



LABORATÓRIO DE BANCO DE DADOS

IBD100

IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: SELEÇÃO
- Cria uma tabela, copiando as tuplas de outra tabela que atendem a uma condição.

- Dadas a tabela

- Em AR escreve-se:

$R \leftarrow A[[C2 \text{ contenha } W]]$

- R terá os mesmos campos de A e somente as tuplas que atendam a condição.

- Em SQL:

`SELECT * FROM A WHERE A.C2 LIKE '%w%';`

A	C1	C2	R	C1	C2
	Asd	Qwe		Asd	Qwe
	Rfv	Tgb			
	Zxc	Rty			
	Fgh	Erw		Fgh	Erw
	Wsx	Edc			

Diagram illustrating the selection operation. Table A contains 6 tuples. Table R contains 2 tuples. Red arrows indicate the selection of tuples from A that match the condition 'C2 tem W' (C2 contains W). The selected tuples are (Asd, Qwe) and (Fgh, Erw).

IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: SELEÇÃO

- Nota:

- Em AR escrevemos o condicional no modo fácil de ser entendido pelo programador.
- Em SQL o condicional pode depender das funções de ambiente do SGBD.
- O Resultado da operação pode ser uma tabela SEM TUPLAS (o que seria o conceito de conjunto vazio).
- A condição aceita qualquer tipo de operação, dependendo do SGBD aceitar ou não estas operações.
- A condição pode processar todos os operadores de comparação matematicamente aceitos, tais como:
 - =
 - $\geq$ ou $\geq$
 - $\leq$ ou $\leq$
 - $>$
 - $<$
 - $<>$ ou $\neq$ ou $\neq$
 - E, OU, OU exclusive, NÃO, ENTRE (interval fechado), DENTRE (entre discreto) e COMO (operadores lógicos)
- Escreva uma SELEÇÃO em alguma tabela da Base de Dados do PRATICASQL.

IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: SQL (SELECT)

- Implementa as sete operações da Álgebra Relacional
- Tem uma estrutura rígida mas com partes opcionais

- **SELECT** [ALL | DISTINCT [ON (expressao[,...])]]
* | **Campo1**[AS nome_saída1][,...]
FROM TAB1[, TAB2, TAB3, ... , TABn] | TAB1[**compl.** JOIN TAB2 ON condição]
[WHERE condição - seleção
[GROUP BY chave1[,chave2,...]]
[HAVING condição[,...]] [{ } **select**]
[ORDER BY campo1 [ASC|DESC|USING operador][,
campo2 [ASC|DESC|USING operador][,
...campon] ;

- Tudo o que está entre [] é opcional
- Note que o ';' é parte obrigatória no comando.

IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: SQL (SELECT)
- Descrevendo cada parte do comando...
- Não se esqueça de praticar este comando no ambiente do pgAdmin 4.

Parte	Ação
SELECT	Indica ao núcleo que deverá ser iniciado o processamento de leitura de dados das tabelas
* campo [as c1]	Indica quais campos devem ser exibidos no resultado do comando. Se usarmos * vemos todos os campos processados, senão devemos indicar quais queremos ver. O uso do "as" permite o renomear os campos dinamicamente.
FROM	Especifica a(s) tabela(s) a processar. No caso de mais de uma tabela, os nomes devem ser separados por ','. No caso de uso do JOIN o operador exige o 'ON' para declarar a condição de comparação dos campos envolvidos na condição da junção e o complemento (compl.) pode ser CROSS (Prod. Cartesiano), LEFT (esquerda), RIGHT (Direita), INNER (Natural – Fechada) e FULL OUTER (Esq.↔Dir. – Completa).
WHERE	Especifica uma condição do processamento das linhas da tabela. Pode-se combinar condições com operadores lógicos. Usa-se operadores de comparação específicos de cada SGBD. Os operadores Padrão ANSI/99 são: =, <>, >, <, >=, <=, IN, BETWEEN e os operadores lógicos são: AND, OR, NOT, XOR.
GROUP BY	Permite estabelecer agrupamentos de valores que podem apresentar a execução de funções escalares ou aritméticas
HAVING	Executam funções sobre valores de campos, podem indicar resultados que podem ser usados como parâmetro de agrupamento.
ORDER BY	Classificam os dados das tuplas segundo campos da tabela permitindo estabelecer organizações para execução de funções. As chaves de ordenação podem apresentar sub-chaves de ordenação e, para cada uma delas, ter ordens diferentes indicadas pelo uso de ASC ou DESC seguindo cada chave declarada.

IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: GERAL

- IMPORTANTE:

- Até aqui não escrevemos o comando SQL para o símbolo $\leftarrow$. Este símbolo da AR significa que estamos criando uma tabela que recebe os dados processados na operação.
- Em SQL usamos o comando DDL CREATE. Este comando (criando uma tabela indicando as características de cada campo) tem uma sintaxe 'quase' padrão, isso porque os SGBD NÃO tem um padrão para os tipos de dados. Essa diferença é o que caracteriza os nichos de mercados de cada SGBD. Então o SQL para o símbolo $\leftarrow$ informa o SGBD que deve ser criado o arquivo e lá armazenado o resultado de um comando SQL na forma:

CREATE [temporary] TABLE <nomedatab> AS (<Comando SQL>);

- O [temporary] informa ao SGBD que a tabela deve existir enquanto a sessão de conexão com o SGBD persistir. Na ação física a tabela criada pode ficar na memória RAM do servidor OU ser gravada no HD com um nome físico adotado pelo SGBD (isso vai depender do SGBD).
- O <nomedatab> NÃO PODE SER EXISTENTE se o [temporary] for omitido, pois neste caso a tabela será criada fisicamente na BASE de dados (o comando roda certo na primeira vez e errado a partir da segunda).

- Reescrevendo comandos:

- CREATE TEMPORARY TABLE R AS (SELECT * FROM A WHERE A.C2 LIKE '%w%'); ou
- CREATE TEMPORARY TABLE R AS (SELECT C2,C4 FROM A);

IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: PROJEÇÃO
- Cria uma tabela, copiando os dados de campos de uma tabela para outra tabela.

- Dadas a tabela

A	C1	C2	C3	C4
	Asd	Qwe	Fr4	Dfg
	Rfv	Tgb	Gt5	Edc
	Zxc	Rty	Hy6	Rfv
	Fgh	Ert	Ju7	Tgb
	Wsx	Rty	Ki8	Rfv

R	C2	C4
	Qwe	Dfg
	Tgb	Edc
	Rty	Rfv
	Ert	Tgb
	Rty	Rfv

- Em AR escreve-se:

$R \leftarrow A \text{ } [[\text{C2}, \text{C4}]]$

- R terá todos os dados dos campos de C2 e C4 da tabela A.

- Em SQL:

`SELECT C2,C4 FROM A;`

- Nota:

- R pode apresentar tuplas repetidas. Em AR Ok!! Mas no SQL pode ser que isso acarrete erro. Pode-se indicar que a operação deve ser feita sem a repetição de linhas com o uso de (DISTINTOS) no final da expressão. Assim

$R \leftarrow A \text{ } [[\text{C2}, \text{C4}]]$ (distintos)

- Pesquise como se faz o 'distinto' em SQL e execute o comando na base do PRATICASQL.

IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: PROJEÇÃO
- Nota:
 - Na projeção é possível trocar os nomes dos campos, para isso usa-se o símbolo $\rightarrow$ para indicar a troca, desse modo:
 $R \leftarrow A[[C1 \rightarrow \text{casa}, C2 \rightarrow \text{tel}]]$ produz uma tabela R com campos (casa,tel).
 - Em SQL teremos:
SELECT C1 AS casa, C2 AS tel FROM A;

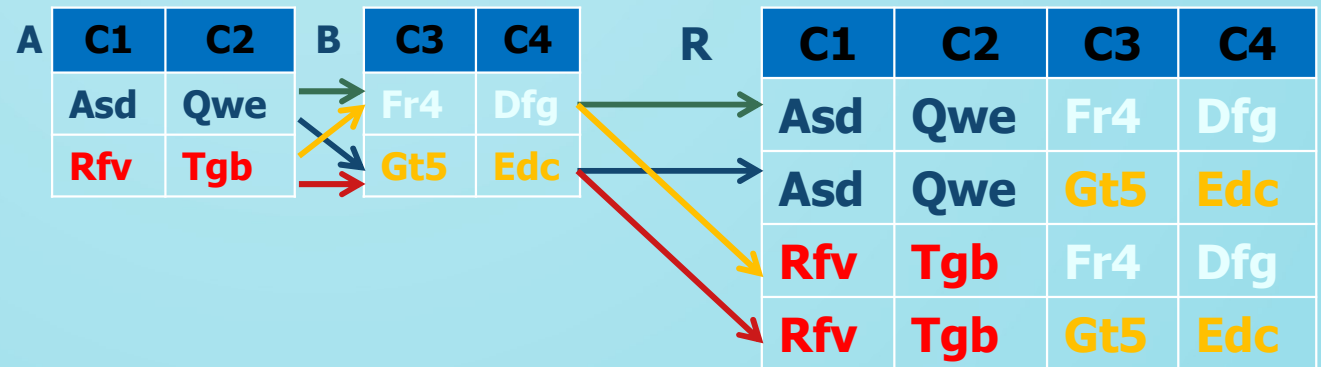
IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: PRODUTO CARTESIANO
- Cria uma tabela colocando cada uma das tuplas de uma tabela ao lado de cada uma das tuplas de outra tabela. Isto se diz "emparelhamento de tuplas";
- Esquema da tabela resultado é igual aos esquemas das tabelas colocados lado a lado, obedecendo, inclusive a ordem de campos.

- Dadas as tabelas:

escreve-se

$R \leftarrow A \times B$



IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: PRODUTO CARTESIANO

- Notas:

- A quantidade de tuplas da tabela resultado, será a quantidade de tuplas da primeira multiplicada pela quantidade de tuplas da segunda.
- É comutativo:

A X B

A	C1	C2	C3	C4	B
	Asd	Qwe	Fr4	Dfg	
	Rfv	Tgb	Gt5	Edc	

R

C1	C2	C3	C4
Asd	Qwe	Fr4	Dfg
Asd	Qwe	Gt5	Edc
Rfv	Tgb	Fr4	Dfg
Rfv	Tgb	Gt5	Edc

R

C1	C2	C3	C4
Asd	Qwe	Fr4	Dfg
Rfv	Tgb	Fr4	Dfg
Asd	Qwe	Gt5	Edc
Rfv	Tgb	Gt5	Edc

R

C3	C4	C1	C2
Fr4	Dfg	Asd	Qwe
Fr4	Dfg	Rfv	Tgb
Gt5	Edc	Asd	Qwe
Gt5	Edc	Rfv	Tgb

B		A	
C3	C4	C1	C2
Fr4	Dfg	Asd	Qwe
Gt5	Edc	Rfv	Tgb

Portanto: B X A

IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: PRODUTO CARTESIANO
- Sintaxe:

SELECT * FROM TABELA1, TABELA2; - SQL:1987

SELECT * FROM TABELA1 CROSS JOIN TABELA2 ON <CONDIÇÃO>; - SQL:1992

Combinado com o Create (exec. na base):

CREATE TABLE possiveisturmas AS

(SELECT * FROM disciplinas, professores); OU

CREATE TABLE possiveisturmas AS

(SELECT * FROM disciplinas CROSS JOIN professores);

Experimente combinar dados de duas tabelas quaisquer (sem relacionamento).

Experimente fazer um produto cartesiano de três tabelas.

Na AR isso não é possível, mas a SQL permite.

IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: JUNÇÃO NATURAL (ou fechada)
- Cria uma tabela a partir de duas outras emparelhando as tuplas e levando para o destino apenas as tuplas que obedecem a uma determinada condição;
- Esquema da tabela resultado é igual aos esquemas das tabelas colocados lado a lado, obedecendo, inclusive a ordem de campos.

Representação:

$R \leftarrow B[[\text{Condição}]]A$, por exemplo: $R \leftarrow B[[C3=C1]]A$ ou $R \leftarrow B[[B.C3=A.C1]]A$

B			A			R			
C3	C4		C1	C2		C3	C4	C1	C2
Fr4	Dfg	$\neq$	Asd	Qwe	$\rightarrow$	Asd	Edc	Asd	Qwe
Asd	Edc	$=$	Rfv	Tgb		Rfv	Jmh	Rfv	Tgb
Rfv	Jmh					Rfv	Vah	Rfv	Tgb
Rfv	Vah								

IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: JUNÇÃO NATURAL (ou fechada)

- Sintaxe:

```
SELECT * FROM TABELA1, TABELA2
      WHERE
      TABELA1.Campo1=TABELA2.Campo1;
```

} ANSI:87

- Ou também:

```
SELECT * FROM Tab1 INNER JOIN Tab2
      ON
      TABELA1.Campo1=TABELA2.Campo1;
```

} ANSI:92

- Ou também:

```
SELECT * FROM Tab1 INNER JOIN Tab2 USING (campo);
```

} SE a chave primária e a Chave estrangeira tiverem o mesmo NOME, então o SGBD pode aceitar esta versão.

- Exemplos:

```
Select * FROM departamentos, funcionarios
      WHERE
      departamentos.id_depto=funcionarios.id_depto;
```

```
Select * FROM departamentos INNER JOIN funcionarios
      ON
      departamentos.id_depto=funcionarios.id_depto;
```

```
Select * FROM departamentos INNER JOIN funcionarios
      USING (id_depto);
```

IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: JUNÇÃO NATURAL (ou fechada)
- Podemos também fazer algumas coisas mais complexas: "quais são os nomes de gerentes e seus subalternos para os departamentos com código do gerente entre os valores 230, 260 ou 290?"

- Podemos escrever:

```
SELECT F1.ID_Funcionario, F1.tx_primeiro_nome,  
       D1.id_depto, D1.tx_nome_depto, D1.ID_Funcionario_Gerente,  
       F2.ID_Funcionario, F2.tx_primeiro_nome  
FROM funcionarios as F1,  
     departamentos AS D1,  
     funcionarios AS F2  
WHERE F1.id_funcionario=D1.id_funcionario_gerente AND  
       D1.id_depto=F2.id_depto AND  
       F1.id_funcionario IN ('230','260','290');
```

- Note que 'funcionarios' aparece DUAS vezes na lista de tabelas, mas com DOIS nomes distintos (f1 e f2). Estes nomes são os que aparecem nas condições da junção.
- Em SQL:1992 este comando fica assim:

```
SELECT F1.ID_Funcionario, F1.tx_primeiro_nome,  
       D1.id_depto, D1.tx_nome_depto, D1.ID_Funcionario_Gerente,  
       F2.ID_Funcionario, F2.tx_primeiro_nome  
FROM funcionarios as F1 INNER JOIN departamentos AS D1  
                        ON F1.id_funcionario=D1.id_funcionario_gerente  
                        INNER JOIN funcionarios AS F2  
                        ON D1.id_depto=F2.id_depto  
WHERE F1.id_funcionario IN ('230','260','290');
```

- Os SGBDs resolvem os dois com a mesma performance.

- Operação | Comando: JUNÇÃO ABERTA

- No SQL:1992 foi apresentado o operador JOIN que consegue implementar as junções abertas.
- Na AR o símbolo usado será [[ou]] com o colchete externo voltado para fora indicando qual lado da junção está apresentando todas as linhas da tabela.
- Como em todas as operações de AR de junção, o esquema da tabela destino é o esquema de cada tabela lado a lado.
- A notação SQL:1992 vale para todas as versões seguintes do padrão ANSI/ISSO.

- Operação | Comando: JUNÇÃO ABERTA PELA DIREITA

- Cria uma tabela a partir de duas outras emparelhando todas as tuplas da tabela do lado direito com cada tupla da tabela da esquerda levando para o destino as tuplas que obedecem a uma determinada condição e colocando as tuplas que não atendem ao lado de registros com valores NULOS.

Representação:

$R \leftarrow B \text{ [Condição] } [A$

por exemplo:

$R \leftarrow B \text{ [C3=C1] } [A \text{ ou } B \text{ [B.C3=A.C1] } [A$

R

C3	C4	C1	C2
Asd	Edc	Asd	Qwe
Rfv	Jmh	Rfv	Tgb
Rfv	Vah	Rfv	Tgb
NULO	NULO	Wse	Tuj

B

C3	C4
Fr4	Dfg
Asd	Edc
Rfv	Jmh
Rfv	Vah

A

C1	C2
Asd	Qwe
Rfv	Tgb
Wse	Tuj

IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: JUNÇÃO ABERTA PELA DIREITA

- Em SQL, escreve-se:

```
select * from livros right join editoras  
ON editoras.pkeditora=livros.fkeditora;
```

- Em SQL pode-se combinar subcomandos e fazer duas (ou mais) operações de AR.

```
select editoras.* from livros right join editoras  
ON editoras.pkeditora=livros.fkeditora  
where pklivro IS NULL;
```

- Interprete o comando escrito acima.

- Operação | Comando: JUNÇÃO ABERTA PELA ESQUERDA

- Cria uma tabela a partir de duas outras emparelhando todas as tuplas da tabela do lado esquerdo com cada tupla da tabela da direita levando para o destino as tuplas que obedecem a uma determinada condição e colocando as tuplas que não atendem ao lado de registros com valores NULOS.

Representação:

$R \leftarrow B \text{] } [\text{Condição}] A$

por exemplo:

$R \leftarrow B \text{] } [C3=C1] A$ ou
 $B \text{] } [B.C3=A.C1] A$

R				B		A	
C3	C4	C1	C2	C3	C4	C1	C2
Asd	Edc	Asd	Qwe	Fr4	Dfg	Asd	Qwe
Rfv	Jmh	Rfv	Tgb	Asd	Edc	Rfv	Tgb
Rfv	Vah	Rfv	Tgb	Rfv	Jmh	Wse	Tuj
Fr4	Dfg	NULO	NULO	Rfv	Vah		

IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: JUNÇÃO ABERTA PELA ESQUERDA

- Em SQL, escreve-se:

```
select * from livros left join editoras  
ON editoras.pkeditora=livros.fkeditora;
```

- Em SQL pode-se combinar subcomandos e fazer duas (ou mais) operações de AR.

```
select livros.* from livros left join editoras  
ON editoras.pkeditora=livros.fkeditora  
where pkeditora IS NULL;
```

- Interprete o comando escrito acima.

IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: **JUNÇÃO COMPLETA** (aberta pelos dois lados)
 - Cria uma tabela a partir de duas outras emparelhando todas as tuplas das tabelas que obedecem a uma determinada condição e colocando as tuplas que não atendem ao lado de registros com valores NULOS.

Representação:

$R \leftarrow B \text{] [Condição] [A$

por exemplo:

$R \leftarrow B \text{] [C3=C1] [A ou}$
 $B \text{] [B.C3=A.C1] [A$

R			
C3	C4	C1	C2
Asd	Edc	Asd	Qwe
Rfv	Jmh	Rfv	Tgb
Rfv	Vah	Rfv	Tgb
Fr4	Dfg	NULO	NULO
NULO	NULO	Wse	Tuj

B	
C3	C4
Fr4	Dfg
Asd	Edc
Rfv	Jmh
Rfv	Vah

A	
C1	C2
Asd	Qwe
Rfv	Tgb
Wse	Tuj

- Operação | Comando: JUNÇÃO COMPLETA (aberta pelos dois lados)

- Em SQL, escreve-se:

```
select * from livros full outer join editoras  
ON editoras.pkeditora=livros.fkeditora;
```

- Interprete o comando

```
select pklivro, txtituloacervo, pkeditora, txnomeeditora  
from livros full outer join editoras  
ON editoras.pkeditora=livros.fkeditora;
```

IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: **UNIÃO**
- Cria uma tabela a partir de duas outras levando as tuplas comuns e não comuns;
- A Operação só pode ser feita se as tabelas de origem forem **União Compatíveis**;
- O Esquema da tabela resultado é igual ao esquema de cada uma das tabelas.

Representação:

$$R \leftarrow A \cup B$$

Esta repetição é 'correta'.

Na implementação as linguagens facilitam e colocam operadores que já omitem a repetição.

R	C1	C2	A	C1	C2	B	C1	C2
	Zxc	Rty		Asd	Qwe		Zxc	Rty
	Fgh	Ert		Rfv	Tgb		Fgh	Ert
	Rfv	Tgb					Rfv	Tgb
	Asd	Qwe						
	Rfv	Tgb						

Notas:

A União é comutativa (a ordem dos fatores na operação não altera o resultado da operação);

$$R \leftarrow A \cup B = B \cup A$$

As Tabelas geradas por operações de Álgebra Relacional são Livres;

Podemos escrever complementos aos comandos da Álgebra Relacional para tratamento de prováveis desvios do Modelo.

A Operação acima poderia ser melhor escrita assim:

$$R \leftarrow A \cup B \text{ (Distintos)}$$

- Operação | Comando: **UNIÃO**

- Em SQL escreve-se

- Sintaxe:

- `SELECT * FROM tabela1 UNION ALL (SELECT * FROM tabela2);`

- Executando na Base:

```
create temporary table f1 as
```

```
(Select * from funcionarios where fkdepto IN ('A01','E11'));
```

```
create temporary table f2 as
```

```
(Select * from funcionarios where fkdepto IN ('D11','E11'));
```

```
SELECT * FROM f1 UNION (SELECT * FROM f2);
```

```
drop table f1, f2;
```

- Tente usar `UNION ALL` e observe a diferença no resultado do processamento do commando.

IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: INTERSECÇÃO
- Cria uma tabela a partir de duas outras levando as tuplas que estão na primeira Tabela E TAMBÉM estão na Segunda;
- A Operação só pode ser feita se as tabelas de origem forem União Compatíveis;
- Esquema da tabela resultado é igual ao esquema de cada uma das tabelas.

Representação:

$$R \leftarrow A \cap B$$

Notas:

A Interseção é comutativa (a ordem dos fatores na operação NÃO ALTERA o resultado da operação);

$$R \leftarrow A \cap B = B \cap A$$

O Resultado da operação pode ser uma tabela SEM TUPLAS (o que seria o conceito de conjunto vazio).

R	C1	C2	A	C1	C2	B	C1	C2
	Rfv	Tgb		Asd	Qwe		Zxc	Rty
				Rfv	Tgb		Fgh	Ert
							Rfv	Tgb

- Operação | Comando: INTERSECÇÃO

- Sintaxe:

```
SELECT * FROM TAB1 INTERSECT (SELECT * FROM TAB2); ou
```

```
SELECT * FROM TAB1 WHERE EXISTS
```

```
(SELECT * FROM TAB2 WHERE TAB1.CAMPO1=TAB2.CAMPO1 AND
```

```
TAB1.CAMPO2=TAB2.CAMPO2 AND...);
```

- Exemplificando com operações na base de dados:

```
create temporary table func1 as
```

```
(Select * from funcionarios where fkdepto IN ('A01','E11'));
```

```
create temporary table func2 as
```

```
(Select * from funcionarios where fkdepto IN ('D11','E11'));
```

```
SELECT * FROM func1 INTERSECT (SELECT * FROM func2);
```

```
drop table func1, func2;
```

- Nota:

- Alguns SGBDs seguem o padrão SQL:1992 MAS NÃO implementam o INTERSECT.
 - Para cada SGBD temos que estudar e ver como implementa a intersecção.

- Operação | Comando: INTERSECÇÃO

- Usando o WHERE EXISTS:

```
create temporary table func1 as
    (Select * from funcionarios where fkdepto IN ('A01','E11'));
create temporary table func2 as
    (Select * from funcionarios where fkdepto IN ('D11','E11'));
SELECT * FROM func1 WHERE EXISTS
    (SELECT * FROM func2 WHERE func1.pkfuncionario = func2.pkfuncionario);
drop table func1, func2;
```

- Nota:

- Em teoria, no comando acima, deveria ter sido comparado TODOS os campos de funcionários. Entretanto, como `pkfuncionario` é a chave primária da tabela `funcionarios`, então foi possível conseguir o mesmo resultado comparando-se somente o campo `'pkfuncionario'`.

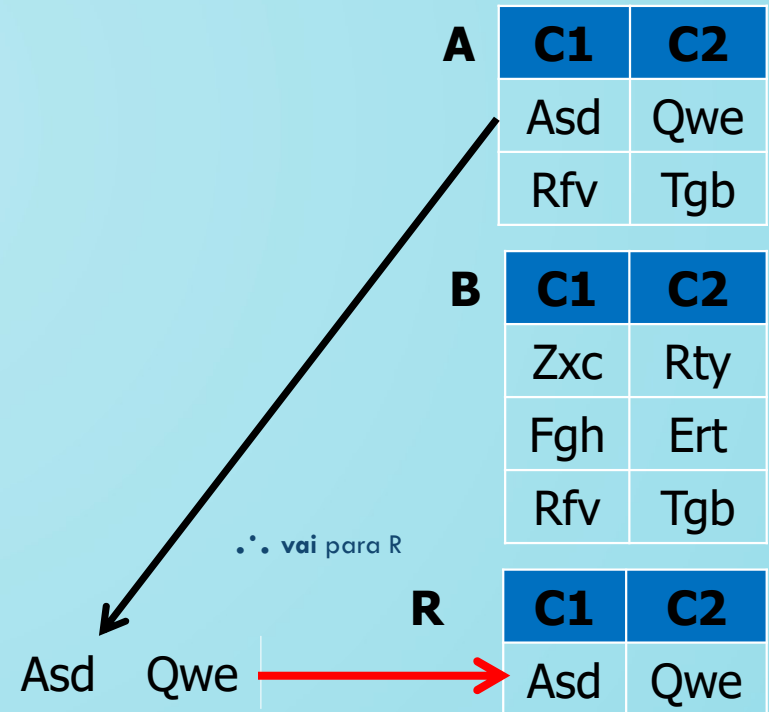
IBD100-ÁLGEBRA RELACIONAL + SQL

- Operação | Comando: SUBTRAÇÃO
- Cria uma tabela a partir de duas outras levando as tuplas que estão na Primeira Tabela e que NÃO estão na Segunda;
- A Operação só pode ser feita se as tabelas de origem forem União Compatíveis;
- Esquema da tabela resultado é igual ao esquema de cada uma das tabelas.

Representação:

$R \leftarrow A - B$

Esta Tupla ESTÁ em A
e NÃO ESTÁ em B



$T \leftarrow B - A$

∴ Não comutativa

T

C1	C2
Zxc	Rty
Fgh	Ert

- Operação | Comando: SUBTRAÇÃO

- Sintaxe:

```
Select * FROM TABELA1 MINUS | EXCEPT (Select FROM TABELA2);
```

- Exemplificando na base de dados

```
create temporary table func1 as
```

```
(Select * from funcionarios where fkdepto IN ('A01','E11'));
```

```
create temporary table func2 as
```

```
(Select * from funcionarios where fkdepto IN ('D11','E11'));
```

```
select * from func1 except (select * from func2);
```

- Tente executar o comando de subtração com o 'minus' e veja o acontece.
- Tente tirar o () e ver o acontece. Por que?



LABORATÓRIO DE BANCO DE DADOS

IBD100