



# SQL para MySQL

Instituto Federal de São Paulo  
Prof. Bianca M Pedrosa



# SQL

- *Structured Query Language*
- Linguagem de Banco de Dados que inclui comandos para definição (DDL), Consulta (QL) e Manipulação de dados(DML).
- Versões
  - SQL 1 - 1986 (ANSI+ISO)
  - SQL 2 - 1992
  - SQL 3 - 1999

# Convenções

- Para definir os comandos serão adotadas as seguintes convenções:

[...] opcional

<...> a ser substituído por um nome apropriado

**Texto** palavra reservada

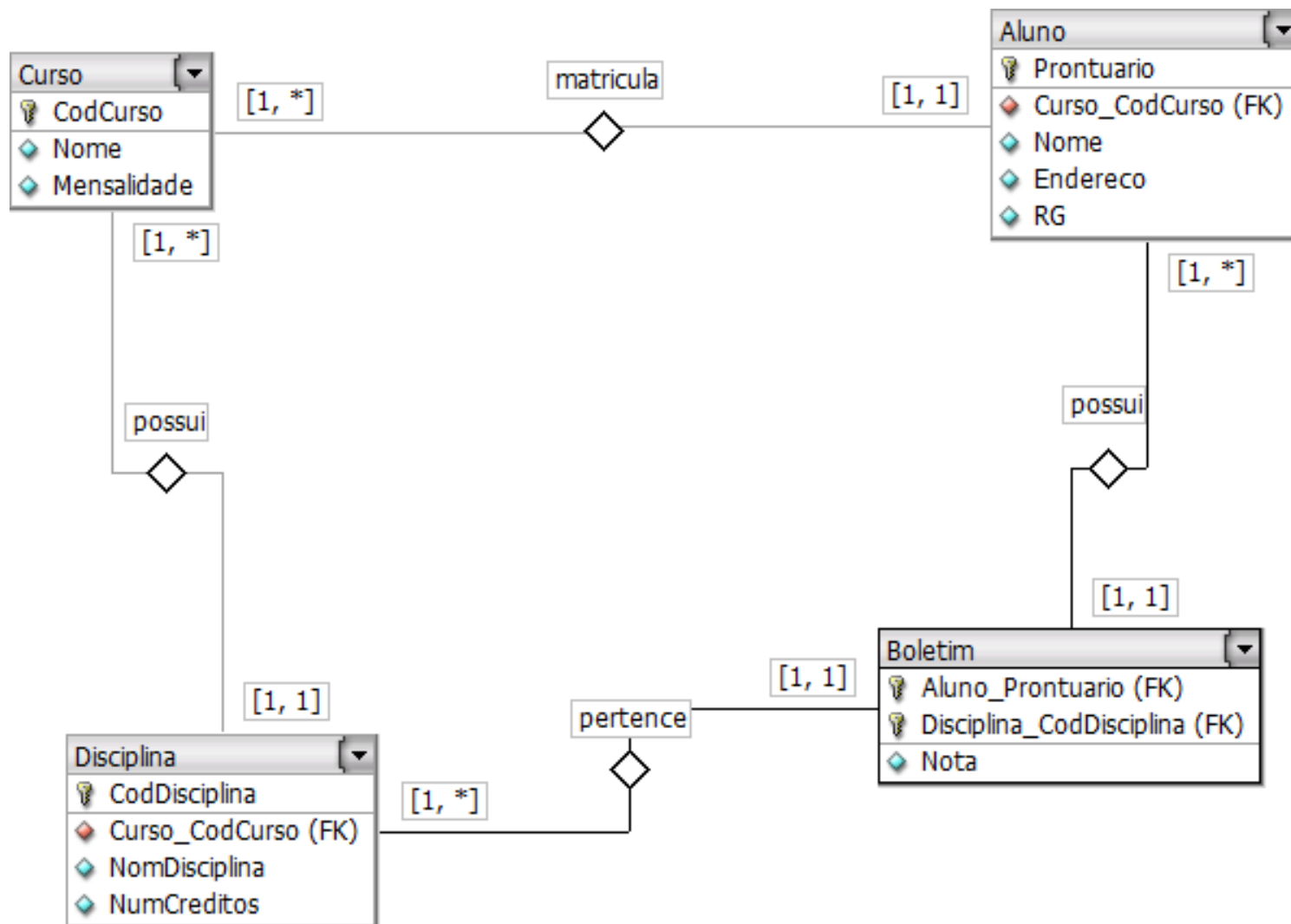
| ou



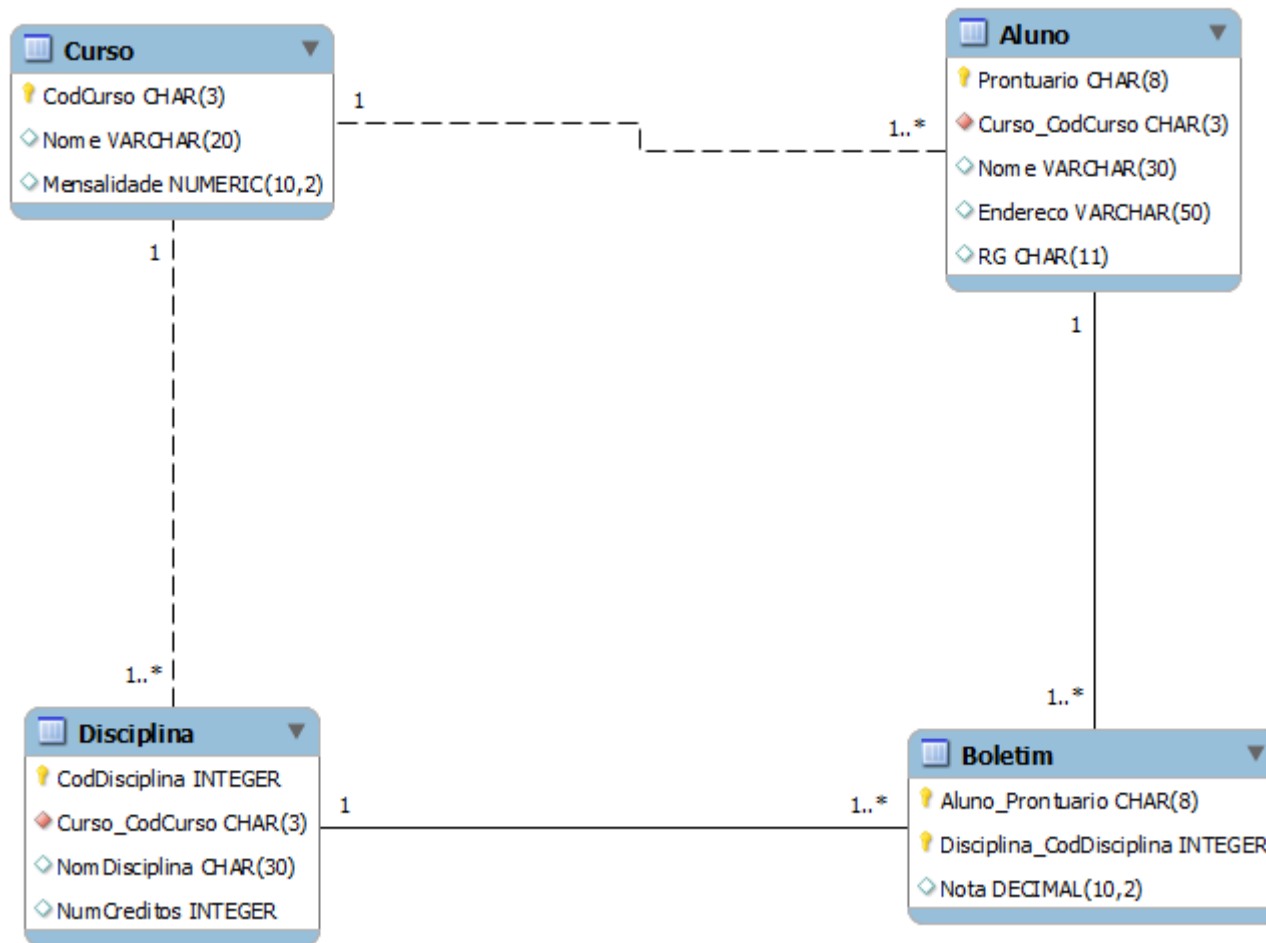
# DEFINIÇÃO DE DADOS

- CREATE DATABASE
- CREATE TABLE
- ALTER TABLE
- DROP TABLE
- DROP DATABASE

# Sistema Acadêmico



# Sistema Acadêmico



# CREATE DATABASE

- Cria um Banco de Dados –

## **CREATE DATABASE [IF NOT EXISTS]**

**<banco\_dados>;**

Exemplo:

**CREATE DATABASE IF NOT EXISTS ACADEMICO;**

Efeito:

Cria o Banco de dados Academico, caso este não exista.

# CREATE TABLE

- Cria uma tabela –

```
CREATE TABLE [IF NOT EXISTS] <tabela> (  
    <coluna> <tipo_coluna> [NULL/NOT NULL]  
                                [DEFAULT VALOR]  
                                [AUTO_INCREMENT]  
                                [PRIMARY KEY]  
                                [FOREIGN KEY]);
```



# CREATE TABLE

Ex.:

**CREATE TABLE** CURSO

(  
 CODCURSO            **CHAR (3) NOT NULL,**  
 NOME                **CHAR (30),**  
 MENSALIDADE        **DECIMAL(6,2),**  
 **PRIMARY KEY** (CODCURSO)

);

**CREATE TABLE** ALUNO

(  
 RA                    **CHAR(9) NOT NULL PRIMARY KEY ,**  
 RG                    **CHAR(9) NOT NULL,**  
 NOME                 **CHAR(30),**  
 CODCURSO            **CHAR(3),**  
 **FOREIGN KEY** (CODCURSO) **REFERENCES** CURSO (CODCURSO)  
);

# CREATE TABLE

**CREATE TABLE** DISCIPLINA

```
(  CodDisc      CHAR(5) NOT NULL,  
   Nome         CHAR(30),  
   CodCurso     CHAR(3),  
   NroCreditos int,  
   PRIMARY KEY (CodDisc),  
   FOREIGN KEY (CodCurso) REFERENCES CURSO(codcurso)  
);
```

**CREATE TABLE** BOLETIM

```
(  Ra           CHAR(9) NOT NULL,  
   CodDisc     CHAR(5) NOT NULL,  
   NotaAlu     DECIMAL(5,2),  
   PRIMARY KEY (Ra,CodDisc),  
   FOREIGN KEY (Ra) REFERENCES ALUNO (Ra),  
   FOREIGN KEY (CodDisc) REFERENCES DISCIPLINA(CodDisc)  
);
```

# Restrições da Foreign Key

[**ON DELETE** {CASCADE | SET NULL | NO ACTION | RESTRICT} ]

[**ON UPDATE** {CASCADE | SET NULL | NO ACTION | RESTRICT}]

## ■ Cascade

- Se um registro na tabela pai for deletado, então todos os registros de da outra tabela, cujos valores de chaves estrangeiras são iguais ao valor da chave referenciada serão apagados/atualizados. Ex: Se apagar um curso, apaga também os alunos daquele curso (Cuidado!!)

## ■ Set Null

- Se um registro na tabela pai for deletado, então todos os registros de da outra tabela, cujos valores de chaves estrangeiras são iguais ao valor da chave referenciada serão automaticamente atualizados com o valor **NULL** do SQL.

## ■ Restrict

- se existir um registro filho com diversos registros pais, o não permite a deleção/atualização de qualquer um dos registros pais.

# Resumo dos tipos de dados

<b>Tipo</b>	<b>Descrição</b>
<b>CHAR</b> (Tamanho Fixo)	Caracter Tamanho Fixo
<b>VARCHAR</b> (Tamanho Máximo )	Caracter Tamanho Variável
<b>INTEGER, INT, SMALLINT, MEDIUMINT, BIGINT</b>	Inteiro (sem casas decimais)
<b>DECIMAL</b> (tamanho, casas decimais)	Real
<b>FLOAT</b> (tamanho, casas decimais)	Real
<b>DATE</b>	Data
<b>TIME</b>	Hora
<b>ENUM</b>	Lista de valores string
<b>BOOLEAN</b>	Lógico (TRUE, FALSE)
<b>BLOB</b>	Binário Longo

# ALTER TABLE

- Altera a estrutura de uma tabela para adicionar (ADD) ou excluir (DROP) colunas e restrições –

```
ALTER TABLE <tabela> ADD <coluna> <tipo> (  
    [NOT NULL]  
    [DEFAULT VALOR]  
    [AUTO_INCREMENT]  
    [PRIMARY KEY]  
    | DROP <coluna>  
  
);
```



# ALTER TABLE

Ex.:

- **alter table** aluno **ADD** endereco char (35);
- **alter table** aluno **DROP** endereco ;
- **alter table** aluno **ADD CONSTRAINT** fkmatricula **FOREIGN KEY** (codcurso) **REFERENCES** curso;



# DROP TABLE

-Apaga uma tabela-

Ex.:

**DROP TABLE** BOLETIM;



# DROP DATABASE

-Apaga um Banco de Dados -

Ex.:

**DROP DATABASE** academico;



# Comandos MySQL

<b>Show Databases;</b>	Mostra os banco de dados criados
<b>Use</b> <banco>;	Seleciona um banco de dados
<b>Show Tables;</b>	Mostra as tabelas do banco de dados selecionado
<b>Desc</b> <tabela>;	Descreve a estrutura de uma tabela



# MANIPULAÇÃO DE DADOS

- INSERT INTO
- DELETE FROM
- UPDATE

# BD Acadêmico

## ALUNO

RA	RG	NOME	CODCURSO
123	12345	BIANCA MARIA PEDROSA	AS
212	21234	TATIANE CITTON	AS
221	22145	ALEXANDRE PEDROSA	CC
231	23144	ALEXANDRE MONTEIRO	CC
321	32111	MARCIA RIBEIRO	CC
661	66123	JUSSARA MARANDOLA	SI
765	76512	WALTER RODRIGUES	SI

## CURSO

CODCURSO	NOME	MENSALIDADE
AS	ANALISE DE SISTEMAS	R\$ 1.000,00
CC	CIENCIA DA COMPUTACAO	R\$ 950,00
SI	SISTEMAS DE INFORMACAO	R\$ 800,00

## BOLETIM

Ra	CodDisc	NotaAlu
123	BDS	10
212	IBD	7,5
231	BD	9
231	BDA	9,6
661	DBD	8
765	DBD	6

## DISCIPLINA

CodDisc	Nome	CodCurso	NrCreditos
BD	BANCO DE DADOS	CC	4
BDA	BANCO DE DADOS AVANCADOS	CC	6
BDOO	BANCO DE DADOS O OBJETOSZ	SI	4
BDS	SISTEMAS DE BANCO DE DADOS	AS	4
DBD	DESENVOLVIMENTO BANCO DE DADOS	SI	6
IBD	INTRODUCAO A BANCO DE DADOS	AS	2

# INSERT INTO

- Insere dados em uma tabela -

Ex.:

**INSERT INTO**

aluno (RA, RG, NOME, CODCURSO)

**VALUES** ('721', '72122', 'CLAUDIO ROGERIO', 'SI');

## ANTES

RA	RG	NOME	CODCURSO
123	12345	BIANCA MARIA PEDROSA	AS
212	21234	TATIANE CITTON	AS
221	22145	ALEXANDRE PEDROSA	CC
231	23144	ALEXANDRE MONTEIRO	CC
321	32111	MARCIA RIBEIRO	CC
661	66123	JUSSARA MARANDOLA	SI
765	76512	WALTER RODRIGUES	SI

## DEPOIS

RA	RG	NOME	CODCURSO
123	12345	BIANCA MARIA PEDROSA	AS
212	21234	TATIANE CITTON	AS
221	22145	ALEXANDRE PEDROSA	CC
231	23144	ALEXANDRE MONTEIRO	CC
321	32111	MARCIA RIBEIRO	CC
661	66123	JUSSARA MARANDOLA	SI
721	72122	CLAUDIO ROGERIO	SI
765	76512	WALTER RODRIGUES	SI



# INSERT INTO

Obs: para inserir dados na ordem das colunas, e sem a omissão de nenhuma coluna, podemos omitir o nomes das colunas:

```
INSERT INTO aluno  
VALUES ('721', '72122', 'CLAUDIO ROGERIO', 'SI');
```



# Inserindo Cursos

- > insert into CURSO VALUES  
('AS','ANALISE DE SISTEMAS',1000);
- > insert into CURSO VALUES  
('CC','CIENCIA DA COMPUTACAO',950);
- > insert into CURSO VALUES  
('SI','SISTEMAS DE INFORMACAO',800);



# Inserindo Alunos

> insert into ALUNO values  
( '123','12345','BIANCA MARIA PEDROSA','AS '),  
( '212','21234','TATIANE CITTON','AS'),  
( '221','22145','ALEXANDRE PEDROSA ','CC'),  
( '231','23144','ALEXANDRE MONTEIRO','CC'),  
( '321','32111','MARCIA RIBEIRO ','CC'),  
( '661','66123','JUSSARA MARANDOLA','SI'),  
( '765','76512','WALTER RODRIGUES','SI');



# Inserindo Notas

- > insert into BOLETIM values ('123','BDS',10);
- > insert into BOLETIM values ('212','IBD',7.5);
- > insert into BOLETIM values ('231','BD',9);
- > insert into BOLETIM values ('231','BDA',9.6);
- > insert into BOLETIM values ('661','DBD',8);
- > insert into BOLETIM values ('765','DBD',6);





# Inserindo Disciplinas

- > insert into DISCIPLINA values  
('BD','BANCO DE DADOS','CC',4);
- > insert into DISCIPLINA values  
('BDA','BANCO DE DADOS AVANCADOS','CC',6);
- > insert into DISCIPLINA values  
('BDOO','BANCO DE DADOS O OBJETOS','SI',4);
- > insert into DISCIPLINA values  
('BDS','SISTEMAS DE BANCO DE DADOS','AS', 4);
- > insert into DISCIPLINA values  
('DBD','DESENVOLVIMENTO BANCO DE DADOS','SI',6);
- > insert into DISCIPLINA values  
('IBD','INTRODUCAO A BANCO DE DADOS','AS',2);

# DELETE FROM

- Apaga linhas de uma tabela -

**Ex.:**

**DELETE FROM ALUNO  
WHERE NOME='MARCIA RIBEIRO'**

**ANTES**

RA	RG	NOME	CODCURSO
123	12345	BIANCA MARIA PEDROSA	AS
212	21234	TATIANE CITTON	AS
221	22145	ALEXANDRE PEDROSA	CC
231	23144	ALEXANDRE MONTEIRO	CC
321	32111	MARCIA RIBEIRO	CC
661	66123	JUSSARA MARANDOLA	SI
721	72122	CLAUDIO ROGERIO	SI
765	76512	WALTER RODRIGUES	SI

**DEPOIS**

RA	RG	NOME	CODCURSO
123	12345	BIANCA MARIA PEDROSA	AS
212	21234	TATIANE CITTON	AS
221	22145	ALEXANDRE PEDROSA	CC
231	23144	ALEXANDRE MONTEIRO	CC
661	66123	JUSSARA MARANDOLA	SI
721	72122	CLAUDIO ROGERIO	SI
765	76512	WALTER RODRIGUES	SI

# UPDATE

- Altera o conteúdo de linhas da tabela -

**Ex.:**

**UPDATE CURSO**

**SET Mensalidade = 1.2 \* Mensalidade;**

**ANTES**

CODCURSO	NOME	MENSALIDADE
AS	ANALISE DE SISTEMAS	R\$ 1.000,00
CC	CIENCIA DA COMPUTACAO	R\$ 950,00
SI	SISTEMAS DE INFORMACAO	R\$ 800,00

**DEPOIS**

CODCURSO	NOME	MENSALIDADE
AS	ANALISE DE SISTEMAS	R\$ 1.200,00
CC	CIENCIA DA COMPUTACAO	R\$ 1.140,00
SI	SISTEMAS DE INFORMACAO	R\$ 960,00



# CONSULTAS

SELECT	<lista de Atributos>
FROM	<lista de tabelas>
WHERE	<condição>
ORDER BY	<lista de atributos>
GROUP BY	< lista de atributos>
HAVING	<condição relativa a group by>



# CONSULTAS

- SELECT \*
- FROM \*
- WHERE
- GROUP BY
- HAVING
- ORDERED BY

\* apenas SELECT e FROM são obrigatórias



# CONSULTA SIMPLES

**SELECT NOME, RA**  
**FROM ALUNO**

NOME	RA
BIANCA MARIA PEDROSA	123
TATIANE CITTON	212
ALEXANDRE PEDROSA	221
ALEXANDRE MONTEIRO	231
JUSSARA MARANDOLA	661
CLAUDIO ROGERIO	721
WALTER RODRIGUES	765

# CONSULTA COM CONDIÇÃO

```
SELECT NOME, RA  
FROM ALUNO  
WHERE Codcurso='AS';
```

se não tem certeza, use:

```
WHERE upper(codcurso)='AS'
```

NOME	RA
BIANCA MARIA PEDROSA	123
TATIANE CITTON	212



# OPERADORES

- BETWEEN
- LIKE
- IN



# BETWEEN

- Selecciona tuplas entre dois valores -

- Ex.: Selecione os cursos com mensalidades entre 1000 e 1500

**SELECT \* FROM** curso

**WHERE** mensalidade **BETWEEN** 1000 **AND** 1500

CODCURSO	NOME	MENSALIDADE
AS	ANALISE DE SISTEMAS	R\$ 1.200,00
CC	CIENCIA DA COMPUTACAO	R\$ 1.140,00

# LIKE

- Selecciona tuplas utilizando curingas -

Ex.: Selecione os cursos cujo NOME contenham a palavra 'SISTEMAS'

Select \* from curso  
where nome **like** '%SISTEMAS%'

CODCURSO	NOME	MENSALIDADE
AS	ANALISE DE SISTEMAS	R\$ 1.200,00
SI	SISTEMAS DE INFORMACAO	R\$ 960,00

# IN

- Selecciona tuplas a partir de uma lista de valores -

- Ex.: Selecione os alunos que dos cursos 'AS' ou 'CC'

Select \* from aluno

where codcurso **in** ('AS', 'CC')

RA	RG	NOME	CODCURSO
123	12345	BIANCA MARIA PEDROSA	AS
231	23144	ALEXANDRE MONTEIRO	CC
221	22145	ALEXANDRE PEDROSA	CC
212	21234	TATIANE CITTON	AS

# CONSULTA COM ORDENAÇÃO

```
SELECT A.NOME, A.RA  
FROM   ALUNO A  
WHERE  A.CODCURSO='AS'  
ORDER BY A.NOME DESC;
```

NOME	RA
TATIANE CITTON	212
BIANCA MARIA PEDROSA	123

Notas:

- 1) **ASC (CRESCENTE)** é o padrão (pode ser omitido)
- 2) **DESC =DECRESCENTE**

# CONSULTA COM ORDENAÇÃO

```
SELECT A.NOME, A.RA  
FROM   ALUNO A  
WHERE  A.CODCURSO='AS'  
ORDER BY 1 DESC;
```

NOME	RA
TATIANE CITTON	212
BIANCA MARIA PEDROSA	123

## Notas:

- 1) A cláusula ORDER BY pode ser usada de forma posicional, ISTO É, pode-se fazer referência a uma coluna pela posição que ela ocupa na cláusula SELECT



# Junções

# JOIN ou JUNÇÃO

$$\text{Relação1} \mid \underset{\text{condição}}{X} \mid \text{Relação2}$$

- Combina uma tupla de cada relação, se esta combinação satisfizer a condição
- A condição de uma Junção envolve sempre atributos das duas relações e tem a forma:  $A_i$  operador  $B_j$ , onde  $A_i$  e  $B_j$  são atributos das duas relações. Os operadores da condição de uma Junção são: =, <, <=, >, >=, #, AND, OR, NOT

# Exemplo de uma junção

**Select \***

**From** curso **inner join** aluno **using** (curso)

CODCURSO	NOME	MENSALIDADE
AS	ANALISE DE SISTEMAS	1000
CC	CIENCIA DA COMPUTACAO	950
SI	SISTEMAS DE INFORMACAO	800

RA	RG	NOME	CODCURSO
212	21234	TATIANE CITTON	AS
221	22145	ALEXANDRE PEDROSA	CC
231	23144	ALEXANDRE MONTEIRO	CC
661	66123	JUSSARA MARANDOLA	SI
765	76512	WALTER RODRIGUES	SI

CODCURSO	RA	RG	NOME	NOME	MENSALIDADE
AS	212	21234	TATIANE CITTON	ANALISE DE SISTEMAS	1000
CC	221	22145	ALEXANDRE PEDROSA	CIENCIA DA COMPUTACAO	950
CC	231	23144	ALEXANDRE MONTEIRO	CIENCIA DA COMPUTACAO	950
SI	661	66123	JUSSARA MARANDOLA	SISTEMAS DE INFORMACAO	800
SI	765	76512	WALTER RODRIGUES	SISTEMAS DE INFORMACAO	800



# Junção Externa Esquerda

Neste caso a tabela aluno irá toda para o resultado, incluindo alunos que não tenham ra em boletim

**SELECT \***  
**FROM** aluno **LEFT OUTER JOIN** boletim **USING**( ra )

aluno

RA	RG	NOME	CODCURSO
212	21234	TATIANE CITTON	AS
221	22145	ALEXANDRE PEDROSA	CC
231	23144	ALEXANDRE MONTEIRO	CC
661	66123	JUSSARA MARANDOLA	SI
765	76512	WALTER RODRIGUES	SI

boletim

Ra	CodDisc	NotaAlu
212	IBD	7.5
231	BD	9
231	BDA	9.6
661	DBD	8
765	DBD	6

resultado

RA	RG	NOME	CODCURSO	Ra	<u>CodDisc</u>	<u>NotaAlu</u>
212	21234	TATIANE CITTON	AS	212	IBD	7.5
221	22145	ALEXANDRE PEDROSA	CC	NULL	NULL	NULL
231	23144	ALEXANDRE MONTEIRO	CC	231	BD	9
231	23144	ALEXANDRE MONTEIRO	CC	231	BDA	9.6
661	66123	JUSSARA MARANDOLA	SI	661	DBD	8
765	76512	WALTER RODRIGUES	SI	765	DBD	6

Obs: outer pode ser omitido em mysql

Neste caso a tabela disciplina irá toda para o resultado, incluindo disciplinas que não tenham coddisc em boletim

# Junção Externa Direita

**SELECT \***

**FROM** boletim **RIGHT OUTER JOIN** disciplina **USING**( coddisc )

boletim

Ra	CodDisc	NotaAlu
212	IBD	7.5
231	BD	9
231	BDA	9.6
661	DBD	8
765	DBD	6

CodDisc	Nome	CodCurso	NroCreditos
BD	BANCO DE DADOS	CC	4
BDA	BANCO DE DADOS AVANCADOS	CC	6
BDOO	BANCO DE DADOS O OBJETOS	SI	4
BDS	SISTEMAS DE BANCO DE DADOS	AS	4
DBD	DESENVOLVIMENTO BANCO DE DADOS	SI	6
IBD	INTRODUCAO A BANCO DE DADOS	AS	2

disciplina

resultado

CodDisc	Nome	CodCurso	NroCreditos	Ra	NotaAlu
BD	BANCO DE DADOS	CC	4	231	9
BDA	BANCO DE DADOS AVANCADOS	CC	6	231	9.6
BDOO	BANCO DE DADOS O OBJETOS	SI	4	NULL	NULL
BDS	SISTEMAS DE BANCO DE DADOS	AS	4	NULL	NULL
DBD	DESENVOLVIMENTO BANCO DE DADOS	SI	6	661	8
DBD	DESENVOLVIMENTO BANCO DE DADOS	SI	6	765	6
IBD	INTRODUCAO A BANCO DE DADOS	AS	2	212	7.5

Obs. Outer pode ser omitido em mysql

# Sintaxe

tabela\_ref, tabela\_ref

tabela\_ref                    [INNER | CROSS] JOIN                    table\_reference [join\_condition]

tabela\_ref                    STRAIGHT\_JOIN                    tabela\_ref

tabela\_ref                    LEFT [OUTER] JOIN                    table\_reference [join\_condition]

tabela\_ref                    NATURAL [LEFT [OUTER]] JOIN tabela\_ref

{ OJ tabela\_ref LEFT OUTER JOIN tabela\_ref ON expr\_condicional }

tabela\_ref RIGHT [OUTER] JOIN table\_reference [join\_condition]

tabela\_ref NATURAL [RIGHT [OUTER]] JOIN tabela\_ref

- Onde **tabela\_ref** é definido como:
  - nome\_tabela [[AS] alias] [[USE INDEX (lista\_indice)] | [IGNORE INDEX (lista\_indice)] | [FORCE INDEX (lista\_indice)]]
- a **condição\_join** é definido como:
  - ON expr\_condicional |
  - USING (lista\_colunas)

# Join ou Inner Join

```
SELECT *  
FROM `aluno` , curso  
WHERE aluno.codcurso = curso.codcurso
```

<u>RA</u>	<u>RG</u>	<u>NOME</u>	<u>CODCURSO</u>	<u>CODCURSO</u>	<u>NOME</u>	<u>MENSALIDADE</u>
123	12345	BIANCA MARIA PEDROSA	AS	AS	ANALISE DE SISTEMAS	1000
212	21234	TATIANE CITTON	AS	AS	ANALISE DE SISTEMAS	1000
221	22145	ALEXANDRE PEDROSA	CC	CC	CIENCIA DA COMPUTACAO	950
231	23144	ALEXANDRE MONTEIRO	CC	CC	CIENCIA DA COMPUTACAO	950
321	32111	MARCIA RIBEIRO	CC	CC	CIENCIA DA COMPUTACAO	950
661	66123	JUSSARA MARANDOLA	SI	SI	SISTEMAS DE INFORMACAO	800
765	76512	WALTER RODRIGUES	SI	SI	SISTEMAS DE INFORMACAO	800



# Funções de Agregação

COUNT, SUM, AVG, MIN, MAX

# FUNÇÕES DE AGREGAÇÃO

- Funções de agregação efetuam cálculos sobre grupos de linhas –

Ex.:Liste o nome e número de alunos em cada curso

```
SELECT C.Nome,  
       Count(A.CodCurso) AS NALUNOS  
FROM   CURSO C INNER JOIN ALUNO A  
       ON  C.CodCurso = A.CodCurso  
GROUP BY C.Nome
```

Nome	ALUNOS
ANALISE DE SISTEMAS	2
CIENCIA DA COMPUTACAO	2
SISTEMAS DE INFORMACAO	3



# FUNÇÕES DE AGREGAÇÃO

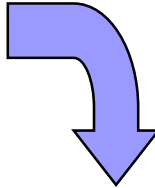
- COUNT
- SUM
- AVG
- MIN
- MAX

# Funções de Agregação e Groupby

- Quando as funções de agregação são usadas com a cláusula group by, as linhas resultantes da consulta são agrupadas segundo o critério de agrupamento (group by)
- Ex.

```
SELECT C.Nome,  
       Count(A.CodCurso) AS NALUNOS  
FROM   CURSO C INNER JOIN ALUNO A  
       ON C.CodCurso = A.CodCurso  
GROUP BY C.Nome
```

Nome	ALUNOS
ANALISE DE SISTEMAS	2
CIENCIA DA COMPUTACAO	2
SISTEMAS DE INFORMACAO	3



<u>RA</u>	<u>RG</u>	<u>NOME</u>	<u>CODCURSO</u>
123	12345	BIANCA MARIA PEDROSA	AS
212	21234	TATIANE CITTON	AS
221	22145	ALEXANDRE PEDROSA	CC
231	23144	ALEXANDRE MONTEIRO	CC
321	32111	MARCIA RIBEIRO	CC
661	66123	JUSSARA MARANDOLA	SI
765	76512	WALTER RODRIGUES	SI



# HAVING

- A cláusula HAVING permite a definição de condições para a exibição do resultado da função de agrupamento –

Ex.: Exibir apenas os cursos com pelo menos 3 alunos

```
SELECT      C.Nome,  
            Count(A.CodCurso) AS N_ALUNOS  
FROM        CURSO C INNER JOIN ALUNO A  
ON C.codcurso = A.CodCurso  
GROUP BY   C.Nome  
HAVING      COUNT(A.CODCURSO) >= 3;
```

Nome	N_ALUNOS
SISTEMAS DE INFORMACAO	3

# CÓPIA DE TABELAS

- Insere os dados resultantes de uma consulta em uma tabela –

Ex.:

```
CREATE TABLE ALUNOCC AS  
SELECT *  
FROM ALUNO  
WHERE CODCURSO='CC';
```



# VISÕES

- Uma visão é uma representação dos dados armazenados em uma ou mais tabelas.
- Uma visão não corresponde necessariamente à forma como os dados estão armazenados
- Usa-se visões para restringir o acesso aos dados por alguns usuários.
- Uma visão pode ser utilizada na cláusula from de outras consultas SQL
- Visões são criadas com o comando CREATE VIEW

# VIEW

```
CREATE VIEW name [(col1, col2, ..., coln)]  
AS <select> [WITH CHECK OPTION];
```

obs:

- (col<sub>1</sub>, col<sub>2</sub>, ..., col<sub>n</sub>) (opcional): corresponde às colunas a serem adicionadas à visão
- <select> (obrigatório) corresponde a uma expressão de consulta qualquer
- WITH CHECK OPTION (opcional): evita a atualização e inclusão de dados que não satisfaçam a cláusula select

# VIEW

- **CREATE VIEW ALUNOS\_AS AS**  
**SELECT \***  
**FROM** alunos  
**WHERE** codcurso='AS';

Use a view em  
uma expressão  
de consulta

```
SELECT *  
FROM ALUNOS_AS
```



# Manipulação de Datas

# Tipos de Dados

Tipo de Dados	Valor
DATETIME	'aaaa-mm-dd hh:mm:ss'
DATE	'aaaa-dd-mm'
TIMESTAMP	aaaammddhhmmss (tamanho depende do tamanho do display)
TIME	'hh:mm:ss'
YEAR	aaaa

# Funções para obter data/hora atual

Função	Retorno
NOW()	Retorna a data e hora atuais (aaaa-mm-dd hh:mm:ss)
CURDATE()	Retorna a data atual (aaaa-mm-dd ) do sistema
CURTIME()	Retorna a hora atual (hh:mm:ss) do sistema
UTC_DATE()	Retorna a data atual (aaaa-mm-dd ) fuso horário UTC
UTC_TIME()	Retorna a hora atual (hh:mm:ss) fuso horário UTC
UTC_TIMESTAMP()	Retorna a data e hora atuais (aaaa- mm-dd hh:mm:ss) fuso horário UTC



# Exemplos

Data atual: 27/10/2009 21:23:58

- `SELECT NOW();`
  - 2009-10-27 21:23:58
  
- `SELECT CURDATE();`
  - 2009-10-27
  
- `SELECT UTC_TIMESTAMP();`
  - 2009-10-27 23:32:10

# Funções para obter partes da data

Função	Retorno
DAY(data_qualquer)	Retorna o dia de uma data
MONTH(data_qualquer)	Retorna o mês de uma data
YEAR(data_qualquer)	Retorna o ano de uma data
SECOND(data/hora_qualquer)	Retorna os segundos de uma data/hora
MINUTE(data/hora_qualquer)	Retorna os minutos de uma data/hora
HOUR(data/hora_qualquer)	Retorna a hora de uma data/hora

# Exemplos

- `SELECT DAY(CURDATE());` Data atual: 27/10/2009 21:23:58
  - 27
- `SELECT MONTH(CURDATE());`
  - 10
- `SELECT YEAR(CURDATE());`
  - 2009

# Operações com datas

Função	Retorno
ADDDATE(data, intervalo) ou DATE_ADD(data,intervalo)	Retorna data resultante da adição de intervalo a data
DATEDIFF(data_inicial, data_final)	retorna o número de dias entre a data inicial e a data final

# Exemplos

Data atual: 27/10/2009 21:23:58

- `SELECT ADDDATE(now (), interval 24 day)`
  - 2009-11-20 22:33:42
  
- `SELECT DATEDIFF(NOW(), '2009-12-21')`
  - 65

# Formatos de data

## ■ DATE\_FORMAT(data,formato)

- Formata o valor de data de acordo com a string formato string. Os seguintes identificadores podem ser utilizados:

Specifier	Description
%M	Nome do mês (January..December)
%m	Mês, numerico (00..12)
%D	Dia do mês com sufixo Inglês (0th, 1st, 2nd, 3rd, etc.)
%Y	Ano, numerico, 4 digitos
%y	Ano, numerico, 2 digitos
%a	Nome da semana abreviado (Sun..Sat)
%d	Dia do mês, numerico (00..31)
%e	Dia do mês, numerico (0..31)
%c	Mês, numerico (0..12)
%b	Nome do mês abreviado (Jan..Dec)

# Exemplos

- `Select date_format('2009-10-27', '%d/%m/%y');`
  - 27/10/09
- `Select date_format('2009-12-31','%M %D of %Y');`
  - December 31st of 2009