WWW.UNICARIOCA.EDU.BR

Banco de Dados II

Tema 6

Profª Giselle Batalha

MELHOR CENTRO UNIVERSITÁRIO DO RIO DE JANEIRO!

Fonte: MEC

SQL – ÁLGEBRA RELACIONAL

A grande maioria das aplicações que desenvolvemos atualmente utilizam um banco de dados relacional o que implica na utilização de consultas para obtenção de resultados.

Para isso usamos a linguagem SQL que é a linguagem de pesquisa declarativa padrão para banco de dados relacional.

Muitas das características originais do SQL foram inspiradas na álgebra relacional.

UNI CARIOCA
Centro Universitário

SQL – ÁLGEBRA RELACIONAL

A álgebra relacional consiste em um conjunto de operações utilizadas para manipular relações.

Uma consulta em um banco de dados que segue o modelo relacional é realizada através da aplicação de uma serie de operações da álgebra relacional que são executadas e retornam os dados na forma de uma tabela.

UNI CARIOCA
Centro Universitário

União 

Produz como resultado uma Relação que contém todas as linhas da primeira Relação seguidas de todas as linhas da segunda tabela. A Relação resultante possui a mesma quantidade de colunas que as relações originais, e tem um número de linhas que é no máximo igual à soma das linhas das relações fornecidas como operandos, já que as linhas que são comuns a ambas as relações aparecem uma única vez no resultado.

Retorna a união das tuplas de duas relações R1 e R2 com eliminação automática de duplicatas

SQL – ÁLGEBRA RELACIONAL

União ∪

Notação: Relação1 ∪ Relação2 (R1 ∪ R2)

Obs: As relações devem possuir o mesmo número de atributos.

Alunos

id	nome	idade	curso
10	Macoratti	45	Quimica
20	Miriam	43	Artes
30	Bianca	21	Fisica

Professores

id	nome	idade	setor
100	Pedro	50	Quimica
200	Maria	45	Fisica
300	Bianca	21	Artes

Funcionarios

id	nome	setor	idade
10	Margarida	Quimica	46
20	Jamil	Fisica	32

Domínio:

id = int
nome = varchar(30)
idade = int
curso = varchar(30)
setor = varchar(30)

A relação **Alunos** é compatível com **Professores** mas **não é compatível** com **Funcionarios.**

Ex1: Encontre uma relação com todos os alunos e com todos os professores:

resultado: **Alunos ∪ Professores**

id	nome	idade	curso
10	Macoratti	45	Quimica
20	Miriam	43	Artes
30	Bianca	21	Fisica
100	Pedro	50	Quimica
200	Maria	45	Fisica
300	Bianca	21	Artes

Diferença -

É uma operação que requer como operandos duas relações união-compatíveis, ou seja, estruturalmente idênticas. O resultado é uma relação que possui todas as linhas que existem na primeira relação e não existem na segunda.

Retorna as tuplas presentes em R1 e ausentes em R2

Diferença -

Notação : relação1 - relação2 (R1 - R2)

Alunos (R1)

id	nome	idade	curso
10	Macoratti	45	Quimica
20	Miriam	43	Artes
30	Bianca	21	Fisica

Professores(R2)

id	nome	idade	setor
100	Pedro	50	Quimica
200	Maria	45	Artes
300	Bianca	21	Fisica

Domínio:

id = int
nome = varchar(30)
idade = int
curso = varchar(30)
setor = varchar(30)

Ex1 : Apresente uma relação de todos os alunos que não são professores

Resultado : Aluno - Professor

id	nome	idade	curso
10	Macoratti	45	Quimica
20	Miriam	43	Artes

Note-se que a DIFERENÇA não é comutativa !

Resultado : Professor - Aluno

id	nome	idade	setor
100	Pedro	50	Quimica
200	Maria	45	Artes

UNI CARIOCA
Centro Universitário

Interseção ∩

Esta é uma operação que produz como resultado uma tabela que contém, sem repetições, todos os elementos que são comuns às duas tabelas fornecidas como operandos. As tabelas devem ser união-compatíveis..

Retorna as tuplas comuns a R1 e R2

Interseção ∩

Notação : relação1 ∩ relação2 (R1 ∩ R2)

Alunos(R1)

id	nome	idade	curso
10	Macoratti	45	Quimica
20	Miriam	43	Artes
30	Bianca	21	Fisica

Professores(R2)

id	nome	idade	setor
100	Pedro	50	Quimica
200	Maria	45	Artes
300	Bianca	21	Fisica

Domínio:

id = int
nome = varchar(30)
idade = int
curso = varchar(30)
setor = varchar(30)

Ex1 : Apresente uma relação de todos os alunos que são professores;

Resultado : **Alunos ∩ Professores**

id	nome	idade	curso
30	Bianca	21	Fisica

UNI CARIOCA
Centro Universitário

Junção Natural $|X|$

O resultado da operação junção natural é uma relação com todas as combinações das tuplas na relação1 (R1) e relação2 (R2) nas quais os seus atributos em comum são iguais.

É uma operação que produz uma combinação entre as linhas de uma relação com as linhas correspondentes de outra relação, sendo em princípio correspondente a uma seleção pelos atributos de relacionamento sobre um produto cartesiano dessas relações.

Retorna a combinação de tuplas de duas relações R1 e R2 que satisfazem um predicado

Junção Natural |X|

Notação: R1 |x| R2

No exemplo a seguir temos as relações **Empregados e Setores** a sua junção natural :

Empregados

id	nome	setor
100	Macoratti	Admin
200	Jefferson	Contab
300	Bianca	Admin
400	Janice	Contab

Setores

setor	gerente
Admin	Paulino
Contab	Amelia
RH	Francisca

|X|

Empregados |x| Setores

id	nome	setor	gerente
100	Macoratti	Admin	Paulino
200	Jefferson	Contab	Amelia
300	Bianca	Admin	Paulino
400	Janice	Contab	Amelia

A junção natural pode ser vista como uma combinação de uma operação de seleção aplicada sobre uma operação de produto cartesiano:

σ <critério> (<relação1> X <relação2)

UNI CARIOCA
Centro Universitário

Produto Cartesiano X

O resultado do produto cartesiano de duas relações é uma terceira relação contendo todas as combinações possíveis entre os elementos das relações originais.

Retorna todas as combinações de tuplas de duas R1 e R2

Produto Cartesiano X

Alunos

id	nome	sexo	curso
123	Macoratti	M	100
234	Miriam	F	110

Cursos

id	nome
100	Quimica
110	Inglês
120	Matemática
130	Física

Alunos X Cursos

id	nome	sexo	curso	id	nome
123	Macoratti	M	100	100	Quimica
123	Macoratti	M	100	110	Inglês
123	Macoratti	M	100	120	Matematica
123	Macoratti	M	100	130	Fisica
234	Miriam	F	110	100	Quimica
234	Miriam	F	110	110	Inglês
234	Miriam	F	110	120	Matematica
234	Miriam	F	1100	130	Fisica

UNI CARIOCA
Centro Universitário

BANCO DE DADOS BASE PARA ESSA AULA

```
create database aula9;
use aula9;
create table CIDADE(idCidade integer primary key not null,
                    nome varchar(25));

create table CLIENTE(idCliente integer primary key not null,
                     nome varchar(25) null,
                     idcidade integer null,
                     foreign key (idcidade) references CIDADE(idCidade));


insert into CIDADE values (1, 'Rio de Janeiro'),
                          (2, 'São Paulo'),
                          (3, 'Salvador'),
                          (4, 'Vitória');


insert into CLIENTE values (100, 'Jupira',1),
                           (101, 'Jucilda',2),
                           (102, 'Joslane',3),
                           (103, 'Jospirane',1),
                           (104, 'Jerilene',2),
                           (105, 'Jercenilda',1);
```

UNI CARIOCA Centro Universitário

Utilizaremos o script tema6 disponível na Midiateca como base para essa aula

CHAVE PRIMÁRIA E ESTRANGEIRA

- As chaves são usadas para garantir integridade num banco de dados. Após sua definição, o próprio SGBD se encarrega de fazer as validações.
- A **chave primária** (PK) define um identificador único para cada tupla. Seu uso garante que não haverão 2 registros com valores repetidos na tabela. Só é possível criar uma chave primária por tabela. É possível criar chaves primárias compostas por 2 ou mais campos.
- A **chave estrangeira** (FK) garante que a informação inserida no campo seja consistente. Ou seja, não é permitido gravar um valor na chave estrangeira se esse valor não existir na tabela referenciada. Uma tabela pode ter diversas chaves estrangeiras, cada uma referenciando sua tabela de origem.

UNI CARIOCA
Centro Universitário

Recuperando Dados de Várias Tabelas (JOINS)

Necessidade de acessar simultaneamente várias tabelas relacionadas entre si.

Qualificadores de Nome

- Qualificador de nome consiste no nome da tabela seguido de um ponto e o nome da coluna na tabela, por exemplo:

  PRODUTO.descricao

- Os qualificadores de nome são utilizados em uma consulta para efetivar a junção (JOIN) entre tabelas, uma vez que o relacionamento entre tabelas é realizado por meio de chaves estrangeiras.

JOIN via Cláusula WHERE

- **Não recomendada.**
- **Consulta:** Listar os nomes dos clientes e suas respectivas cidades.

```
SELECT cliente.nome, cidade.nome
FROM cliente,cidade
WHERE cliente.idcidade = cidade.idcidade;
```

Result Grid | Filter Rows:

nome	nome
Jupira	Rio de Janeiro
Jospirane	Rio de Janeiro
Jercenilda	Rio de Janeiro
Jucilda	São Paulo
Jerilene	São Paulo
Joslane	Salvador

Result 1

UNI CARIOCA Centro Universitário

Se não fizer a junção, será gerado um produto cartesiano:

```
SELECT cliente.nome, cidade.nome
FROM cliente,cidade;
```

nome	nome
Jupira	Rio de Janeiro
Jupira	São Paulo
Jupira	Salvador
Jupira	Vitória
Jucilda	Rio de Janeiro
Jucilda	São Paulo
Jucilda	Salvador
Jucilda	Vitória
Joslane	Rio de Janeiro
Joslane	São Paulo
Joslane	Salvador
Joslane	Vitória
Jospir	Rio de Janeiro
Jospir	São Paulo
Jospir	Salvador
Jospir	Vitória
Jerilene	Rio de Janeiro
Jerilene	São Paulo
Jerilene	Salvador

ALIAS (Apelido) nas Tabelas

```
SELECT cl.nome, ci.nome

FROM cliente cl , cidade ci

WHERE cl.idcidade = ci.idcidade;
```

Result Grid | Filter Rows:

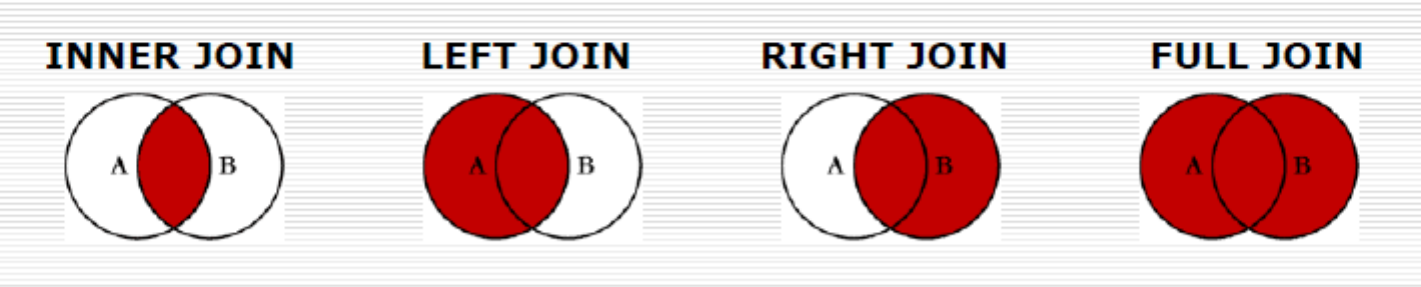
	nome	nome
▶	Jupira	Rio de Janeiro
	Jospirane	Rio de Janeiro
	Jercenilda	Rio de Janeiro
	Jucilda	São Paulo
	Jerilene	São Paulo
	Joslane	Salvador

UNI CARIOCA
Centro Universitário

Comparação entre os tipos de JOIN

•Não é possível afirmar que um tipo de instrução JOIN seja melhor que outro.

•É preciso analisar qual a informação desejada e utilizar a instrução adequada.

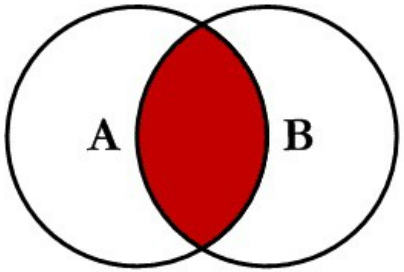


UNI CARIOCA
Centro Universitário

Junção Estilo ANSI: JOIN ... ON ...

```
SELECT   cl.nome, ci.nome
FROM  cliente cl INNER JOIN cidade ci
            ON cl.idCidade = ci.idCidade;
```

O Inner Join é o método de junção mais conhecido e, como ilustra a figura abaixo, retorna os registros que são comuns às duas tabelas.



Result Grid | Filter Rows:

	nome	nome
▶	Jupira	Rio de Janeiro
	Jospirane	Rio de Janeiro
	Jercenilda	Rio de Janeiro
	Jucilda	São Paulo
	Jerilene	São Paulo
	Joslane	Salvador

UNI CARIOCA Centro Universitário

```
insert into CLIENTE values (106, 'Jopiri',null),    (107, 'Juremilde',null);
```

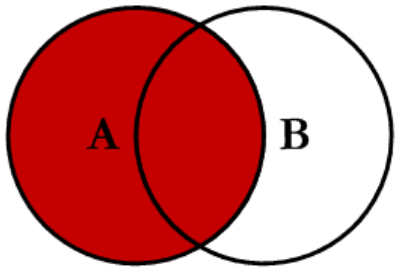
Junção Estilo ANSI: LEFT JOIN ... ON ...

```
SELECT   cl.nome, ci.nome

FROM  cliente cl LEFT JOIN cidade ci

        ON cl.idCidade = ci.idCidade;
```

O Left Join, cujo funcionamento é ilustrado na figura abaixo, tem como resultado todos os registros que estão na tabela A (mesmo que não estejam na tabela B) e os registros da tabela B que são comuns à tabela A.



Result Grid | Filter Rows:

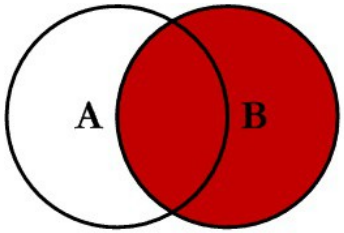
	nome	nome
▶	Jupira	Rio de Janeiro
	Jucilda	São Paulo
	Joslane	Salvador
	Jospirane	Rio de Janeiro
	Jerilene	São Paulo
	Jercenilda	Rio de Janeiro
	Jopiri	NULL
	Juremilde	NULL

Result 7

Junção Estilo ANSI: JOIN ... ON ...

```
SELECT   cl.nome, ci.nome
FROM  cliente cl RIGHT JOIN cidade ci
            ON cl.idCidade = ci.idCidade;
```

Usando o Right Join, conforme mostra a figura abaixo, teremos como resultado todos os registros que estão na tabela B (mesmo que não estejam na tabela A) e os registros da tabela A que são comuns à tabela B.



Result Grid | Filter Rows:

nome	nome
Jupira	Rio de Janeiro
Jospirane	Rio de Janeiro
Jercenilda	Rio de Janeiro
Jucilda	São Paulo
Jerilene	São Paulo
Joslane	Salvador
NULL	Vitória