (tabela2 INNER JOIN [( ]tabela3
[INNER JOIN [( ]tabelax [INNER JOIN ...)]
ON tabela3.campo3 opercomp tabelax.campox)]
ON tabela2.campo2 opercomp tabela3.campo3)
ON tabela1.campo1 opercomp tabela2.campo2;

Uma instrução LEFT JOIN ou RIGHT JOIN pode ser aninhada dentro de uma INNER JOIN, mas uma INNER JOIN não pode ser aninhada dentro de uma LEFT JOIN ou de uma RIGHT JOIN.

24. Operações LEFT JOIN, RIGHT JOIN

Combina registros da tabela de origem quando usados em qualquer cláusula FROM.

24.1. Sintaxe

FROM tabela1 [LEFT | RIGHT] JOIN tabela2
ON tabela1.campo1 opercomp tabela2.campo2

24.2. As operações LEFT JOIN e RIGHT JOIN possuem as partes a seguir:

Parte	Descrição
tabela1, tabela2	Os nomes das tabelas das quais os registros são combinados.
campo1, campo2	Os nomes dos campos que são associados. Os campos devem ser do mesmo tipo de dados e conter o mesmo tipo de dados, mas não precisam ter o mesmo nome.
opercomp	Qualquer operador de comparação relacional: "=," "<," ">," "<=," ">=," ou "<>."

24.3. Comentários

Utilize uma operação LEFT JOIN para criar uma associação externa esquerda. As associações externas esquerdas incluem todos os registros da primeira

(esquerda) de duas tabelas, mesmo que não haja valores correspondentes para os registros na segunda tabela (direita).

Utilize uma operação RIGHT JOIN para criar uma associação externa direita. As associações externas direitas incluem todos os registros da segunda (direita) de duas tabelas, mesmo que não haja valores correspondentes para registros na primeira (esquerda) tabela.

Por exemplo, você poderia usar LEFT JOIN com as tabelas Departamentos (esquerda) e Funcionários (direita) para selecionar todos os departamentos, inclusive os que não tenham funcionários designados para eles. Para selecionar todos os funcionários, inclusive os que não estão designados para um departamento, você usaria RIGHT JOIN.

O exemplo a seguir mostra como você poderia associar as tabelas Categorias e Produtos no campo CódigoDaCategoria. A consulta produz uma lista de todas as categorias, inclusive aquelas que não contêm produtos:

```
SELECT NomeDaCategoria, NomeDoProduto FROM Categorias
LEFT JOIN Produtos
ON Categorias.CódigoDaCategoria = Produtos.CódigoDaCategoria;
```

Neste exemplo, CódigoDaCategoria é o campo associado, mas não é incluído nos resultados da consulta pois não está incluído na instrução SELECT. Para incluir o campo associado, digite o nome do campo na instrução SELECT ¾ neste caso, Categorias.CódigoDaCategoria.

Para criar uma consulta que inclua somente registros nos quais os dados nos campos associados sejam os mesmos, utilize uma operação INNER JOIN.

Uma operação LEFT JOIN ou RIGHT JOIN pode ser aninhada dentro de uma operação INNER JOIN, mas uma INNER JOIN não pode ser aninhada dentro de uma LEFT JOIN ou de uma RIGHT JOIN. Consulte a matéria sobre como aninhar no tópico INNER JOIN para ver como aninhar associações dentro de outras associações.

Você pode vincular várias cláusulas ON. Consulte a matéria sobre vinculação de cláusulas no tópico INNER JOIN para ver como isso é feito.

Se você tentar associar campos contendo dados de Memorando ou Objeto OLE, ocorrerá um erro.

25. Operação UNION

Cria uma consulta união, que combina os resultados de duas ou mais consultas ou tabelas independentes.

25.1. Sintaxe

[TABLE] consulta1 UNION [ALL] [TABLE] consulta2 [UNION [ALL] [TABLE] consultan [...]]

25.2. A operação UNION possui as partes a seguir:

Parte	Descrição
consulta1-n	Uma instrução SELECT, o nome de uma consulta armazenada ou o nome de uma tabela armazenada precedida da palavra-chave TABLE.

Você pode mesclar os resultados de duas ou mais consultas, tabelas e instruções SELECT, em qualquer combinação, em uma única operação UNION. O exemplo a seguir mescla uma tabela existente denominada Novas Contas com uma instrução SELECT:

```
TABLE [Novas Contas] UNION ALL SELECT * FROM Clientes
WHERE QuantiaDoPedido > 1000;
```

Como padrão, nenhum registro duplicado é retornado quando você usa uma operação UNION; entretanto, você pode incluir o atributo ALL para assegurar que todos os registros sejam retornados. Isso faz com que a execução da consulta seja mais rápida.

Todas as consultas em uma operação UNION devem solicitar o mesmo número de campos; contudo, os campos não deverão ter o mesmo tamanho ou tipo de dados.

Use aliases somente na primeira instrução SELECT pois eles são ignorados em qualquer outra. Na cláusula ORDER BY, refira-se aos campos pelo que são chamados na primeira instrução SELECT.

Você pode usar uma cláusula GROUP BY ou HAVING em cada argumento consulta para agrupar os dados retornados.

Você pode usar uma cláusula ORDER BY no fim do último argumento consulta para exibir os dados retornados em uma ordem especificada.

26. Subconsultas SQL

Uma subconsulta é uma instrução SELECT aninhada dentro de uma instrução SELECT, SELECT...INTO, INSERT...INTO, DELETE ou UPDATE ou de uma outra subconsulta.

26.1. Sintaxe

Você pode usar três formas de sintaxe para criar uma subconsulta:

comparação [ANY | ALL | SOME] (instruçãosql)
expressão [NOT] IN (instruçãosql)
[NOT] EXISTS (instruçãosql)

26.2. Uma subconsulta possui as partes a seguir:

Parte	Descrição
comparação	Uma expressão e um operador de comparação que compara a expressão com os resultados da subconsulta.
expressão	Uma expressão para a qual o conjunto de resultados da subconsulta é procurado.
instruçãosql	Uma instrução SELECT, que segue o mesmo formato e regras de qualquer outra instrução SELECT. Deve estar entre parênteses.

26.3. Comentários

Você pode usar uma consulta em lugar de uma expressão na lista de campos de uma instrução SELECT ou ema cláusula WHERE ou HAVING. Em uma subconsulta, você usa uma instrução SELECT para proporcionar um conjunto de um ou mais valores específicos para avaliar na expressão de cláusula WHERE ou HAVING.

Use o atributo ANY ou SOME, que são sinônimos, para recuperar registros na consulta principal que satisfaçam a comparação com qualquer registro recuperado na subconsulta. O exemplo a seguir retorna todos os produtos cujo preço unitário é maior que o de qualquer produto vendido com um desconto de 25% ou maior:

```
SELECT * FROM Produtos WHERE PreçoUnitário > ANY
(SELECT PreçoUnitário FROM DetalhesDoPedido WHERE Desconto >= .25);
```

Use o atributo ALL para recuperar somente os registros da consulta principal que satisfaçam a comparação com todos os registros recuperados na subconsulta. Se você tivesse alterado ANY para ALL no exemplo anterior, a consulta retornaria somente os produtos cujo preço unitário fosse maior que o de todos os produtos vendidos com um desconto de 25% ou maior. Isso é bem mais restritivo.

Use o atributo IN para recuperar somente os registros da consulta principal para os quais algum registro na subconsulta contenha um valor igual. O exemplo a seguir retorna todos os produtos com um desconto de 25% ou maior:

```
SELECT * FROM Produtos WHERE CódigoDoProduto IN
(SELECT CódigoDoProduto FROM DetalhesDoPedido
```

WHERE Desconto >= .25);

Da mesma forma, você pode usar NOT IN para recuperar somente os registros na consulta principal para os quais nenhum registro na subconsulta contenha um valor igual.

Use o atributo EXISTS (com a palavra reservada NOT opcional) em comparações verdadeiro/falso para determinar se a subconsulta retorna algum registro.

Você pode também usar aliases de nome de tabela em uma consulta para se referir a tabelas relacionadas em uma cláusula FROM fora da subconsulta. O exemplo a seguir retorna os nomes dos funcionários cujos salários são iguais ou maiores do que o salário médio de todos os funcionários que têm o mesmo cargo. A tabela Funcionários recebe o alias "T1":

SELECT Sobrenome, Nome, Título, Salário FROM Funcionários AS T1
WHERE Salário >= (SELECT Avg(Salário) FROM Funcionários
WHERE T1.Título = Funcionários.Título) Order by Título;

No exemplo anterior, a palavra reservada AS é opcional.

Algumas subconsultas são permitidas em consultas de tabela de referência cruzada — especificamente, como atributos (os da cláusula WHERE). Subconsultas como saída (as da lista SELECT) não são permitidas em consultas de tabela de referência cruzada.

27. Instrução TRANSFORM

Cria uma consulta tabela de referência cruzada.

27.1. Sintaxe

TRANSFORM funçãoagrg instruçãoselect
PIVOT campocentral [IN (valor1[, valor2[, ...]])]

27.2. A instrução TRANSFORM possui as partes a seguir:

Parte	Descrição
funçãoagrg	Uma função agregada SQL que opera sobre os dados selecionados.
instruçãoselect	Uma instrução SELECT.

Campodinâmicol	O campo ou expressão que você quer usar para criar títulos de coluna no conjunto de resultados da consulta.
valor1, valor2	valores fixos usados para criar títulos de colunas.

27.3. Comentários

Quando resume dados utilizando uma consulta tabela de referência cruzada, você seleciona valores de campos ou expressões especificadas como títulos de colunas para poder visualizar os dados em um formato mais compacto do que com uma consulta seleção.

A instrução TRANSFORM é opcional, mas quando for incluída será a primeira instrução em uma seqüência SQL. Ela antecede uma instrução SELECT que especifica os campos usados como títulos de linhas e uma cláusula GROUP BY que especifica o agrupamento de linhas. Opcionalmente, você pode incluir outras cláusulas, como WHERE, que especifiquem critérios adicionais de seleção ou de classificação. Você pode também usar subconsultas como atributos — especificamente, aqueles em uma cláusula WHERE — em uma consulta tabela de referência cruzada.

Os valores retornados em campodinâmico são usados como títulos de colunas no conjunto de resultados da consulta. Por exemplo, articular os números das vendas no mês da venda em uma consulta tabela de referência cruzada criaria 12 colunas. Você pode restringir campodinâmico para criar títulos a partir de valores fixos

(valor1, valor2) relacionados na cláusula IN opcional. Você pode também incluir valores fixos para os quais não existam dados para criar colunas adicionais.

28. Declaração PARAMETERS

Declara o nome e o tipo de dados de cada parâmetro em uma consulta parâmetro.

28.1. Sintaxe

PARAMETERS nome tipodedados [, nome tipodedados [, ...]]

28.2. A declaração PARAMETERS possui as partes a seguir:

Parte	Descrição
nome	O nome do parâmetro. Atribuído à propriedade Name do objeto Parameter e usado para identificar esse parâmetro na

	coleção Parameters. Você pode usar nome como uma seqüência de caracteres que é exibida em uma caixa de diálogo enquanto o aplicativo executa a consulta. Use colchetes ([ ]) para envolver o texto que contém espaços ou pontuação. Por exemplo, [Preço baixo] e [Começar relatório com que mês?] são argumentos nome válidos.
tipodedados	Um dos tipos de dados SQL do Microsoft Jet primários ou seus sinônimos.

28.3. Comentários

Para consultas executadas regularmente, você pode utilizar uma declaração PARAMETERS para criar uma consulta parâmetro. Uma consulta parâmetro pode ajudar a automatizar o processo de alteração dos critérios da consulta. Em uma consulta parâmetro, o código precisará fornecer os parâmetros a cada vez que a consulta for executada.

A declaração PARAMETERS é opcional, mas quando incluída precede qualquer outra instrução, inclusive SELECT.

Se a declaração incluir mais de um parâmetro, separe-os com vírgulas. O exemplo a seguir inclui dois parâmetros:

```
PARAMETERS [Preço baixo] Currency, [Data inicial] DateTime;
```

Você pode usar nome, mas não tipodedados em uma cláusula WHERE ou HAVING. O exemplo a seguir espera que dois parâmetros sejam fornecidos e, então, aplica os critérios aos registros na tabela Pedidos:

```
PARAMETERS [Preço baixo] Currency, [Data inicial] DateTime;
SELECT NúmeroDoPedido, QuantiaDoPedido
FROM Pedidos
WHERE QuantiaDoPedido > [Preço baixo] AND DataDoPedido >= [Data
inicial];
```

29. Operador Between...And

Determina se o valor de uma expressão se situa dentro de um intervalo especificado de valores. Você pode utilizar este operador em instruções SQL.

29.1. Sintaxe

```
expr [Not] Between valor1 And valor2
```

29.2. A sintaxe do operador Between...And possui as partes a seguir:

Parte	Descrição

expr	Expressão que identifica o campo que contém os dados a serem avaliados.
valor1, valor2	Expressões em relação as quais você deseja avaliar expr.

29.3. Comentários

Se o valor de expr estiver entre valor1 e valor2 (inclusive), o operador Between...And retornará True; caso contrário, retornará False. Você pode incluir o operador lógico Not para avaliar a condição oposta (isto é, se expr estiver situado fora do intervalo definido por valor1 e valor2).

Você poderia utilizar Between...And para determinar se o valor de um campo está situado em um intervalo numérico especificado. O exemplo a seguir determina se um pedido foi enviado a um local situado em um intervalo de códigos postais. Se o código postal estiver entre 98101 e 98199, a função IIf retornará "Local". Caso contrário, retornará "Nãolocal".

```
SELECT IIf(CódigoPostal Between 98101 And 98199, "Local", "Nãolocal")
FROM Editores
```

Se expr, valor1 ou valor2 forem Null, Between...And retornará um valor Null.

Uma vez que os caracteres curinga, como *, são tratados como literais, você não pode utilizá-los com o operador Between...And. Por exemplo, você não pode utilizar 980* e 989* para localizar todos os códigos postais que começam com 980 e 989. Em vez disso, você tem duas alternativas: pode adicionar uma expressão para a consulta que pegue os três caracteres da esquerda do campo de texto e utilizar Between...And nesses caracteres, ou pode preencher os valores superior e inferior com caracteres extras — neste caso, 98000 a 98999, ou 98000 a 98999 – 9999 se estiver utilizando códigos postais estendidos.
(Você deve omitir o – 0000 dos valores inferiores pois, caso contrário, 98000 será excluído se alguns códigos postais tiverem seções e outros não.)

30. Operador In

Determina se o valor de uma expressão é igual a algum dos vários valores em uma lista especificada.

30.1. Sintaxe

```
expr [Not] In(valor1, valor2, . . .)
```

30.2. A sintaxe do operador In possui as partes a seguir:

Parte	Descrição
expr	Expressão que identifica o campo que contém os dados a serem avaliados.

valor1, valor2	Expressão ou lista de expressões em relação a qual você deseja avaliar expr.

Se expr for encontrado na lista de valores, o operador In retornará True; caso contrário, retornará False. Você pode incluir o operador lógico Not para avaliar a condição oposta (isto é, se expr não está na lista de valores).

Por exemplo, você pode utilizar In para determinar os pedidos a serem enviados a um conjunto de regiões especificadas:

```
SELECT * FROM Pedidos
WHERE RegiãoDeRemessa In ('Avon','Glos','Som')
```

31. Operador Like

Compara uma expressão de seqüência com um padrão em uma expressão SQL.

31.1. Sintaxe

expressão Like "padrão"

31.2. A sintaxe do operador Like possui as partes a seguir:

Parte	Descrição
expressão	Expressão SQL utilizada em uma cláusula WHERE.
padrão	Seqüência ou literal de seqüência de caracteres em relação a qual expressão é comparada.

31.3. Comentários

Você pode utilizar o operador Like para localizar valores em um campo que correspondam ao padrão especificado. Para padrão, você pode especificar o valor completo (por exemplo, Like "Smith") ou pode utilizar caracteres curinga para encontrar um intervalo de valores (por exemplo, Like "Sm*").

Em uma expressão, você pode utilizar o operador Like para comparar um valor de campo a uma expressão de seqüência. Por exemplo, se você digitar Like "C*" em uma consulta SQL, a consulta retornará todos os valores de campo que começam com a letra C. Em uma consulta parâmetro, você pode solicitar ao usuário que forneça um padrão pelo qual procurar.

O exemplo a seguir retorna dados que começam com a letra P, seguida por qualquer letra entre A e F e três dígitos:

Like "P[A-F]###"

A tabela a seguir mostra como você pode utilizar Like para testar expressões para diferentes padrões.

Tipo de correspondência

Padrão Coincidente (retorna True) Não coincidente (retorna False)
Vários caracteres aZb, bac a*a aa, aBa, aBBBa aBC *ab* abc, AABB, Xab
Caractere especial a[*]a a*a aaa
Vários caracteres ab* abcdefg, abc cab, aab
Um único caractere a?a aaa, a3a, aBa aBBBa

Um único dígitoa#a a0a, a1a, a2a aaa, a10a

Intervalo de caracteres [a-z] f, p, j 2, &

Fora de um intervalo [!a-z] 9, &, % b, a

Nenhum dígito [!0-9] A, a, &, ~ 0, 1, 9

Combinado a[!b-m]# An9, az0, a99 abc, aj0

32. CARACTERES CURINGA

A correspondência interna de padrões oferece uma ferramenta versátil para fazer comparações de seqüências. A tabela a seguir mostra os caracteres curinga que você pode utilizar com o operador Like e o número de dígitos ou seqüências aos quais eles podem corresponder.

Caractere(s)
em padrão
Coincide com expressão

? Qualquer caractere isolado

* Zero ou mais caracteres
# Qualquer dígito isolado (0 — 9)
[listadecaract] Qualquer caractere isolado em listadecaract
[!listadecaract] Qualquer caractere isolado não-presente em listadecaract
Você pode utilizar um grupo de um ou mais caracteres (listadecaract) entre colchetes ([ ]) para coincidir com qualquer caractere isolado em expressão, e listadecaract pode incluir praticamente qualquer caractere do conjunto de caracteres ANSI, inclusive dígitos. De fato, você pode utilizar os caracteres

especiais colchete de abertura ([ ), ponto de interrogação (?), sinal de número (#) e asterisco (*) para que correspondam diretamente a eles mesmos somente se estiverem entre colchetes. Você pode utilizar o colchete de fechamento ( ]) dentro de um grupo para que corresponda a ele mesmo, mas pode utilizá-lo fora de um grupo como um caractere individual.
Além de uma simples lista de caracteres entre colchetes, listadecaract pode especificar um intervalo de caracteres através da utilização de um hífen (-) para separar os limites superior e inferior do intervalo. Por exemplo, a utilização de [A-Z] em padrão resultará em uma correspondência se a posição do caractere correspondente em expressão contiver qualquer uma das letras maiúsculas no intervalo de A a Z. Você pode incluir vários intervalos entre os colchetes sem delimitar os intervalos. Por exemplo, [a-zA-Z0-9] corresponde a qualquer caractere alfanumérico.

Outras regras importantes para correspondência de padrão incluem:

Um ponto de exclamação (!) no início de listadecaract significa que uma correspondência será feita se qualquer caractere, exceto aqueles na listadecaract, for encontrado em expressão. Quando utilizado sem colchetes, o ponto de exclamação corresponderá a si mesmo.

Você pode utilizar o hífen (-) seja no início (depois de um ponto de exclamação, se este for utilizado) ou no fim de listadecaract para corresponder a si mesmo. Se estiver em qualquer outro local, o hífen identificará um intervalo de caracteres ANSI.

Quando você especifica um intervalo de caracteres, os caracteres devem aparecer em ordem de classificação crescente (A-Z ou 0-100). [A-Z] é um padrão válido, mas [Z-A] não é.

A seqüência de caracteres [ ] é ignorada; ela é considerada uma seqüência de comprimento zero ("").