Linguagem SQL

Aula 03 – Tipos de Dados

Sand Onofre

Sand.Onofre@FaculdadeImpacta.com.br

Sumário

- Tipos de Dados
  - Numérico, string, data
- Data Query Language – DQL
  - Convenção de nomes de objetos em 4 partes
  - SCHEMAS
  - Cláusula WHERE - Filtrando Consultas
  - Operadores Lógicos (AND, OR, NOT)
  - Cláusula Simples e com Predicado
  - Predicados: BETWEEN, IN, LIKE
  - Trabalhando com Valores NULL
  - Cláusula ORDER BY
- Exercícios

Tipos de Dados

Bancos de dados associam tipos de dados a colunas, expressões, variáveis e parâmetros.

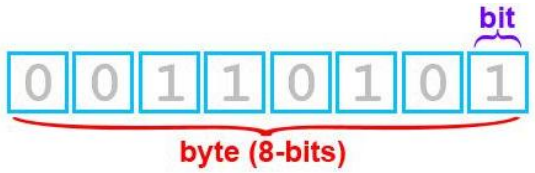
Tipos de dados determinam quais os tipos de valores serão permitidos no armazenamento.

Todos os dados são armazenados nos bancos de dados em formato de Bytes. Essa é a forma como os computadores trabalham, ou sejam, quando irão armazenar a letra A, na realidade, armazenam o código binário “01000001” que a representa.

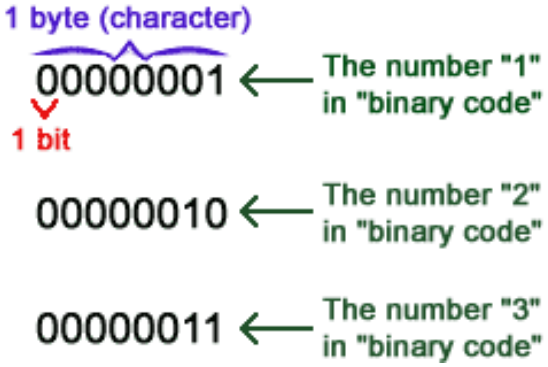
Quando esse dado precisa ser mostrado, o banco de dados traduz o formato binário gravado, na letra A.

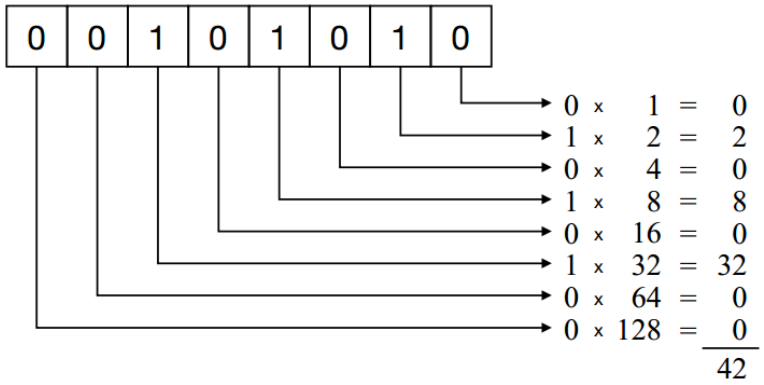
Tipos de Dados

- Armazenamento de Dados em Bytes



$= 2^1 * 2^1 * 2^1 * 2^1 * 2^1 * 2^1 * 2^1 * 2^1 = 2^8 = 256$





Tipos de Dados

Lower Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00		32	20		64	40	@	96	60	`
1	01	☺	33	21	!	65	41	A	97	61	a
2	02	☻	34	22	"	66	42	B	98	62	b
3	03	♥	35	23	#	67	43	C	99	63	c
4	04	♦	36	24	$	68	44	D	100	64	d
5	05	♣	37	25	%	69	45	E	101	65	e
6	06	♠	38	26	&	70	46	F	102	66	f
7	07	•	39	27	'	71	47	G	103	67	g
8	08	◘	40	28	(	72	48	H	104	68	h
9	09	○	41	29	)	73	49	I	105	69	i
10	0A	◙	42	2A	*	74	4A	J	106	6A	j
11	0B	♂	43	2B	+	75	4B	K	107	6B	k
12	0C	♀	44	2C	,	76	4C	L	108	6C	l
13	0D	♪	45	2D	-	77	4D	K	109	6D	m
14	0E	♫	46	2E	.	78	4E	N	110	6E	n
15	0F	☼	47	2F	/	79	4F	O	111	6F	o
16	10	►	48	30	0	80	50	P	112	70	p
17	11	◄	49	31	1	81	51	Q	113	71	q
18	12	↕	50	32	2	82	52	R	114	72	r
19	13	‼	51	33	3	83	53	S	115	73	s
20	14	¶	52	34	4	84	54	T	116	74	t
21	15	§	53	35	5	85	55	U	117	75	u
22	16	▬	54	36	6	86	56	V	118	76	v
23	17	↨	55	37	7	87	57	W	119	77	w
24	18	↑	56	38	8	88	58	X	120	78	x
25	19	↓	57	39	9	89	59	Y	121	79	y
26	1A	→	58	3A	:	90	5A	Z	122	7A	z
27	1B	←	59	3B	;	91	5B	[	123	7B	{
28	1C	∟	60	3C	<	92	5C	\	124	7C	|
29	1D	↔	61	3D	=	93	5D	]	125	7D	}
30	1E	▲	62	3E	>	94	5E	^	126	7E	~
31	1F	▼	63	3F	?	95	5F	_	127	7F	⌂

Upper Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

- Exemplo de Armazenamento em Bytes - Tabela ASCII

Tipos de Dados

- Exemplo de Armazenamento do Alfabeto em sistema binário

Carácter	ASCII	Carácter	ASCII
A	0100 0001	W	0101 0111
B	0100 0010	X	0101 1000
C	0100 0011	Y	0101 1001
D	0100 0100	Z	0101 1010
E	0100 0101	0	0011 0000
F	0100 0110	1	0011 0001
G	0100 0111	2	0011 0010
H	0100 1000	3	0011 0011
I	0100 1001	4	0011 0100
J	0100 1010	5	0011 0101
K	0100 1011	6	0011 0110
L	0100 1100	7	0011 0111
M	0100 1101	8	0011 1000
N	0100 1110	9	0011 1001
O	0100 1111	+	0010 1011
P	0101 0000	-	0010 1101
Q	0101 0001	*	0010 1010
R	0101 0010	:	0011 1010
S	0101 0011	=	0011 1101
T	0101 0100	<	0011 1100
U	0101 0101	;	0011 1011
V	0101 0110		

Tipos de Dados

Tipos de dados determinam quais os tipos de valores serão permitidos no armazenamento e os principais tipos são agrupados em categorias conforme mostrado abaixo:

Categorias dos Tipos de Dados	
Numéricos Exatos	Caractere Unicode
Numéricos Aproximados	Binários
Data e Hora	Outros Tipos
Strings de Caractere	

Tipos de Dados

- Numéricos Exatos

Data type	Range	Storage (bytes)
tinyint	0 to 255	1
smallint	-32,768 to 32,767	2
int	2^31 (-2,147,483,648) to 2^31-1 (2,147,483,647)	4
Bigint	-2^63 - 2^63-1 (+/- 9 quintillion)	8
bit	1, 0 or NULL	1
decimal/numeric	- 10^38 +1 through 10^38 – 1 when maximum precision is used	5-17
money	-922,337,203,685,477.5808 to 922,337,203,685,477.5807	8
smallmoney	- 214,748.3648 to 214,748.3647	4

Tipos de Dados

- Dados Caractere Non-Unicode

Data Type	Range	Storage
CHAR(n)	1-8000 characters	n bytes
VARCHAR(n)	1-8000 characters	n+2 bytes
VARCHAR(MAX)	1-2^31-1 characters	Actual length + 2

- Dados Caractere Unicode

Data Type	Range	Storage
NCHAR(n)	1-4000 characters	2*n bytes
NVARCHAR(n)	1-4000 characters	(2*n) +2 bytes
NVARCHAR(MAX)	1-2^31-1 characters	(Actual length) * 2 + 2

Tipos de Dados

- Data e Hora

Data Type	Storage (bytes)	Date Range	Accuracy	Recommended Entry Format
DATETIME	8	January 1, 1753 to December 31, 9999	3-1/3 milliseconds	'YYMMDD hh:mm:ss:nnn'
SMALLDATETIME	4	January 1, 1900 to June 6, 2079	1 minute	'YYMMDD hh:mm:ss'
DATETIME2	6 to 8	January 1, 0001 to December 31, 9999	100 nanoseconds	'YYMMDD hh:mm:ss.nnnnnnn'
DATE	3	January 1, 0001 to December 31, 9999	1 day	'YYYY-MM-DD'
TIME	3 to 5		100 nanoseconds	'hh:mm:ss:nnnnnnn'
DATETIMEOFFSET	8 to 10	January 1, 0001 to December 31, 9999	100 nanoseconds	'YY-MM-DD hh:mm:ss:nnnnnnn [+|-]hh:mm'

Tipos de Dados

- Data e Hora

Data Type	Language-Neutral Formats	Examples
DATETIME	'YYYYMMDD hh:mm:ss.nnn' 'YYYY-MM-DDThh:mm:ss.nnn' 'YYYYMMDD'	'20120212 12:30:15.123' '2012-02-12T12:30:15.123' '20120212'
SMALLDATETIME	'YYYYMMDD hh:mm' 'YYYY-MM-DDThh:mm' 'YYYYMMDD'	'20120212 12:30' '2012-02-12T12:30' '20120212'
DATETIME2	'YYYY-MM-DD' 'YYYYMMDD hh:mm:ss.nnnnnnn' 'YYYY-MM-DD hh:mm:ss.nnnnnnn' 'YYYY-MM-DDThh:mm:ss.nnnnnnn' 'YYYYMMDD' 'YYYY-MM-DD'	'20120212 12:30:15.1234567' '2012-02-12 12:30:15.1234567' '2012-02-12T12:30:15.1234567' '20120212' '2012-02-12'
DATE	'YYYYMMDD' 'YYYY-MM-DD'	'20120212' '2012-02-12'
TIME	'hh:mm:ss.nnnnnnn'	'12:30:15.1234567'
DATETIMEOFFSET	'YYYYMMDD hh:mm:ss.nnnnnnn [+|-]hh:mm' 'YYYY-MM-DD hh:mm:ss.nnnnnnn [+|-]hh:mm' 'YYYYMMDD' 'YYYY-MM-DD'	'20120212 12:30:15.1234567 +02:00' '2012-02-12 12:30:15.1234567 +02:00' '20120212' '2012-02-12'

Tipos de Dados

- Numéricos Aproximados

Data Type	Range	Storage (bytes)
float(n)	- 1.79E+308 to -2.23E-308, 0 and 2.23E-308 to 1.79E+308	Depends on value of n, 4 or 8 bytes
real	- 3.40E + 38 to -1.18E - 38, 0 and 1.18E - 38 to 3.40E + 38	4

- Binários

Data Type	Range	Storage (bytes)
binary(n)	1-8000 bytes	n bytes
varbinary(n)	1-8000 bytes	n bytes + 2
varbinary(MAX)	1-2.1 billion (approx) bytes	actual length + 2

Tipos de Dados

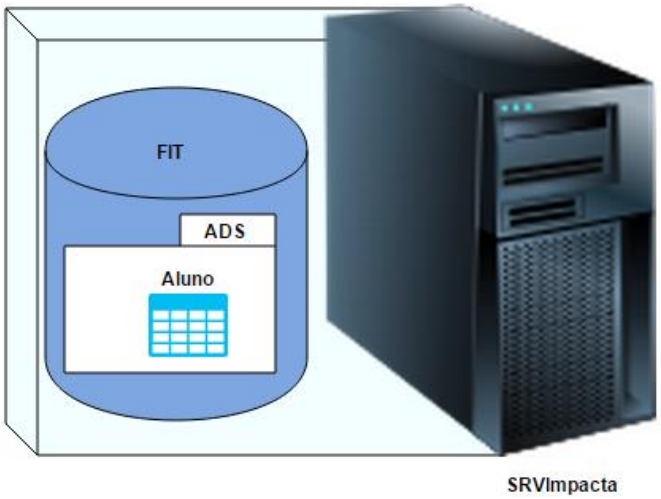
- Outros Tipos

Data Type	Range	Storage (bytes)	Remarks
rowversion	Auto-generated	8	Successor type to timestamp
uniqueidentifier	Auto-generated	16	Globally unique identifier (GUID)
xml	0-2 GB	0-2 GB	Stores XML in native hierarchical structure
cursor	N/A	N/A	Not a storage data type
hierarchyid	N/A	Depends on content	Represents position in a hierarchy
sql_variant	0-8000 bytes	Depends on content	Can store data of various data types
table	N/A	N/A	Not a storage data type, used for query and programmatic operations

DQL – Four Part Naming

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

SELECT * FROM Server . Database . Schema . Object



SELECT * FROM SRVImpacta . FIT . ADS . Aluno

DQL – Schema

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

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

```
SELECT * FROM ALUNO
```

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

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

DQL – Cláusula WHERE

A cláusula WHERE faz o filtro horizontal em uma consulta, ou seja, permite uma redução do número de linhas que retornarão na consulta.

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

DQL – Cláusula WHERE

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

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

Operadores de Comparação Escalar

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

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

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

DQL – Cláusula WHERE

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

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

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

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

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

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

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

DQL – Cláusula WHERE

- Cláusula WHERE Simples

```
SELECT BusinessEntityID AS 'Employee Identification Number', HireDate,
       VacationHours, SickLeaveHours
FROM   HumanResources.Employee
WHERE BusinessEntityID <= 1000
```

```
SELECT FirstName, LastName, Phone
FROM   Person.Person
WHERE FirstName = 'John';
```

DQL – Cláusula WHERE

- Cláusula WHERE usando Predicado

Nem sempre usamos operadores de comparação. Em algumas situações podemos usar outros operadores que são chamados de predicados, simplificando a escrita do script.

Alguns exemplos de Predicado são: IN, BETWEEN, ANY, SOME, IS, ALL, OR, AND, NOT, EXISTS ...

```
SELECT FirstName, LastName, Phone
FROM Person.Person
WHERE EmailAddress IS NULL;
```

DQL – Cláusula WHERE

- **BETWEEN** – restringe dados através de uma faixa de valores possíveis.

```
SELECT OrderDate, AccountNumber, SubTotal, TaxAmt
FROM    Sales.SalesOrderHeader
WHERE OrderDate BETWEEN ’20110801’ AND ’ 20110831’
```

- **BETWEEN** – A mesma lógia do uso de >= AND <=

```
SELECT OrderDate, AccountNumber, SubTotal, TaxAmt
FROM    Sales.SalesOrderHeader
WHERE OrderDate >= ’ 20110801’
          AND OrderDate <= ’ 20110831’
```

OrderDate	AccountNumber	SubTotal	TaxAmt
2001-08-01 00:00:00.000	10-4020-000018	39677.4848	3174.1988
2001-08-01 00:00:00.000	10-4020-000353	24299.928	1943.9942
2001-08-01 00:00:00.000	10-4020-000206	10295.8366	823.6669
2001-08-01 00:00:00.000	10-4020-000318	1133.2967	90.6637
2001-08-01 00:00:00.000	10-4020-000210	1086.6152	86.9292
2001-08-01 00:00:00.000	10-4020-000164	21923.9352	1753.9148
2001-08-01 00:00:00.000	10-4020-000697	24624.706	1969.9765
2001-08-01 00:00:00.000	10-4020-000191	12286.7218	982.9377

DQL – Cláusula WHERE

- **IN** – fornece uma lista de possibilidades de valores que poderiam atender a consulta.

```
SELECT SalesOrderID, OrderQty, ProductID, UnitPrice
FROM Sales.SalesOrderDetail
WHERE ProductID IN (750, 753, 765, 770)
```

- **IN** – Usa a mesma lógica de múltiplas comparações com o predicado OR entre elas.

```
SELECT SalesOrderID, OrderQty, ProductID, UnitPrice
FROM Sales.SalesOrderDetail
WHERE ProductID = 750 OR ProductID = 753
      OR ProductID = 765 OR ProductID = 770
```

SalesOrderID	OrderQty	ProductID	UnitPrice
43662	5	770	419.4589
43662	3	765	419.4589
43662	1	753	2146.962
43666	1	753	2146.962
43668	2	753	2146.962
43668	6	765	419.4589
43668	2	770	419.4589
43671	1	753	2146.962
43673	2	770	419.4589

DQL – Cláusula WHERE

- **LIKE** – Permite consultas mais refinadas em colunas do tipo string (CHAR, VARCHAR, …).

```
WHERE LastName = 'Johnson'
```

```
WHERE LastName LIKE 'Johns%n'
```

FirstName	LastName	MiddleName
Abigail	Johnson	NULL
Alexander	Johnson	M
Alexandra	Johnson	J
Alexis	Johnson	J
Alyssa	Johnson	K
Andrew	Johnson	F
Anna	Johnson	NULL

FirstName	LastName	MiddleName
Meredith	Johnsen	NULL
Rebekah	Johnsen	J
Ross	Johnsen	NULL
Willie	Johnsen	NULL
Abigail	Johnson	NULL
Alexander	Johnson	M
Alexandra	Johnson	J

DQL – Cláusula WHERE

- **LIKE** – Este predicado é usado para verificar padrões dentro de campos strings e utiliza símbolos, chamados de coringas, para permitir a busca desses padrões.

- Símbolos (coringas)
  - **%** (Percent) representa qualquer string e qualquer quantidade de strings
  - **_** (Underscore) representa qualquer string, mas apenas uma string
  - [<List of characters>] representa possíveis caracteres que atendam a string procurada
  - [<Character> **-** <character>] representa a faixa de caracteres, em ordem alfabética, para a string procurada
  - [**^**<Character list or range>] representa o caractere que não queremos na pesquisa

```
SELECT categoryid, categoryname, description
FROM Production.Categories
WHERE description LIKE 'Sweet%'
```

DQL – Utilização do NULL

~~NULL = 0 (zero)~~

~~NULL = ‘’ (branco ou vazio)~~

~~NULL = ‘NULL’ (string NULL)~~

~~NULL = NULL~~

DQL – Cláusula WHERE

- **NULL** – É ausência de valor ou valor desconhecido. Nenhuma das sentenças anteriores são verdadeiras pois o banco de dados não pode comparar um valor desconhecido com outro valor que ele também não conhece.
- Para trabalhar com valores NULL, temos que utilizar os predicados IS NULL e IS NOT NULL.

```
SELECT custid, city, region, country
FROM Sales.Customers
WHERE region IS NOT NULL;
```

- Predicados retornam UNKNOWN quando comparados com valores desconhecidos (valores faltando), ou seja, não são retornados na consulta.

DQL – Cláusula ORDER BY

Conforme mencionado anteriormente, por padrão, não há garantia de ordenação no retorno dos dados de uma consulta.

Para garantir que o retorno da consulta tenha uma ordenação, utilizamos a cláusula ORDER BY.

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

As cláusulas ASC e DESC podem ser usadas após cada campo do commando ORDER BY. A ordenação ASCendente é a padrão quando não mencionamos explicitamente.

DQL – Cláusula ORDER BY

- ORDER BY com nome de colunas:

```
SELECT orderid, custid, orderdate
FROM Sales.Orders
ORDER BY orderdate;
```

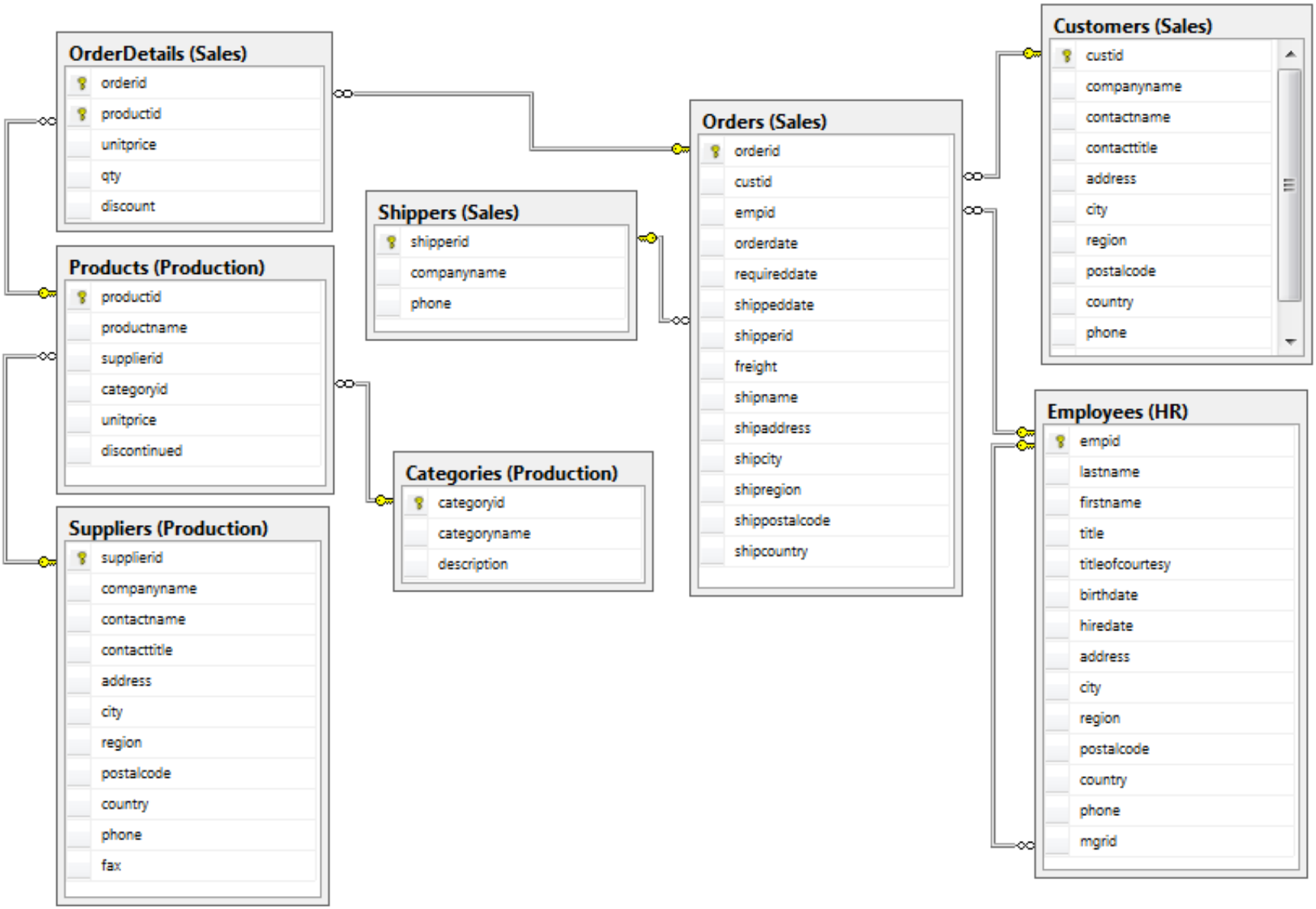
- ORDER BY com apelido:

```
SELECT orderid, custid, YEAR(orderdate) AS
orderyear
FROM Sales.Orders
ORDER BY orderyear;
```

- ORDER BY with descending order:

```
SELECT orderid, custid, orderdate
FROM Sales.Orders
ORDER BY orderdate DESC;
```

TSQL


OrderDetails (Sales)
orderid
productid
unitprice
qty
discount
Products (Production)
productid
productname
supplierid
categoryid
unitprice
discontinued
Suppliers (Production)
supplierid
companyname
contactname
contacttitle
address
city
region
postalcode
country
phone
fax
Shippers (Sales)
shipperid
companyname
phone
Categories (Production)
categoryid
categoryname
description
Orders (Sales)
orderid
custid
empid
orderdate
requireddate
shippeddate
shipperid
freight
shipname
shipaddress
shipcity
shipregion
shippostalcode
shipcountry
Customers (Sales)
custid
companyname
contactname
contacttitle
address
city
region
postalcode
country
phone
Employees (HR)
empid
lastname
firstname
title
titleofcourtesy
birthdate
hiredate
address
city
region
postalcode
country
phone
mgrid

Exercícios

1. Construa duas consultas que retornem o **Número do Produto** (productid), **Nome** (productname) e **Preço** (unitprice) onde os preços estejam entre **19** e **22**. Construir a 1ª. consulta, utilizando o operador ***AND***.

Na 2ª. Consulta, que deverá retornar exatamente as mesmas colunas e linhas, utilizar o predicado ***BETWEEN***. (tabela ***Production.Products***)

2. Construa duas consultas que retornem o **Número do Produto** (productid), **Nome** (productname) e **Preço** (unitprice) onde os preços sejam quaisquer um a seguir: **18** ou **10** ou **21,35**. A 1ª. consulta, utilizar o operador ***OR***. Na 2ª. Consulta, que deverá retornar exatamente as mesmas colunas e linhas, utilizar o predicado ***IN***.

3. Retorne o **Número do Empregado** (empid), **Título** (titleofcourtesy), a **Data de Nascimento** (birthdate) e concatenar as colunas nome (firstname) e sobrenome (lastname) em apenas uma coluna apelidada de **Nome Completo**, apenas para os empregados que nasceram em qualquer dia e mês desde **1947** até **1965**. (tabela ***HR.Employees***)

4. Retorne a **Cidade** (city), **Região** (region) e **País** (country) dos empregados da cidade de **Seattle** ou do país **UK**.

5. Retorne o **Número do Empregado** (empid), **Nome** (firstname) e **Data de Contratação** (hiredate) dos empregados contratados em qualquer dia e mês de 2002 **ou** de 2004.

Exercícios

6. Retorne todas as colunas dos últimos 20 registros de **Clientes** (tabela ***Sales.Customers***), ou seja, do maior **Número do Cliente** (**custid**) para o menor.

7. Primeiramente faça uma query para visualizar os **países distintos** existentes na tabela anterior. Agora construa uma consulta que retorne o **Número do Cliente** (custid), o **Nome do Contato** (contactname), **Cidade** (city) e **País** (country), apenas para os clientes da **América do Sul**, lembrando que apenas pelo campo **País** conseguiremos essa filtragem. Ordene pelo **País**.

8. Retorne o **Número do Cliente** (custid), o **Nome do Contato** (contactname) e o **Fax** onde esta última coluna **NÃO** seja **NULL**. Ordene do maior **Número do Cliente** ao menor.

9. Retorne o **Número do Cliente** (custid), **Cidade** (city) e o **País** (country) onde a **Região** (region) seja **NULL**. Ordene de forma **ascendente** para o **País** e **descendente** para a **Cidade**.

Obrigado !

Sand Onofre
Sand.Onofre@FaculdadeImpacta.com.br