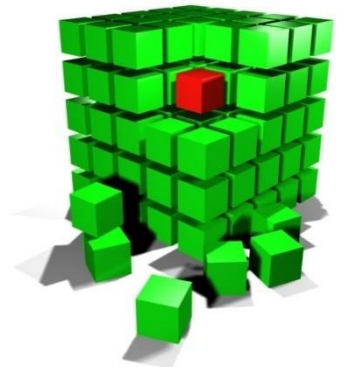
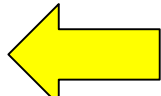


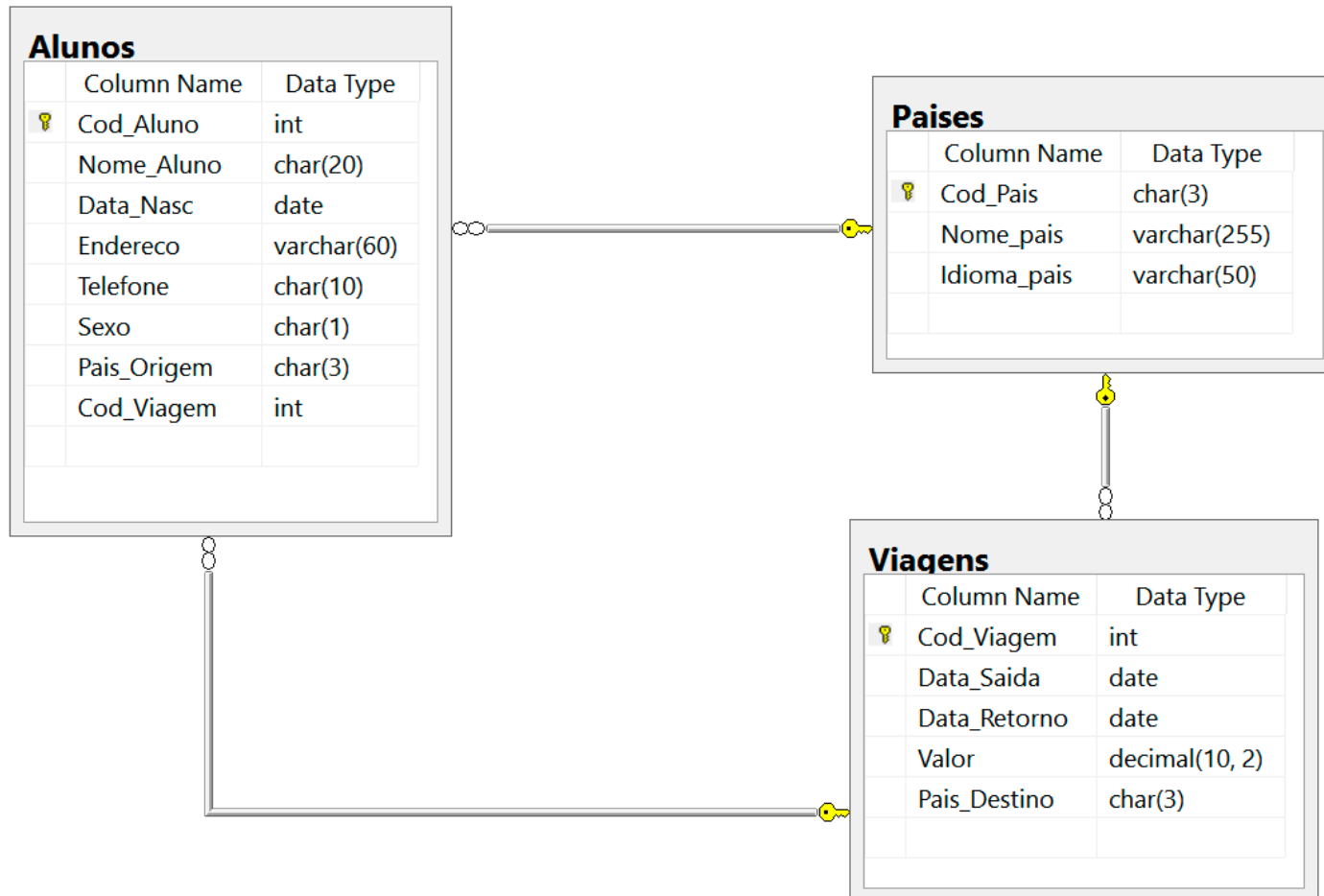
- **Aula 013 – Linguagem SQL**
 - Consultas relacionais de união, intersecção e diferença.
 - Subconsultas.



- **Banco de dados INTERCAMBIO**
 - Para esta aula, utilizaremos um banco de dados de exemplo, denominado **INTERCAMBIO**. Este banco contém dados de alunos, de países e das viagens que eles realizam para esses países.
 - **01_INTERCAMBIO_Cria_Banco.sql** 



■ Banco de dados INTERCAMBIO

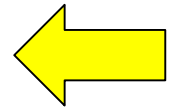


■ Banco de dados INTERCAMBIO

-- Exibe o nome das tabelas que existem no
-- banco de dados em uso

```
SELECT name  
FROM sys.tables  
GO
```

```
name  
-----  
Países  
Viagens  
Alunos  
AlunosCOPIA
```



(4 row(s) affected)



■ Operadores de junção de SQL

- A operação de junção relacional combina os valores de duas tabelas e retorna as linhas com uma das seguintes condições:
 - Tenham valores comuns em colunas comuns: **junção natural**;
 - Atendam a determinada condição de junção: **igualdade** ou **desigualdade**;
 - Tenham valores comuns em colunas comuns ou não tenham valores correspondentes: **junção externa**.
- **Junção interna** – apenas as linhas que atendam a determinados critérios são selecionadas.



■ Operadores de junção de SQL

Tipo de Junção	Descrição
CROSS JOIN	Retorna o produto cartesiano de duas tabelas.
JOIN ON	Retorna apenas as linhas que atendam à condição de junção descrita na cláusula ON.
NATURAL JOIN	Retorna apenas as linhas com valores correspondentes nas cláusulas correspondentes
JOIN USING	Retorna apenas as linhas com valores correspondentes nas colunas indicadas pela cláusula USING.
LEFT OUTER JOIN	Retorna linhas com valores correspondentes e inclui todas as colunas da tabela à esquerda de LEFT.
RIGHT OUTER JOIN	Retorna linhas com valores correspondentes e inclui todas as colunas da tabela à direita de RIGHT.
FULL OUTER JOIN	Retorna linhas com valores correspondentes e inclui todas as colunas de ambas as tabelas.



■ União, intersecção e diferença

- Na linguagem SQL, os comandos **UNION**, **INTERSECT** e **MINUS** podem ser utilizados para implementar os operadores relacionais de união, intersecção e diferença.
- Esses comandos permitem a combinação de duas ou mais tabelas, para formar novas tabelas.



■ União, intersecção e diferença

- Os comandos **UNION**, **INTERSECT** e **MINUS** funcionam adequadamente somente se as tabelas utilizadas na consulta forem **compatíveis para a união**.
- Isso significa que os nomes dos atributos devem ser os mesmos e os seus tipos de dados devem ser semelhantes.



- **União, intersecção e diferença**
 - O **padrão SQL** define as operações que todos os SGBDs devem executar sobre os dados, mas deixa os detalhes de implementação para os fornecedores de sistemas.
 - Portanto, alguns recursos e comandos podem não funcionar em todas as implementações de SGBDs.



- **União, intersecção e diferença**
 - **UNION**, **INTERSECT** e **MINUS** são nomes dos comandos de SQL implementados no Oracle.
 - O padrão SQL utiliza a palavra-chave **EXCEPT** para se referir ao operador relacional de diferença (**MINUS**).
 - Entretanto, outros fornecedores podem não implementar um determinado comando SQL ou utilizar um nome diferente para ele.



■ União, intersecção e diferença

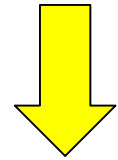
- **UNION** – combina duas ou mais consultas, sem exibir linhas duplicadas.
- **UNION ALL** – combina duas ou mais consultas, incluindo as linhas duplicadas.
- **INTERSECT** – combina linhas de duas consultas, retornando apenas aquelas que são comuns.
- **MINUS** – combina linhas de duas consultas e retorna apenas as que aparecem no primeiro conjunto, mas não no segundo.
- **EXCEPT** – funciona de maneira idêntica ao operador **MINUS**.



■ Banco de dados INTERCAMBIO

-- Exibe o nome das tabelas que existem no
-- banco de dados em uso

```
SELECT TABLE_NAME AS 'Nome da Tabela'  
FROM INFORMATION_SCHEMA.TABLES  
WHERE TABLE_TYPE = 'BASE TABLE' AND  
       TABLE_CATALOG = 'INTERCAMBIO'  
GO
```



Nome da Tabela

Países

Viagens

{ Alunos
AlunosCOPIA

(4 row(s) affected)



■ União de tabelas – UNION e UNION ALL

-- UNION ALL -> Exibe as linhas duplicadas

```
SELECT  Cod_Aluno  AS 'Código do Aluno',  
        Nome_Aluno AS 'Nome do Aluno',  
        Sexo
```

```
FROM Alunos
```

```
UNION ALL ←
```

```
SELECT  Cod_Aluno  AS 'Código do Aluno',  
        Nome_Aluno AS 'Nome do Aluno',  
        Sexo
```

```
FROM AlunosCOPIA
```

```
ORDER BY Cod_Aluno, Nome_Aluno
```

```
GO
```



■ União de tabelas – UNION e UNION ALL

Código do Aluno	Nome do Aluno	Sexo
-----------------	---------------	------

1	Maria Cristina	F
---	----------------	---

1	Maria Cristina	F
---	----------------	---

2	Ana Paula Lima	F
---	----------------	---

2	Ana Paula Lima	F
---	----------------	---

3	Carlos Renato	M
---	---------------	---

3	Felipe Neto	M
---	-------------	---

48	Gabriela Pereira	F
----	------------------	---

49	André César	M
----	-------------	---

50	Edson Lopes	M
----	-------------	---

(70 row(s) affected) ←



■ União de tabelas – UNION e UNION ALL

-- UNION -> Não exibe as linhas duplicadas

```
SELECT  Cod_Aluno  AS 'Código do Aluno',  
        Nome_Aluno AS 'Nome do Aluno',  
        Sexo  
FROM Alunos  
        UNION ←  
SELECT  Cod_Aluno  AS 'Código do Aluno',  
        Nome_Aluno AS 'Nome do Aluno',  
        Sexo  
FROM AlunosCOPIA  
ORDER BY Cod_Aluno, Nome_Aluno  
GO
```



■ União de tabelas – UNION e UNION ALL

Código do Aluno	Nome do Aluno	Sexo
-----------------	---------------	------

1	Maria Cristina	F
---	----------------	---

2	Ana Paula Lima	F
---	----------------	---

3	Carlos Renato	M
---	---------------	---

3	Felipe Neto	M
---	-------------	---

4	Carlos Eduardo	M
---	----------------	---

4	Hugo Silva	M
---	------------	---

48	Gabriela Pereira	F
----	------------------	---

49	André César	M
----	-------------	---

50	Edson Lopes	M
----	-------------	---

(68 row(s) affected) ←



■ Intersecção de tabelas – INTERSECTION

-- INTERSECT -> Retorna as linhas que existem nos dois conjuntos

```
SELECT  Cod_Aluno  AS 'Código do Aluno',  
        Nome_Aluno AS 'Nome do Aluno',  
        Sexo
```

```
FROM Alunos
```

INTERSECT ←

```
SELECT  Cod_Aluno  AS 'Código do Aluno',  
        Nome_Aluno AS 'Nome do Aluno',  
        Sexo
```

```
FROM AlunosCOPIA
```

```
GO
```



■ Intersecção de tabelas – INTERSECTION

Código do Aluno	Nome do Aluno	Sexo
-----	-----	-----
1	Maria Cristina	F
2	Ana Paula Lima	F

(2 row(s) affected) ←



■ Diferença de tabelas – EXCEPT ou MINUS

-- EXCEPT (MINUS) -> Retorna as linhas que existem nos dois conjuntos

```
SELECT  Cod_Aluno  AS 'Código do Aluno',  
        Nome_Aluno AS 'Nome do Aluno',  
        Sexo
```

```
FROM Alunos
```

```
EXCEPT ←
```

```
SELECT  Cod_Aluno  AS 'Código do Aluno',  
        Nome_Aluno AS 'Nome do Aluno',  
        Sexo
```

```
FROM AlunosCOPIA
```

```
GO
```



■ Diferença de tabelas – EXCEPT ou MINUS

Código do Aluno	Nome do Aluno	Sexo
3	Carlos Renato	M
4	Hugo Silva	M
5	Marcos Antônio	M
6	Gislaine Silva	F
7	Antônio Pereira	M
8	Jair Lopes	M
.....		
48	Gabriela Pereira	F
49	André César	M
50	Edson Lopes	M

(48 row(s) affected) ←



■ Alternativa de sintaxe

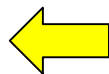
- Alguns SGBDs não fornecem suporte para os comandos **INTERSECT** ou **MINUS**. Neste caso, podemos utilizar subconsultas em conjunto com os operadores **IN** e **NOT IN**:

-- Selecciona somente os alunos cujo nome aparece nas duas
-- tabelas. Versão com INTERSECT.

```
SELECT Nome_Aluno AS 'Nome do Aluno'
```

```
FROM Alunos
```

```
INTERSECT
```



```
SELECT Nome_Aluno AS 'Nome do Aluno'
```

```
FROM AlunosCOPIA
```

```
GO
```



■ Alternativa de sintaxe

- Alguns SGBDs não fornecem suporte para os comandos **INTERSECT** ou **MINUS**. Neste caso, podemos utilizar subconsultas em conjunto com os operadores **IN** e **NOT IN**:

-- Selecciona somente os alunos cujo nome aparece nas duas
-- tabelas. Versão com uma subconsulta IN.

```
SELECT Nome_Aluno AS 'Nome do Aluno'
```

```
FROM Alunos
```

```
WHERE Nome_Aluno IN
```

```
(SELECT Nome_Aluno FROM AlunosCOPIA) ←
```

```
GO
```



■ Alternativa de sintaxe

Nome do Aluno

Maria Cristina

Ana Paula Lima

Carlos Pereira

Cristiano Leite

Alberto Carlos

Leandro Leite

Mônica Silva

Maurício dos Santos

Gabriela Pereira

André César

Edson Lopes

(11 row(s) affected) 



■ Subconsultas e consultas correlacionadas

- Uma **subconsulta** é uma consulta (comando **SELECT**) no interior de outra consulta.
- A subconsulta normalmente é expressa entre parênteses.
- A primeira consulta no comando de SQL é conhecida como consulta externa.
- A consulta no interior do comando de SQL é conhecida como consulta interna. Esse tipo de consulta é executada primeiro.
- A saída de uma consulta interna é utilizada como entrada da consulta externa.
- O comando SQL inteiro, às vezes, é chamado de consulta integrada.



- **Subconsultas e consultas correlacionadas**
 - Uma subconsulta pode retornar um único valor, uma lista de valores ou uma tabela virtual (**VIEW**).
 - É importante observar que uma subconsulta também pode retornar um valor NULO. Nesses casos, o resultado da consulta externa pode provocar um erro ou retornar um conjunto vazio, dependendo de onde ela estiver sendo utilizada.



- **Exemplo de subconsultas SELECT**
 - Insere todas as linhas da tabela P na tabela PRODUCT. A subconsulta retorna todas as linhas da tabela P.

```
INSERT INTO PRODUCT  
SELECT * FROM P  
GO
```



■ Exemplo de subconsultas SELECT

- Atualiza o preço dos produtos para o seu preço médio, somente para os produtos fornecidos pelo fornecedor de código de área igual a 615.

UPDATE PRODUCT

SET P_PRICE = (SELECT AVG(P_PRICE) ←
FROM PRODUCT)

WHERE V_CODE IN (SELECT V_CODE ←
FROM VENDOR
WHERE V_AREACODE = '615')



- **Exemplo de subconsultas SELECT**
 - Exclui as linhas da tabela PRODUCT cujo fornecedor possui o código de área igual a 615.

```
DELETE FROM PRODUCT
WHERE V_CODE IN (
    SELECT V_CODE ←
    FROM VENDOR
    WHERE V_AREACODE = '615')
```

GO



- **Subconsultas de listas de atributos**
 - O comando **SELECT** utiliza a lista de atributos para indicar quais colunas devem ser projetadas no conjunto resultante.
 - Essas colunas podem ser atributos de tabelas de base, atributos computados ou o resultado de uma função agregada.
 - A lista de atributos também pode incluir uma expressão de subconsulta.
 - Devem retornar um único valor.



■ Subconsultas de listas de atributos

-- Exibe informações sobre os alunos e as viagens que eles realizaram

```
SELECT  Viagens.Cod_Viagem    AS 'Código da Viagem',
        Alunos.Nome_Aluno    AS 'Nome',
        Alunos.Telefone,
        Alunos.Sexo,
        → (SELECT Nome_pais FROM Países WHERE Cod_Pais =
Alunos.Pais_Origem)          AS 'Origem',
        → (SELECT Nome_pais FROM Países WHERE Cod_Pais =
Viagens.Pais_Destino)        AS 'Destino',
        Viagens.Data_Saida    AS 'Data de Saída',
        Viagens.Data_Retorno  AS 'Data de Retorno',
        Viagens.Valor         AS 'Preço da Viagem R$'

FROM    Alunos INNER JOIN Viagens
        ON Alunos.Cod_Viagem = Viagens.Cod_Viagem
```



■ Subconsultas de listas de atributos

Código da Viagem	Nome	Telefone	Sexo	Origem	Destino	Data de Saída	Data de Retorno	Preço da Viagem R\$
1	Maria Cristina	1155686899	F	Brasil	Virgens Britânicas, Ilhas	2010-10-12	2010-12-12	5350.00
2	Ana Paula Lima	1156923232	F	Brasil	Vietnã	2010-09-14	2010-12-22	8900.00
3	Carlos Renato	1156121010	M	Brasil	Ucrânia	2010-12-05	2011-12-05	18525.35
4	Hugo Silva	1154788901	M	Brasil	Turquia	2010-08-11	2011-05-21	12350.25
5	Marcos Antônio	1156010201	M	Brasil	Toquelaui	2010-06-25	2010-08-25	7520.00
6	Gislaine Silva	1234598910	F	Brasil	Estados Unidos da América	2010-03-16	2011-03-16	14255.35
7	Antônio Pereira	1234966852	M	Brasil	Timor-Leste	2010-12-01	2010-12-30	3250.00
8	Jair Lopes	1236691857	M	Brasil	Tailândia	2010-10-12	2010-12-30	9500.00
9	Miguel Firmino	1234581212	M	Brasil	Rússia	2010-06-21	2010-09-05	9250.00
10	Marta da Silva	1234568989	F	Brasil	Senegal	2010-09-10	2010-10-02	6525.00
11	Jorge Augusto	1158988810	M	Brasil	Qatar	2010-07-27	2011-11-15	12355.00
.....								
9	Leandro Leite	2156894510	M	Brasil	Rússia	2010-06-21	2010-09-05	9250.00
9	Mônica Silva	2189562121	F	Brasil	Rússia	2010-06-21	2010-09-05	9250.00
4	Marília dos Santos	2134588585	F	Brasil	Turquia	2010-08-11	2011-05-21	12350.25
1	Guilherme dos Santos	1155661010	M	Brasil	Virgens Britânicas, Ilhas	2010-10-12	2010-12-12	5350.00
2	Heidi Lima	1236645605	F	Brasil	Vietnã	2010-09-14	2010-12-22	8900.00
8	Amílcar Júnior	1155640290	M	Brasil	Tailândia	2010-10-12	2010-12-30	9500.00
11	Alexandro Duarte	1236661515	M	Brasil	Qatar	2010-07-27	2011-11-15	12355.00
16	Maurício dos Santos	1236665650	M	Brasil	Estados Unidos da América	2010-11-05	2012-11-05	22850.25
17	Mary Ann Duarte	1158854516	F	Brasil	México	2010-12-02	2012-12-30	18600.10
18	Gabriela Pereira	1159404014	F	Brasil	Dinamarca	2010-11-10	2012-03-02	19000.50
14	André César	2128002525	M	Brasil	Nova Zelândia	2010-06-04	2011-10-22	15300.00
14	Edson Lopes	2126900099	M	Brasil	Nova Zelândia	2010-06-04	2011-10-22	15300.00

(50 row(s) affected)



■ Subconsultas do tipo **WHERE**

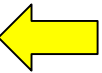
- O tipo mais comum de subconsulta utiliza uma subconsulta **SELECT** interna ao lado direito da expressão de comparação **WHERE**.
- O valor gerado pela subconsulta deve apresentar um tipo de dado que permite comparações.
- Subconsultas **WHERE** podem ser utilizadas em conjunto com junções.



■ Subconsultas do tipo WHERE

-- Exibe os dados dos países utilizados como
-- destino nas viagens dos alunos cadastrados,
-- cujo código seja 'USA'.

```
SELECT  Cod_Pais          AS 'Código',  
        Nome_pais        AS 'País de Destino',  
        Idioma_pais      AS 'Idioma'  
FROM    Países  
WHERE   Cod_Pais = (SELECT DISTINCT Pais_Destino  
                   FROM Viagens WHERE  
                   Pais_Destino = 'USA' )
```



■ Subconsultas do tipo WHERE

Código	País de Destino	Idioma
-----	-----	-----
USA	Estados Unidos da América	Inglês

(1 row(s) affected) ←



■ Subconsultas do tipo IN

- Esse tipo de subconsulta é utilizada quando se deseja comparar um atributo em relação a uma lista de valores.
- Os valores retornados pelo operador **IN** permitem as comparações envolvendo diversos valores, os quais são obtidos pelo comando **SELECT** e tratados como uma lista.



■ Subconsultas do tipo IN

-- Exibe os dados dos países utilizados como
-- destino nas viagens dos alunos cadastrados.

```
SELECT  Cod_Pais      AS 'Código',  
        Nome_pais     AS 'País de Destino',  
        Idioma_pais   AS 'Idioma'
```

```
FROM Países
```

```
WHERE Cod_Pais IN
```

```
(SELECT Pais_Destino FROM Viagens) ←
```

```
GO
```



■ Subconsultas do tipo IN

Código	País de Destino	Idioma

CHE	Suíça	Alemão e Francês
COG	Congo, República do	Francês
DNK	Dinamarca	Dinamarquês
FRA	França	Francês
.....		
TKL	Toquelau	Toquelauano e Inglês
TLS	Timor-Leste	Português
VGB	Virgens Britânicas, Ilhas	Inglês
VNM	Vietnã	Vietnamita

(21 row(s) affected) 



■ Subconsultas do tipo HAVING

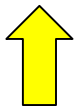
- Esse tipo de subconsulta é utilizada quando se deseja restringir o resultado de uma consulta envolvendo o operador **GROUP BY**.
- Para isso, aplicamos um critério condicional às linhas que estão sendo agrupadas.



■ Subconsultas do tipo HAVING

-- Exibe o código, nome e quantidade de viagens cadastradas para o país de
-- destino. Exibe somente as informações para os países cujo quantidade de
-- viagens seja maior ou igual a quantidade de viagens realizadas para o
-- México.

```
SELECT  P.Cod_Pais          AS 'Código',  
        P.Nome_pais        AS 'País de Destino',  
        COUNT(Cod_Pais)    AS 'Total de Viagens'  
FROM    Países P INNER JOIN Viagens V  
        On P.Cod_Pais = V.Pais_Destino  
GROUP BY P.Cod_Pais, P.Nome_pais  
HAVING  COUNT(P.Cod_Pais) >= (SELECT COUNT(Pais_Destino)  
                                FROM Viagens  
                                WHERE Pais_Destino = 'MEX')
```



■ Subconsultas do tipo HAVING

Código País de Destino		Total de Viagens
-----		-----
MEX	México	2
THA	Tailândia	2
USA	Estados Unidos da América	8

(3 row(s) affected) ←



■ Operadores ANY, SOME e ALL


- Os operadores **SOME** e **ANY** são semelhantes e funcionam de modo semelhante ao operador **IN**, porém retornam um valor booleano.
- Os valores para comparação sempre devem ser baseados em uma única coluna. Ao menos um dos valores deve atender a condição indicada.
- O operador **ALL** também possui um retorno booleano, porém neste caso somente retornará o resultado verdadeiro quando encontrar ao menos um valor comparado entre o conjunto de uma coluna.



■ Operadores ANY, SOME e ALL

-- Exibe os dados dos países utilizados como destino nas viagens dos alunos
-- cadastrados, desde que esses países sejam os Estados Unidos, México ou
-- Brasil.

```
SELECT Cod_Pais      AS 'Código',  
       Nome_pais     AS 'País de Destino',  
       Idioma_pais   AS 'Idioma'  
  
FROM Países  
  
WHERE Cod_Pais = ANY (SELECT Pais_Destino  
                      FROM Viagens  
                      WHERE Pais_Destino IN ('USA',  
                                             'MEX', 'BRA'))  
  
GO
```



■ Operadores ANY, SOME e ALL

Código	País de Destino	Idioma
-----	-----	-----
MEX	México	Espanhol ou Castelhanos
USA	Estados Unidos da América	Inglês

(2 row(s) affected) ←



■ Operadores ANY, SOME e ALL

- Exibe os dados dos alunos cujo código seja maior do que
- todos os valores da lista de parâmetros retornada por
- um construtor de valor de tabela.

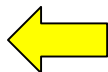
```
SELECT  Cod_Aluno  AS 'Código',  
        Nome_Aluno AS 'Nome do Aluno',  
        Endereco   AS 'Endereço'  
  
FROM Alunos  
  
WHERE Cod_Aluno > ALL (SELECT * FROM  
                        ↑ (VALUES (1), (3), (10))  
                          AS Codigos(a))
```



■ Operadores ANY, SOME e ALL

Código	Nome do Aluno	Endereço
11	Jorge Augusto	Rua Califórnia, 3451 - São Paulo
12	Renato César	Rua Cafezal, 10 - Cafezal do Sul
13	Júlio Antunes	Rua do Descanso, 99 - Camaçari
14	Márcio Gomes	Rua Miguel Pereira, 15 - São Paulo
15	Carlos Pereira	Rua Maria Augusta, 290 - São Paulo
16	Carlos Barreto	Rua Augusta, 90 - São Paulo
.....		
43	Heidi Lima	Rua César Conceição, 12 - Santo Antônio do Pinhal
44	Amílcar Júnior	Rua Senador Kennedy, 901 - São Paulo
45	Alexandro Duarte	Rua Maria Cíntia - Cruzeiro
46	Maurício dos Santos	Rua Marta Silva - Jacareí
47	Mary Ann Duarte	Av. Brasil, 6320 - São Paulo
48	Gabriela Pereira	Rua Franco Silva, 1599 - São Paulo
49	André César	Rua Militar, 349 - Rio de Janeiro
50	Edson Lopes	Av. Pedro Silva, 3047 - Rio de Janeiro

(40 row(s) affected)



■ Subconsultas do tipo FROM

- A cláusula **FROM** é utilizada para especificar as tabelas a partir das quais os dados serão obtidos.
- Como o resultado de um comando **SELECT** é outra tabela (tabela virtual), podemos utilizar uma subconsulta **SELECT** na cláusula **FROM**.
- Também podemos utilizar um nome de uma **VIEW** em qualquer posição de comando em que deve haver uma tabela.



■ Subconsultas do tipo FROM

-- Exemplo do livro

```
SELECT DISTINCT C.CUS_CODE,  
                C.L_NAME
```

```
FROM Customer C,
```

➔

```
(SELECT I.CUS_CODE  
    FROM Invoice I NATURAL JOIN Line L  
    WHERE L.P_CODE = '13-Q2/P2') CP1,
```

➔

```
(SELECT I.CUS_CODE  
    FROM Invoice I NATURAL JOIN Line L  
    WHERE L.P_CODE = '23109-HB') CP2
```

```
WHERE C.CUS_CODE = CP1_CUS_CODE AND CP1.CUS_CODE =  
CP2.CUS_CODE;
```



■ Subconsultas correlacionadas

- Uma **subconsulta correlacionada** é aquela que é executada uma vez para cada linha na pesquisa externa dentro de um comando SQL. A consulta é chamada correlacionada, pois a consulta interna é relacionada à externa pelo fato de referenciar uma coluna da subconsulta externa.
- As subconsultas correlacionadas também podem ser utilizadas em conjunto com o operador especial **EXISTS**.



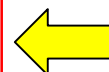
■ Subconsultas correlacionadas

- Exibe os dados dos alunos cujo código seja maior do que
- todos os valores da lista de parâmetros retornada por
- um construtor de valor de tabela.

```
SELECT  Cod_Aluno  AS 'Código',  
        Nome_Aluno AS 'Nome do Aluno',  
        Endereco   AS 'Endereço'
```

```
FROM Alunos
```

```
WHERE Cod_Aluno > ALL (SELECT * FROM  
                        (VALUES (1), (3), (10))  
                        AS Codigos(a))
```



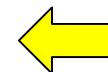
■ Subconsultas correlacionadas

-- Exibe informações sobre os alunos e as viagens que eles realizaram, somente caso
-- houve mais de 5 viagens para o mesmo país.

```
SELECT Viagens.Cod_Viagem AS 'Código da Viagem',  
       Alunos.Nome_Aluno AS 'Nome',  
       Alunos.Telefone,  
       Alunos.Sexo,  
       (SELECT Nome_pais FROM Países WHERE Cod_Pais = Alunos.Pais_Origem) AS 'Origem',  
       (SELECT Nome_pais FROM Países WHERE Cod_Pais = Viagens.Pais_Destino) AS 'Destino',  
       Viagens.Data_Saida AS 'Data de Saída',  
       Viagens.Data_Retorno AS 'Data de Retorno',  
       Viagens.Valor AS 'Preço da Viagem R$'  
FROM Alunos INNER JOIN Viagens  
      ON Alunos.Cod_Viagem = Viagens.Cod_Viagem
```

WHERE EXISTS

```
(SELECT Pais_Destino, COUNT(Pais_Destino)  
 FROM Viagens  
 GROUP BY Pais_Destino  
 HAVING COUNT(Pais_Destino) > 5)
```



GO



- **Na próxima aula veremos**
 - Manipulação de data e hora e strings.

