



WWW.UNICARIOCA.EDU.BR

Banco de Dados II

Tema 9 – Parte 2

Profª Giselle Batalha



MELHOR CENTRO UNIVERSITÁRIO DO RIO DE JANEIRO!

BANCO BASE PARA ESSA AULA

```
create database tema9p2;
```

```
use tema9p2;
```

```
3 create table ALUNO(  
  matALUNO int not null primary key,  
  nome varchar(30),  
  sexo varchar(10),  
  ~ media decimal(4,10));
```

```
3 create table CURSO(  
  idCURSO int not null primary key,  
  nome varchar(30),  
  turno varchar(10),  
  ~ investimento decimal(8,2));
```

```
/*cadastrar os dados na tabela ALUNO */
```

```
insert into ALUNO values (20171001, "ANA SILVA", "FEMININO",9.0),  
                          (20171002, "PEDRO SILVA", "MASCULINO",6.0),  
                          (20171003, "ROBERTO SILVA", "MASCULINO",8.0),  
                          (20171004, "ELAINE CRISTINA", "FEMININO",5.0),  
                          (20171005, "ARNALDO JORGE", "MASCULINO",7.0);
```

```
/*cadastrar os dados na tabela TURMA */
```

```
insert into CURSO values (1, "BD MODELAGEM", "MANHÃ",1000),  
                          (2, "BD SQL", "TARDE",2000),  
                          (3, "ALGORITMOS", "TARDE",3000),  
                          (4, "COBOL", "NOITE",4000),  
                          (5, "PROGRAMAÇÃO", "NOITE",3000);
```

Uma **VIEW** é uma tabela virtual gerada a partir de uma consulta, na qual as informações são obtidas diretamente nas tabelas usadas nas consultas. Atualizando-se os dados das tabelas de origem, **automaticamente** atualiza-se a **VIEW**.

Tipos de Visão:

1. Visão Idêntica
2. Visão por Seleção de Linhas
3. Visão por Seleção de Colunas
4. Visão por Seleção de Linhas e Colunas
5. Visão por Junção de Tabelas

Vantagens do uso de VIEW

- Uma das vantagens da utilização de **VIEW** está na facilidade de se manipular dados originados de diferentes tabelas.
- Outra vantagem se relaciona a segurança, pois é possível 'ocultar' campos ou dados através da consulta.
- Propicia a independência física da arquitetura do banco de dados.

Criação de Visões

CREATE VIEW <nome_da_visao> AS <expressão_consulta>

onde, <expressão_consulta> é qualquer consulta SQL válida.

Exemplo:

```
CREATE VIEW view_alunos AS  
SELECT matAluno, nome  
FROM aluno;
```

Criação de Visões

*Para acessar uma **VIEW**, usamos o comando **SELECT** como se fosse uma tabela normal (inclusive via linguagem de programação)*

```
select * from view_alunos;
```

CREATE VIEW

42 •

43

```
select * from view_alunos;
```

Result Grid



Filter Rows:

Export:



Wrap Cell

	matAluno	nome
▶	20171001	ANA SILVA
	20171002	PEDRO SILVA
	20171003	ROBERTO SOUZA
	20181001	ELAINE CRISTINA
	20181002	ARNALDO JORGE

Alterando uma visão

ALTER VIEW <nome_da_visao> AS <expressão_consulta>

onde, <expressão_consulta> é qualquer consulta SQL válida.

Exemplo:

```
ALTER VIEW view_alunos AS  
SELECT matAluno, nome  
FROM aluno order by nome;
```


CREATE VIEW

```
43 |  
44 • ALTER VIEW view_alunos AS  
45     SELECT matAluno, nome  
46         FROM aluno order by nome;  
47 • select * from view_alunos ;
```

< **Result Grid**   Filter Rows: Export:  Wrap Cell Content: 

	matAluno	nome
▶	20171001	ANA SILVA
	20181002	ARNALDO JORGE
	20181001	ELAINE CRISTINA
	20171002	PEDRO SILVA
	20171003	ROBERTO SOUZA

Exclusão de uma VIEW

DROP VIEW <nome_da_visao>

onde, <expressão_consulta> é qualquer consulta SQL válida.

Exemplo:

DROP VIEW view_alunos;

- ✓ Os índices são utilizados, principalmente, para melhorar o desempenho do banco de dados (embora a utilização não apropriada possa resultar em uma degradação de desempenho).
- ✓ O índice do banco de dados segue a mesma filosofia de um índice de um livro: achar a informação que procura mais rapidamente.
- ✓ O comando CREATE INDEX constrói o índice nome_do_índice na tabela especificada.

- ✓ Estudos apontam que 80% das atualizações de hardware por novos equipamentos seriam desnecessários, bastando apenas o ajuste adequado do software.
- ✓ ATENÇÃO: O uso excessivo de indexação pode tornar o acesso lento!!!
- ✓ Assim, deve-se criar índices apenas para tabelas com grande número de registros (alguns milhares, pelo menos) e para campos que sejam frequentemente utilizados em buscas.

ÍNDICE PRIMÁRIO

MATR	TSL
001	220
002	321
003	231
005	110
.	.

ÍNDICE SECUNDÁRIO

NOME	TSL
Ana	321
José	220
José	231
Maria	110
.	331

TRILHA, SETOR E LADO DO DISCO

(endereço físico)

chave primária (endereço lógico)

TSL	MATR	NOME	ENDEREÇO	DT_NASC
110	005	Maria	SQS 308 ...	23/08/78
231	003	José	QND 14	25/09/70
321	002	Ana	SQN 410 ...	10/08/85
220	001	José	GAMA	05/04/76
331

Instrução:

```
CREATE [UNIQUE] INDEX <nome_indice>  
ON <nome_tabela> (<nome_coluna> [ASC / DESC])
```

Exemplo:

```
CREATE INDEX idx_CURSO_TURN0  
ON CURSO(TURNO);  
describe CURSO;
```

CREATE INDEX

```
48
49 • CREATE INDEX idx_CURSO_TURN0
50 ON CURSO(TURNO);
51
52 • describe CURSO;
```

Result Grid |  Filter Rows: | Export:  | Wrap Cell Content

	Field	Type	Null	Key	Default	Extra
▶	idCURSO	smallint(6)	NO	PRI	NULL	
	nome	varchar(20)	YES		NULL	
	turno	varchar(10)	YES	MUL	NULL	
	investimento	decimal(8,2)	YES		NULL	

CREATE INDEX

```
48  
49 • CREATE UNIQUE INDEX idx_CURSO_NOME  
50 ON CURSO(NOME);  
51  
52 • describe CURSO;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	Field	Type	Null	Key	Default	Extra
▶	idCURSO	smallint(6)	NO	PRI	NULL	
	nome	varchar(20)	YES	UNI	NULL	
	turno	varchar(10)	YES	MUL	NULL	
	investimento	decimal(8,2)	YES		NULL	

Result 21 x

CREATE INDEX

54 • `insert into CURSO values(5, "COBOL", "NOITE", 4000);`

Output

Action Output

#	Time	Action	Message
56	22:32:34	insert into CURSO values(5, "COBOL", "NOITE", 4000)	Error Code: 1062. Duplicate entry '5' for key 'PRIMARY'

53
54 • `insert into CURSO values(6, "COBOL", "NOITE", 4000);`

Output

Action Output

#	Time	Action	Message
59	22:34:38	insert into CURSO values(6, "COBOL", "NOITE", 4000)	Error Code: 1062. Duplicate entry 'COBOL' for key 'idx_CURSO_NOME'

CREATE INDEX

Instrução:

SHOW INDEX FROM <nome_tabela>;

Exemplo:

```
SHOW INDEX FROM CURSO;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index
curso	0	PRIMARY	1	idCURSO	A	0	NULL	NULL		BTREE		
curso	0	ix_CURSO_NOME	1	nome	A	5	NULL	NULL	YES	BTREE		
curso	1	idx_CURSO_TURN0	1	turno	A	3	NULL	NULL	YES	BTREE		

Instrução:

DROP INDEX <nome_indice> **ON** <nome_tabela>;

```
52 • DROP INDEX idx_CURSO_NOME on CURSO;  
53 • describe CURSO;  
54
```



Result Grid



Filter Rows:

Export:



Wrap Cell Content: [IA](#)

	Field	Type	Null	Key	Default	Extra
▶	idCURSO	smallint(6)	NO	PRI	NULL	
	nome	varchar(20)	YES		NULL	
	turno	varchar(10)	YES	MUL	NULL	
	investimento	decimal(8,2)	YES		NULL	