Banco de Dados
Data Manipulation Language

Comando SELECT

- Seleciona colunas e linhas de uma tabela
- Três palavras chaves compõe o comando
  - SELECT → especifica as colunas
  - FROM → especifíca as tabelas
  - WHERE → especifica o filtro para as linhas
- SELECT * exibe todas as colunas
- SELECT sem a claúsula WHERE exibe todas as linhas

Sintaxe

```
SELECT [ALL|DISTINCT] select_list
      INTO [new_table_name]
FROM table_name|view_name, table_name2|view_name2 …
WHERE clausula
GROUP BY clausula
HAVING clausula
ORDER BY clausula
COMPUTE clausula
```

SELECT *

Sintaxe `SELECT * from table_name`
Exemplo `SELECT * from authors`

*O commando SELECT * sem a clausula WHERE sobrecarrega desnecessáriamente os recursos do sistema. Por essa razão, é aconselhável sempre incluir a clausula WHERE.*

Escolhendo as colunas

Sintaxe `SELECT column_name, column_name ... FROM table_name`
Exemplo `SELECT au_id, au_fname, au_lname FROM authors`

Alterando a ordem das colunas

Sintaxe `SELECT column_name, column_name ... FROM table_name`
Exemplo `SELECT au_fname, au_lname, au_id  FROM authors`

Usando Literais

Sintaxe `SELECT column_name|'string literal', column_name|'string literal' … FROM table_name`

Exemplo `SELECT au_fname, au_lname, 'Identification number:', au_id FROM authors`

Alterando o cabeçalho das colunas

Sintaxe

```
SELECT column_header = column_name, column_header = column_name, FROM table_name
SELECT column_name column_header,    column_name column_header FROM table_name
```

Exemplo `SELECT FIRST = au_fname, LAST = au_lname, Identification# = au_id FROM authors`

Manipulação de dados Numéricos

- Operadores Matemáticos

Simbolo	Operação	Tipos de Dados
+	Adição	Int, smallint, tinyint, numeric, decimal, float, real, money e smallmoney
-	Subtração	Int, smallint, tinyint, numeric, decimal, float, real, money e smallmoney
/	Divisão	Int, smallint, tinyint, numeric, decimal, float, real, money e smallmoney
*	Multiplicação	Int, smallint, tinyint, numeric, decimal, float, real, money e smallmoney
%	Módulo	Int, smallint e tinyint

Sintaxe

```
{constante|column_name|function|subquery}
[{arithmetic_operator|string_operator|AND|OR|NOT}
{constante|column_name|function|subquery}]
```

Considerações

- Operadores matemáticos podem executar operações em colunas numéricas ou em constantes numéricas.
- O módulo (resto) não pode ser executado em tipos de dados money, smallmoney, float ou real.

Precedência de execução

Grupo Primário ( )

Multiplicação * / %

Adição - +

Quando todos os operadores em uma expressão tem o mesmo nível de precedência, a ordem de execução é da esquerda para a direita.

Exemplo `select price, (price*1.1), title FROM titles`

Manipulação de dados Caracter

Dado caracter consiste em qualquer combinação de letras, simbolos e numeros. A maioria das funções podem apenas ser utilizadas em tipos de dado char e varchar e algumas poucas em tipos binary e varbinary. O dado caracter deve estar fechado entre apostrofos.
d
Sintaxe `SELECT function_name(parameters)`

Função	Parametros	Resultado
+	(expressão expressão)	Concatena dois ou mais caracteres ou binary strings ou colunas
ASCII	(char_expr)	Retorna o valor ASCII do caracter mais a esquerda
CHAR	(integer_expr)	Retorna o caracter corresponde de um código ASCII
CHARINDEX	('pattern',expr)	Retorna a posição inicial de pattern em expr
DIFFERENCE	(char_expr1,char_expr2)	Compara e avalia a similiaridade entre ambas, retornando de 0 a 4.
LOWER	(char_expr)	Converte para caixa baixa (minúscula)
LTRIM	(char_expr)	Remove caracteres em branco a esquerda
REPLICATE	(char_expr,int_expr)	Repete a expressão o numero de vezes especificado
RIGHT	(char_expr,int_expr)	Parte um dado caracter iniciando de int_expr caracteres a direita
RTRIM	(char_expr)	Remove caracteres em branco a direita
STR	(float_expr,length,decimal)	Converte um numero em string
STUFF	(char_expr1,start,len,char_expr2)	Troca os a partir de start por len posições pelo valor de char_expr2
SUBSTRING	(char_expr, start,len)	Retorna os len caracteres constantes em char_expr a partir de start
UPPER	(char_expr)	Converte para caixa alta (maiúscula)

Exemplos

```
SELECT STUFF('1234567',3,2,'xxxx')

SELECT 'Author Name:' + space(2), au_lname FROM pubs..authors
```

Banco de Dados
Data Manipulation Language

Manipulação de dados Datetime

Sintaxe `SELECT date_function(parameters)`

Função	Parametros	Resultado
DATEADD	(datepart,number,date)	Adiciona o number de datepart em date
DATEDIFF	(datepart, date1,date2)	Numero de dateparts entre as datas informadas
DATENAME	(datepart,date)	Retorna o datepart especificado para um dado valor ASCII (date) – string
DATEPART	(datepart,date)	Retorna o datepart especificado para uma data data – int
GETDATE	()	Date e hora corrente do servidor

Dateparts

Datepart	Abreviação	Valores
year	yy	1753-9999
quarter	qq	1 à 4
month	mm	1 à 12
dayofyear	dy	1 à 366
day	dd	1 à 31
week	wk	1 à 51
weekday	dw	1 à 7
hour	hh	0 à 23
minute	mm	0 à 59
second	ss	0 à 59
millisecond	ms	0 à 999

Exemplos

```
SELECT pubdate, DATEDIFF(MONTH,pubdate,GETDATE() FROM titles

SELECT title_id, pubdate, DATEADD(DAY,3,pubdate) FROM titles
```

Conversão de Valores

CONVERT permite a conversão de expressões de um tipo de dado para outro. Ele também permite a formatação de datas com grande variedade de estilos.

Sintaxe `CONVERT(datatype(length),expression,style)`

Ano 2 digitos	Ano 4 digitos	Padrão	Saída
-	0 ou 100	Default	mon dd yyyy hh:mm AM/PM
1	101	USA	mm/dd/yy
2	102	ANSI	yy.mm.dd
3	103	British/French	dd/mm/yy
4	104	German	dd.mm.yy
5	105	Italian	dd-mm-yy

6	106	-	dd mon yy
7	107	-	mon dd, yy
8	108	-	hh:mm:ss
-	9 ou 109	Default + milliseconds	mon dd,yyyy hh:mm:ss AM/PM
10	110	USA	mm-dd-yy
11	111	Japan	yy/mm/dd
12	112	ISO	yymmdd
-	13 ou 113	Europe default + milliseconds	dd mon yyyy hh:mm:ss:ms (24)
14	114	-	hh:mm:ss:ms (24)

Exemplos

```
SELECT CONVERT(char(30),GETDATE(),102)

SELECT 'Title Code' = pub_id + UPPER(SUBSTRING(type,1,3)) +
SUBSTRING(CONVERT(CHAR(4),DATEPART(YY,pubdate)),3,2) FROM titles
```

Filtrando resultados

A clausula WHERE no comando SELECT especifica quais linhas deverão ser exibidas.

Sintaxe `SELECT select_list FROM table_list WHERE search_condition`

Comparação de valores	=, >, <, >=, <=, <>, !=, !< e !>
Intervalos	BETWEEN e NOT BETWEEN
Listas	IN e NOT IN
Procura de strings	LIKE e NOT LIKE
Valores NULL	IS NULL e IS NOT NULL
Combinações	AND e OR
Negação	NOT

Quando especificar um critério de pesquisa, é uma boa pratica não utilizar condições negativas, como NOT IN e NOT BETWEEN, porque estas condições não são reconhecidas pelo Query Optimizer.

Exemplos

```
SELECT au_lname, city FROM authors WHERE state = 'CA'

SELECT pubdate, title FROM titles WHERE pubdate BETWEEN '1/1/91' AND
'12/31/91'

SELECT title, type FROM titles WHERE type in ('mod_cook','trad_cook')
SELECT title, type FROM titles WHERE type = 'mod_cood' OR type =
'trad_cook'

SELECT stor_name FROM stores WHERE stor_name LIKE '%BOOK%'
Wildcard  Descrição
    %     Qualquer agrupamento
```

```
_       Caracter único
[]      Caracter único com o intervalo
[^]     Caracter único sem o intervalo
```

Expressão	**Retorno**
LIKE 'BR%'	Toda string que começa com "BR"
LIKE 'Br%	Toda string que começa com "Br"
LIKE '%een'	Toda string que termina com "een"
LIKE '%en%'	Toda string que tem as letras "en"
LIKE '_en'	Toda string de 3 letras que termina com "en"
LIKE '[CK]%'	Toda string que começa com "C" ou "K"
LIKE '[S-V]ing'	Toda string de 4 letras que termina com "ing" e começa com o intervalo de "S" até "V"
LIKE 'M[^c]%	Toda string que começa com "M" e não tem o "c" na segunda posição

```
SELECT title FROM titles WHERE price IS NULL

SELECT title_id, title, pub_id, price, pubdate from titles
WHERE (title LIKE 'T%' OR pub_id = '0877') AND (price > $16.00)

SELECT DISTINCT city, state FROM authors

SELECT pub_id, type, price, title FROM titles ORDER BY type, price DESC
```

Funções de Agregação

Função	Parametros	Descrição
AVG	ALL|DISTINCT expressão	Média
COUNT	ALL|DISTINCT expressão	Quantidade de valores em uma expressão
COUNT	(*)	Quantidade de linhas
MAX	(expressão)	Maior valor
MIN	(expressão)	Menor valor
SUM	ALL|DISTINCT expressão	Somatória

```
SELECT COUNT(*) FROM titles

SELECT COUNT(DISTINCT title_id) FROM titles
```

GROUP BY e HAVING

GROUP BY → Organiza os dados em grupos
- Pode agrupar uma coluna ou uma expressão
- Utilizado tipicamente em conjunto com funções de agregação
- Produz um valor único para cada grupo

HAVING → Restringe os grupos baseados em uma condição
- Pode ser utilizado em uma coluna ou expressão
- Permite funções de agregação

- Similar a clausula WHERE

Uma função de agregação produz um único resultado para uma tabela. A clausula GROUP BY organiza um sumário de informações em grupos. A clausula HAVING se relaciona com a clausula GROUP BY da mesma forma que a clausula WHERE se relaciona com o comando SELECT.

Algumas caracteristicas destas clausulas:
- A clausula WHERE exclui linhas que não satisfazem sua condição de procura.
- A clausula GROUP BY colecina linhas que satisfazem a condição da clausula WHERE e as agrupa de acordo com os valores da especificados. Com a omissão da clausula GROUP BY, ocorre a criação de um único grupo individual.
- A clausula HAVING exclui grupos que não satisfaçam sua condição. Por exemplo: Quando uma query possui a clausula GROUP BY, a clausula HAVING exclui grupos do resultado. Um comando SELECT contendo a clausula HAVING sem a clausula GROUP BY irá processar os resultados como um grupo único.

Exemplos

```
SELECT title_id, copies_sold = sum(qty) FROM sales GROUP BY title_id

SELECT title_id, copies_sold = sum(qty) FROM sales GROUP BY title_id
HAVING sum(qty) > 30

SELECT title_id, copies_sold = sum(qty) FROM sales WHERE ord_date
BETWEEN '1/1/1994' AND '12/31/1994'
GROUP BY ALL title_id
```

Implementando Joins

ANSI

```
SELECT table_name.column_name, table_name.column_name,
table_name.column_name FROM table_name [join_type] table_name on
searh_conditions WHERE search condition
```

SQL Server

```
SELECT table_name.column_name, table_name.column_name,
table_name.column_name FROM table_name, table_name
WHERE table_name.column_name join_operator table_name.column_name
```

INNER JOIN → inclui somente as linhas que satisfazem a condição de join.
CROSS JOIN → inclui toda cominação de todas as linhas entre as tabelas.
OUTER JOIN → inclui tdas as linhas que satisfazem a condição de join e as linhas remanescentes de uma das tabelas.

- LEFT OUTER JOIN (*=) → inclui todas as linhas da primeira tabela (a esquerda da expressão)
- RIGHT OUTER JOIN (=*) → inclui todas as linhas da segunda tabela (a direita da expresão)
- FULL OUTER JOIN → inclui todas as linhas das tabelas declaradas

Exemplo

```
USE pubs
-- ANSI SQL Syntax
SELECT title, stor_id, ord_num, qty, ord_date
FROM titles LEFT OUTER JOIN sales
    ON titles.tiles_id = sales.title_id

-- SQL Server Syntax
SELECT title, stor_id, ord_num, qty, ord_date
FROM titles, sales
WHERE titles.titles_id *= sales.title_id
```

Joins com mais de 2 tabelas

ANSI Join Syntax

```
SELECT coumn_name, column_name, column_name
FROM table_name join_type JOIN table_name ON search_condition
                Join_type JOIN table_name ON search_condition
WHERE search_condition
```

SQL Server Join Syntax

```
SELECT coumn_name, column_name, column_name
FROM table_name, table_name, table_name
WHERE table_name.column_name join_operator table_name.column_name AND
      table_name.column_name join_operator table_name.column_name
```

Executando Sub-Queries

- Pode ser combinada com os commandos SELECT, INSERT, UPDATE e DELETE
- A subquery é executada primeiro
- A subquery deve ser escrita entre parentesis
- Pode retornar 1 valor (1 coluna → 1 valor) ou N valores (1 coluna → n valores)
- Deve especificar uma única coluna (não permitido para tipos text e image)
- Deve ter a clausula DISTINCT implicita em seu resultado

Exemplo

```
SELECT
```

```
   Title = title_id,
   Quantity = qty,
   Total = (SELECT SUM(qty) FROM sales)
   Percentage_of_Total = (CONVERT(float,qty)/(SELECT SUM(qty) FROM
sales))*100
FROM sales

SELECT * FROM titles WHERE title_id IN (SELECT title_id FROM sales)
```

SELECT INTO

Utilizado para a criação de uma nova tabela baseada no resultado na query.

Exemplo

```
SELECT title = SUBSTRING(title,1,40), monthly = ytd_sales/12
INTO #temporarytable
FROM  titles
```

Inserindo Linhas

O comando INSERT é utilizado para inserir linhas em uma tabela.
- Use um comando INSERT para inserir cada linha.
- A ordem e o datatype dos itens a serem inseridos devem corresponder a ordem e o datatype correspondente na tabela.
- Um valor NULL explicito sobrescreve o default.

Sintaxe

```
INSERT INTO table_name|view_name (column list)
VALUES (value_list|SELECT_statement)
```

Exemplo

```
INSERT authors VALUES ('123-45-6789', ' Chen','Sue','900 555-1212','214
Main St. ', 'Kent','WA','98000',0)
```

- Valores Parciais

```
INSERT publishers (pub_id,pub_name) VALUES ('9975','Unbound Press')
```

- A partir de um comando SELECT

```
INSERT stores
     SELECT substring(au_id,8,4), au_lname, address, city, state,zip
     FROM authors
```

- A partir de uma Stored Procedure

```
INSERT into employee_info
```

```
     EXECUTE fetch_employee_data

INSERT into sales_info
     EXECUTE fetch_sales_figures
```

Alterando linhas de dados

O comando UPDATE altera as informações existentes nas linhas de uma tabela.
- SET → Especifica a coluna e o valor
- WHERE → Especifica as linhas a serem alteradas

Sintaxe
```
UPDATE table_name SET  column_list|variable_list = expression
WHERE condition
```

Exemplo
```
UPDATE discounts SET discount =  discount +.10 WHERE lowqty >= 100
UPDATE titles SET ytd_sales = 0
UPDATE titles SET ytd_sales = (SELECT SUM(qty) FROM sales WHERE
sales.title_id = title.title_id AND ord_data BETWEEN '01/01/95' AND
'12/31/95)
```

Excluindo Linhas

- O comando DELETE exclui uma ou mais linhas.
- A clausula WHERE especifica quais linhas deverão ser removidas
- TRUNCATE TABLE remove todas as linhas de uma tabela

Sintaxe
```
DELETE FROM table_name WHERE search_condition
```

Exemplo
```
DELETE FROM sales WHERE DATEDIFF(year,ord_date,GETDATE()) >=3
```