Universidade **Tecnológica**
Federal do Paraná
UTFPR - Cornélio Procópio

Sistemas de Banco de Dados

Professor: André L. S. Domingues
anddomingues@utfpr.edu.br

Especialização em Tecnologia Java
Novembro de 2011

Bibliografia

- ELMASRI, Ramez E.; NAVATHE, Shamkant; Sistemas de Bancos de Dados. São Paulo: Addison Wesley, 2005. 4ª ed.
- SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S.; Sistema de Banco de Dados. Rio de Janeiro: Campus, 2006. Tradução da 5ª edição.
- DATE C.J.; Introdução a Sistemas de Bancos de Dados. Rio de Janeiro: Campus, 2004. Tradução da 8ª edição americana.

Bibliografia

- ÖZSU, M. Tamer; VALDURIEZ, Patrick; Princípios de Sistemas de Bancos de Dados Distribuídos. Rio de Janeiro: Campus, 2001. Tradução da 2ª edição americana.

- Graves, Mark; Projeto de Banco de Dados com XML. Pearson Education do Brasil, 2003.

Sites

- **www.mysql.com** - Site oficial MySQL.
- **www.postgresql.org[.br]** - Site oficial PostgreSQL.
- **www.informatik.uni-trier.de/~ley/db/conf/sbbd/index.html** - Index do SBBD.
- **www.w3schools.com/SQL** - Tutoriais básicos oficiais da W3C, bastante didáticos.

Avaliação

- Exercícios
  - Trabalhos práticos para serem entregues
  - Tempo no final da aula para dúvidas / resolução
  - Em duplas

Média dos trabalhos = nota final da disciplina

Conceitos BD

O que você entende por

de

???

"Conjunto de dados integrados que tem por objetivo atender a uma comunidade específica"

Heuser

Conceitos BD

"Um *Banco de Dados* é uma coleção estruturada de dados relacionados a alguns fenômenos reais que estamos tentando modelar."

Ozsü

"Um *Banco de Dados* é uma coleção de dados relacionados, organizada e armazenada de forma a possibilitar fácil manipulação."

Elmasri

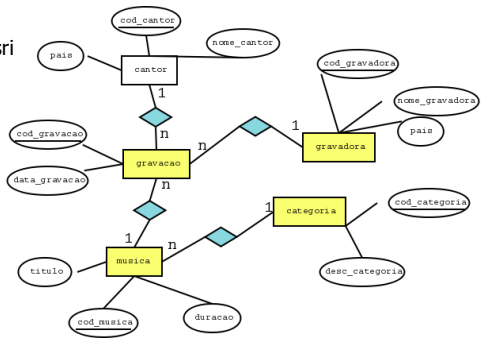
Modelos de Dados

SBDs são construídos baseados em ***Modelos de Dados***

- Modelos Conceituais: Representam a realidade com alto nível de abstração. Ex.: Modelo ER
- Modelos Lógicos: Descrição dos dados da forma como serão processados. Ex.: Modelo Relacional
- Modelos Físicos: Descrevem como os dados são armazenados fisicamente

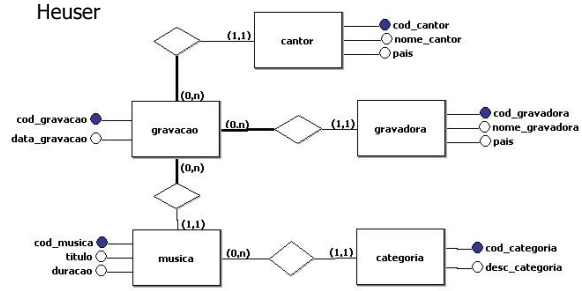
Exemplo de Modelo Conceitual (ER)

Elmasri

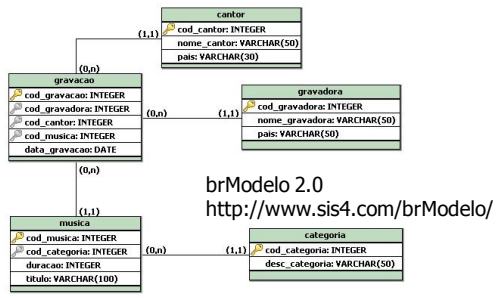


Exemplo de Modelo Conceitual (ER)

Heuser

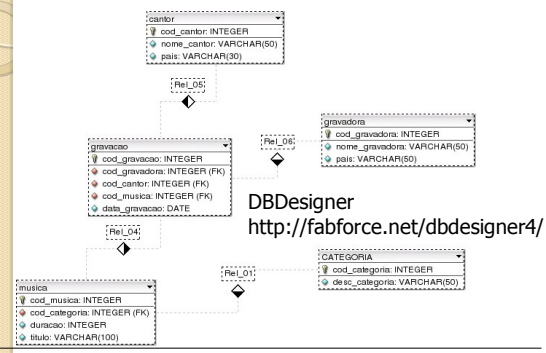


Exemplo de Modelo Relacional



brModelo 2.0
http://www.sis4.com/brModelo/

Exemplo de Modelo Relacional



DBDesigner
http://fabforce.net/dbdesigner4/

Modelo Físico

- SQL *(Structured Query Language)*

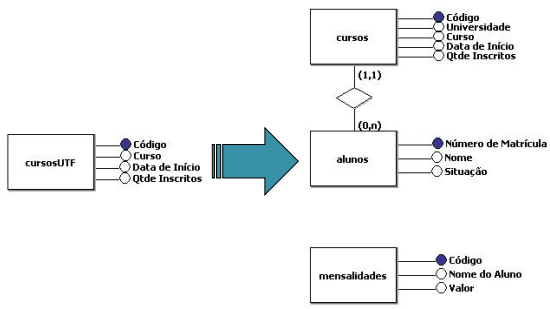
Sintaxe:

- Não diferencia maiúsculas / minúsculas
- Identificadores: sem espaços, cedilha, acentos
- Ponto para separar casas decimais
- %: substring
- ||: concatenação

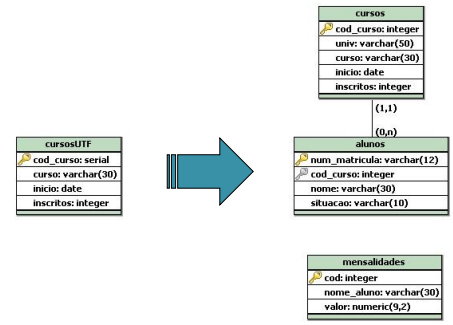
SQL

- DDL *(Data Definition Language)*
  - *Linguagem de definição de dados*
  - *Define a estrutura (esquema) de um BD*
  - Metadados: Criação e Alteração de Tabelas
    - CREATE TABLE, ALTER TABLE, DROP TABLE
  - Restrições (null, unique, PK, FK)
    - Integridade
  - Schemas, Tablespaces: Subdivisões do BD no PostgreSQL
    - CREATE SCHEMA, CREATE TABLESPACE

Exemplo para Modelo Físico



Exemplo para Modelo Físico



SQL/DDL

```
CREATE TABLE cursosUTF(
      cod_curso serial PRIMARY KEY, -- Já tem DEFAULT
      univ varchar(50) NOT NULL,
      curso varchar(30) DEFAULT 'Java',
      inicio date DEFAULT '2007-01-01',
      inscritos integer DEFAULT 0
);

CREATE TABLE cursos(
      cod_curso INTEGER,
      univ varchar(50) NOT NULL,
      curso varchar(30),
      inicio date DEFAULT '2009-01-01',
      inscritos integer DEFAULT 0,
      CONSTRAINT cursos_pk PRIMARY KEY (cod_curso)
);
```

RESTRIÇÕES DIRETAMENTE NA TABELA

RESTRIÇÃO NA FORMA DE *CONSTRAINT*, FACILITA LOCALIZAÇÃO EM CASO DE ERRO

SQL/DDL

```
CREATE TABLE alunos(
      num_matricula VARCHAR(12),
      cod_curso INTEGER,
      nome VARCHAR(30),
      situacao VARCHAR(10) DEFAULT 'Regular',
      CONSTRAINT alunos_pk PRIMARY KEY (num_matricula)
);

ALTER TABLE alunos
 ADD CONSTRAINT alunos_cod_curso_fk FOREIGN KEY
 (cod_curso) REFERENCES cursos;

ALTER TABLE cursosUTF
 ALTER COLUMN univ SET DEFAULT 'UTFPR';

--DROP TABLE alunos CASCADE;
```

NO *ALTER*: EVITA PROBLEMAS NA ORDEM DE CRIAÇÃO

REMOVE TABELA MESMO COM DEPENDÊNCIA

SQL/DDL

```
CREATE TABLE mensalidades(
      cod INTEGER,
      nome_aluno VARCHAR(30),
      valor NUMERIC(9,2)
);

ALTER TABLE mensalidades
 ADD CONSTRAINT mensalidade_cod_pk PRIMARY KEY(cod);
```


SQL/DML

- DML (*Data Manipulation Language*)
  - *Permite o acesso ou manipulação dos dados no BD*
  - *Recuperação, inserção, exclusão e modificação dos dados*
  - INSERT – Inserção de dados
  - UPDATE – Atualização de dados
  - DELETE – Exclusão
  - SELECT – Consulta / Recuperação

SQL/DML - Insert

```
-- Com valores default
INSERT INTO cursosUTF
  VALUES (DEFAULT,DEFAULT,'Pedagogia','2007-08-01',
  DEFAULT);
-- Apenas valores DEFAULT
INSERT INTO cursosUTF DEFAULT VALUES;
-- Omitindo valores
INSERT INTO cursosUTF(curso,inicio,inscritos)
  VALUES ('Redes','2010-08-01',5);
--Inserção com consulta
INSERT INTO cursos
  SELECT * FROM cursosUTF WHERE inicio < CURRENT_DATE;
-- Todos valores
INSERT INTO cursos VALUES (3,'UEL','Web','2009-03-05',25);
INSERT INTO alunos VALUES('11596',2,'Reinaldo Costa',
  DEFAULT);
```


SQL/DML - Update

```
-- UPDATE padrão
UPDATE cursosUTF SET inscritos = 0 WHERE inscritos <
  10;
UPDATE cursos SET inicio = DEFAULT WHERE univ = 'UTFPR';

-- Usando valor DEFAULT
UPDATE cursos SET inicio = '2009-03-06', inscritos =
  DEFAULT WHERE univ = 'UTFPR' AND curso = 'Java';
```


SQL/DML - Delete

```
PARA TESTAR:
INSERT INTO alunos VALUES('11597',2,'João Silva','Inativo');
INSERT INTO alunos VALUES('11598',1,'Paulo Lima','Regular');

-- Com WHERE – Obs.:WHERE não segue padrão SQL ANSI
DELETE FROM alunos WHERE situacao = 'Inativo';

-- Com USING – Obs.: USING não segue padrão SQL ANSI
DELETE FROM alunos USING cursos
WHERE alunos.cod_curso=cursos.cod_curso AND cursos.curso='Java';

-- Sem USING (mesmo da anterior)
DELETE FROM alunos WHERE cod_curso IN
  (SELECT cod_curso FROM cursos WHERE curso = 'Java');

-- Removendo todos os registros
DELETE FROM alunos;
```


SQL/DML - Select

```
PARA TESTAR:
  INSERT INTO cursos
   VALUES(4,'UTFPR','Automação','2009-07-15',22);
  INSERT INTO cursos
   VALUES(5,'UTFPR','Gestão','2008-03-01',18);
-- Retorna todos os registros da tabela cursos
SELECT * FROM cursos;
SELECT cod_curso,univ,curso,inicio,inscritos FROM cursos;
-- "arrecadacao" aparece como coluna no resultado
SELECT univ,curso,(inscritos*300) AS arrecadacao FROM cursos;
-- Qualificando com WHERE
SELECT * FROM cursos WHERE univ <> 'UTFPR' AND inscritos > 15;
-- Ordenando
SELECT * FROM cursos
  WHERE univ = 'UTFPR' AND inscritos > 15 ORDER BY inscritos;
-- Removendo dados duplicados
SELECT DISTINCT univ FROM cursos ORDER BY univ;
```

Select – Funções Agregadas

- Computam um único resultado para vários registros de entrada.
  - **MAX**: Valor máximo
  - **MIN**: Valor mínimo
  - **COUNT**: Quantos registros satisfazem um determinado critério
  - **SUM**: Somatório (somente para valores numéricos)
  - **AVG**: Média artimética (somente para valores numéricos)

Select – Funções Agregadas

- Exemplo: Selecionar curso com maior número de inscritos

```
SELECT univ,curso
FROM cursos
WHERE inscritos = max(inscritos);
-- Errado ! Cláusula WHERE é processada antes da função
```

```
SELECT univ,curso
FROM cursos
WHERE inscritos =
    (SELECT max(inscritos) FROM cursos);
-- Correto
```


Select – Funções Agregadas

- Agregações podem ser usadas em conjunto com GROUP BY

```
SELECT univ,sum(inscritos)
FROM cursos
GROUP BY univ;
-- Uma linha de saída para cada Universidade

SELECT univ,sum(inscritos)
FROM cursos
GROUP BY univ
HAVING sum(inscritos) > 30;
-- Resultado filtrado com HAVING
```


Select – Subconsultas

- Consultas aninhadas
  - Cláusula EXISTS: verifica se um valor existe no resultado da subconsulta.

```
SELECT col1 FROM tab1 WHERE EXISTS(SELECT 1 FROM tab2 WHERE col2 = tab1.col2);
-- O "1" é utilizado pois não há interesse no conteúdo retornado, só há interesse se houve retorno ou não.
-- Se a subconsulta retornar algo, o resultado é TRUE, caso contrário é FALSE.
-- Há ligação entre a consulta externa e a interna
```


Select – Subconsultas

  - IN e NOT IN: verifica se um valor está contido em um grupo de valores.

```
Para testar:
INSERT INTO alunos VALUES('11596',2,'Reinaldo Costa',DEFAULT);
INSERT INTO alunos VALUES('11597',2,'João Silva','Inativo');
INSERT INTO alunos VALUES('11598',1,'Paulo Lima','Regular');
INSERT INTO mensalidades VALUES(1,'Paulo Lima',300.00);
INSERT INTO mensalidades VALUES(2,'Paulo Lima',300.00);
INSERT INTO mensalidades VALUES(3,'João Silva',290.00);
SELECT DISTINCT nome FROM alunos
  WHERE nome NOT IN (SELECT nome_aluno FROM mensalidades);
SELECT DISTINCT nome_aluno FROM mensalidades
  WHERE valor IN ('300.00','250.00');
SELECT * FROM cursos
  WHERE cod_curso IN (SELECT cod_curso FROM alunos);
```


Select – Join (Junções)

- INNER JOIN: Somente as linhas/registros que satisfaçam a ligação determinada pelo JOIN serão recuperados pelo SELECT, sendo assim, os registros que não se enquadram no relacionamento definido pelo JOIN não serão recuperados.

```
Exemplo:
    SELECT alunos.num_matricula,alunos.nome,
    mensalidades.cod,mensalidades.valor
    FROM ALUNOS INNER JOIN MENSALIDADES
    ON alunos.nome=mensalidades.nome_aluno
    ORDER BY alunos.nome;
```

Select – Join (Junções)

cursos	
cod_curso	curso
1	Pedagogia
2	Java
3	Web

alunos	
cod_curso	nome
1	Paulo Lima
2	Reinaldo Costa
2	João Silva

```
SELECT cursos.cod_curso,cursos.curso,alunos.nome
FROM cursos INNER JOIN alunos
ON cursos.cod_curso = alunos.cod_curso;
```

cod_curso	curso	nome
1	Pedagogia	Paulo Lima
2	Java	Reinaldo Costa
2	Java	João Silva

Condições de Junção:

- ON: mais geral
- USING: especifica os nomes das colunas que devem ser comparadas. USING (a, b, c) é equivalente a ON (t1.a = t2.a AND t1.b = t2.b AND t1.c = t2.c)
- NATURAL: quando os nomes de colunas nas duas tabelas são iguais

Select – Join (Junções)

cursos	
cod_curso	curso
1	Pedagogia
2	Java
3	Web

alunos	
cod_curso	nome
1	Paulo Lima
2	Reinaldo Costa
2	João Silva

```
SELECT cursos.cod_curso,cursos.curso,alunos.nome
FROM cursos INNER JOIN alunos
USING (cod_curso);
```

cod_curso	curso	nome
1	Pedagogia	Paulo Lima
2	Java	Reinaldo Costa
2	Java	João Silva

Condições de Junção:

- ON: mais geral
- USING: especifica os nomes das colunas que devem ser comparadas. USING (a, b, c) é equivalente a ON (t1.a = t2.a AND t1.b = t2.b AND t1.c = t2.c)
- NATURAL: quando os nomes de colunas nas duas tabelas são iguais

Select – Join (Junções)

cursos	
cod_curso	curso
1	Pedagogia
2	Java
3	Web

alunos	
cod_curso	nome
1	Paulo Lima
2	Reinaldo Costa
2	João Silva

```
SELECT cursos.cod_curso,cursos.curso,alunos.nome
FROM cursos NATURAL INNER JOIN alunos;
```

cod_curso	curso	nome
1	Pedagogia	Paulo Lima
2	Java	Reinaldo Costa
2	Java	João Silva

Condições de Junção:

- ON: mais geral
- USING: especifica os nomes das colunas que devem ser comparadas. USING (a, b, c) é equivalente a ON (t1.a = t2.a AND t1.b = t2.b AND t1.c = t2.c)
- NATURAL: quando os nomes de colunas nas duas tabelas são iguais

Select – Join (Junções)

- LEFT JOIN: Através do uso do LEFT, todos os registros na tabela à esquerda da consulta serão listados, independente de terem ou não registros relacionados na tabela à direita. Nesse caso, as colunas relacionadas com a tabela da direita retornam nulos (NULL).

```
SELECT alunos.num_matricula,alunos.nome,
mensalidades.cod,mensalidades.valor
FROM ALUNOS LEFT JOIN MENSALIDADES
ON alunos.nome=mensalidades.nome_aluno
ORDER BY alunos.nome;
```

- No exemplo, todos os registros da tabela ALUNOS serão listados, independente de terem ou não registros associados na tabela MENSALIDADES. Caso não existam registros associados na tabela MENSALIDADES, os campos mensalidades.cod e mensalidades.valor retornarão NULL.

Select – Join (Junções)

cursos	
cod_curso	curso
1	Pedagogia
2	Java
3	Web

alunos	
cod_curso	nome
1	Paulo Lima
2	Reinaldo Costa
2	João Silva

```
SELECT cursos.cod_curso,cursos.curso,alunos.nome
FROM cursos LEFT JOIN alunos
ON cursos.cod_curso = alunos.cod_curso
ORDER BY cod_curso;
```

cod_curso	curso	nome
1	Pedagogia	Paulo Lima
2	Java	Reinaldo Costa
2	Java	João Silva
3	Web	

Select – Join (Junções)

cursos	
cod_curso	curso
1	Pedagogia
2	Java
3	Web

alunos	
cod_curso	nome
1	Paulo Lima
2	Reinaldo Costa
2	João Silva

```
SELECT cursos.cod_curso,cursos.curso,alunos.nome
FROM cursos LEFT JOIN alunos
USING (cod_curso)
ORDER BY cod_curso;
```

cod_curso	curso	nome
1	Pedagogia	Paulo Lima
2	Java	Reinaldo Costa
2	Java	João Silva
3	Web	

Select – Join (Junções)

cursos	
cod_curso	curso
1	Pedagogia
2	Java
3	Web

alunos	
cod_curso	nome
1	Paulo Lima
2	Reinaldo Costa
2	João Silva

```
SELECT cursos.cod_curso,cursos.curso,alunos.nome
FROM cursos LEFT JOIN alunos
ON cursos.cod_curso=alunos.cod_curso AND cursos.curso='Java'
ORDER BY cod_curso;
```

cod_curso	curso	nome
1	Pedagogia	
2	Java	Reinaldo Costa
2	Java	João Silva
3	Web	

Select – Join (Junções)

- RIGHT JOIN: É o inverso do Left Join, ou seja, todos os registros da tabela à direita serão listados, independente de terem ou não registros relacionados na tabela à esquerda).

```
Para testar:
    INSERT INTO mensalidades
          VALUES(4,'Luciana Pereira',310.00);

    SELECT alunos.num_matricula,alunos.nome,
    mensalidades.cod,mensalidades.valor
    FROM ALUNOS RIGHT JOIN MENSALIDADES
    ON alunos.nome=mensalidades.nome_aluno;
```

- Todos os registros da tabela MENSALIDADES serão listados, e caso não haja correspondentes na tabela ALUNOS, a consulta retorna NULL para os campos alunos.num_matricula e alunos.nome.

Select – Join (Junções)

alunos	
num_matricula	nome
11596	Reinaldo Costa
11597	João Silva
11598	Paulo Lima

mensalidades	
nome_aluno	valor
Paulo Lima	300.00
Paulo Lima	300.00
João Silva	290.00
Luciana Pereira	310.00

```
SELECT alunos.num_matricula,mensalidades.nome_aluno,
mensalidades.valor
FROM alunos RIGHT JOIN mensalidades
ON alunos.nome = mensalidades.nome_aluno;
```

num_matricula	nome	valor
11598	Paulo Lima	300.00
11598	Paulo Lima	300.00
11597	João Silva	290.00
	Luciana Pereira	310.00

Select – Join (Junções)

- FULL OUTER JOIN: INNER JOIN + LEFT JOIN + RIGHT JOIN

alunos	
num_matricula	nome
11596	Reinaldo Costa
11597	João Silva
11598	Paulo Lima

mensalidades	
nome_aluno	valor
João Silva	290.00
Luciana Pereira	310.00
Paulo Lima	300.00
Paulo Lima	300.00

```
SELECT alunos.num_matricula,alunos.nome,
mensalidades.nome_aluno, mensalidades.valor
FROM alunos FULL JOIN mensalidades
ON alunos.nome = mensalidades.nome_aluno;
```

num_matricula	nome	nome_aluno	valor
11596	Reinaldo Costa		
11597	João Silva	João Silva	290.00
		Luciana Pereira	310.00
11598	Paulo Lima	Paulo Lima	300.00
11598	Paulo Lima	Paulo Lima	300.00

Outras Funções - Data

- CURRENT DATE: Retorna a data atual

```
SELECT * FROM cursos WHERE inicio > CURRENT_DATE;
```

- AGE:
  - Com duas datas: diferença entre as datas informadas
  - Com uma data: diferença entre a data informada e a data atual

```
SELECT age('2010-09-15',cursos.inicio) as duracao_total
FROM cursos
WHERE cursos.cod_curso = '2';

SELECT age(cursos.inicio) as prazo_decorrido
FROM cursos
WHERE cursos.cod_curso = '2';
```


Outras Funções - Data

- DATE_PART: Retorna a parte selecionada de uma data

```
SELECT cursos.univ,cursos.curso,
date_part('day',cursos.inicio)||'/'
||date_part('month',cursos.inicio) as data_inicio_cursos2009
FROM cursos WHERE date_part('year',cursos.inicio) = 2009;
```

- TO_CHAR: Converte data para texto usando máscara

```
SELECT cursos.univ,cursos.curso,
to_char(cursos.inicio,'DD/MM') as data_inicio_cursos2009
FROM cursos
WHERE to_char(cursos.inicio,'YYYY')='2009';
```

Outras Funções - Data

- TO_DATE: Converte string para data

```
UPDATE cursos SET inicio=to_date('01/03/2009','dd/mm/yyyy')
WHERE cod_curso='1';
```

  - Obs.: O último exemplo (update) poderia ser executado diretamente, sem a função to_date. As funções to_char e to_date são úteis para adequar o formato de entrada da aplicação para o formato do Banco de Dados.

EXEMPLOS

```
--Script (preparação):
DELETE FROM alunos; DELETE FROM cursos;
INSERT INTO cursos VALUES(1,'UEL','Web','2009-01-01',20);
INSERT INTO cursos VALUES(2,'UTFPR','Java','2009-03-01',25);
INSERT INTO cursos VALUES(3,'UTFPR','Java','2010-03-06',18);
INSERT INTO cursos VALUES(4,'UEL','Web','2007-02-01',22);
INSERT INTO cursos VALUES(5,'UFPR','Redes','2007-03-10',30);
INSERT INTO cursos VALUES(6,'UFPR','Redes','2009-07-01',25);
INSERT INTO alunos VALUES('156111',1,'João Lima','Inativo');
INSERT INTO alunos VALUES('156123',3,'Eduardo Marques','Regular');
INSERT INTO alunos VALUES('156124',3,'Lauro Rodrigues','Regular');
INSERT INTO alunos VALUES('156128',3,'Maria Soares','Regular');
INSERT INTO alunos VALUES('156131',4,'Heloisa Pires','Inativo');
INSERT INTO alunos VALUES('156133',4,'Ana Pereira','Regular');
INSERT INTO alunos VALUES('156134',5,'Mario Borges','Inativo');
INSERT INTO alunos VALUES('156137',6,'Beatriz Nunes','Regular');
INSERT INTO alunos VALUES('156138',5,'Luciana Vieira','Inativo');
INSERT INTO alunos VALUES('157123',5,'José Cardoso','Regular');
INSERT INTO alunos VALUES('157134',1,'Catarina Brito','Regular');
INSERT INTO alunos VALUES('157135',2,'Denise Braga','Regular');
```


EXEMPLOS

Faça consultas que retornem:

1 - Nome do curso (e Universidade) que tem o maior número de alunos matriculados (mostrar também quantidade de alunos matriculados).

Resposta esperada:

universidade	curso	alunos_matriculados
UTFPR	Java	3
UFPR	Redes	3

EXEMPLOS

2 - Curso / Universidade que tem mais alunos inativos (com a quantidade de alunos inativos).

Resposta esperada:

universidade	curso	alunos_inativos
UFPR	Redes	2

EXEMPLOS

3 - Nome do curso (com Universidade) e dos alunos matriculados nas turmas com início em 2007.

Resposta esperada:

universidade	curso	alunos_matriculados
UEL	Web	Heloisa Pires
UEL	Web	Ana Pereira
UFPR	Redes	Mario Borges
UFPR	Redes	Luciana Vieira
UFPR	Redes	José Cardoso

EXEMPLOS

```
--Resolução Exemplo 1:

SELECT cursos.univ as universidade, cursos.curso,
  count(alunos.num_matricula) as alunos_matriculados
FROM cursos,alunos
WHERE cursos.cod_curso = alunos.cod_curso -- Pode usar JOIN
GROUP BY cursos.cod_curso,cursos.univ,cursos.curso
HAVING count(alunos.num_matricula)=
(SELECT max(matriculas.total)
FROM(SELECT count(num_matricula) as total
FROM alunos
GROUP BY cod_curso) as matriculas);
```

EXEMPLOS

```
--Resolução Exemplo 2:

SELECT cursos.univ as universidade,
  cursos.curso,count(alunos.num_matricula) as alunos_inativos
FROM cursos,alunos
WHERE cursos.cod_curso = alunos.cod_curso AND
  alunos.situacao='Inativo'
GROUP BY cursos.cod_curso,cursos.univ,cursos.curso
HAVING count(alunos.num_matricula)=
(SELECT max(inativos.total)
FROM (SELECT count(alunos.num_matricula) as total
FROM alunos
WHERE alunos.situacao = 'Inativo'
GROUP BY alunos.cod_curso) as inativos);
```


EXEMPLOS

```
--Resolução Exemplo 3:

SELECT cursos.univ as universidade, cursos.curso,
  alunos.nome as alunos_matriculados
FROM cursos NATURAL INNER JOIN alunos
WHERE date_part('YEAR',cursos.inicio)=2007;
```