



Escola Politécnica



PMR - Depto. de Enga. Mecatrônica

* PMR 2490-Laboratório

Prof. José Reinaldo Silva

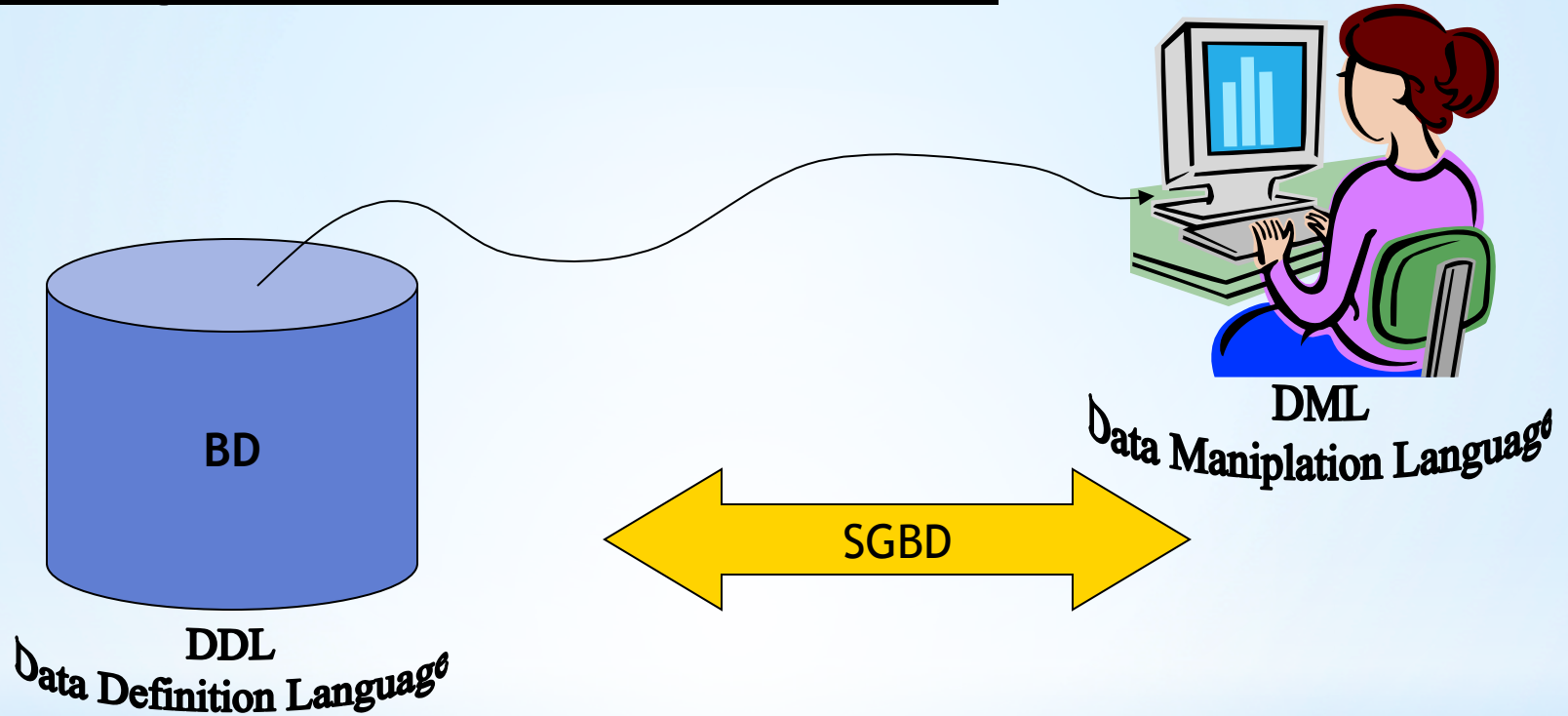
Cada turma tem um acesso diferente nas máquinas do LAB:

Console console epusp	VM clicar em ReqVMw PMR2490	User: aluno pmr2490	MySQLuser aluno	Passwd pmr2490
-----------------------------	--------------------------------------	------------------------	--------------------	-------------------

*
O
comput



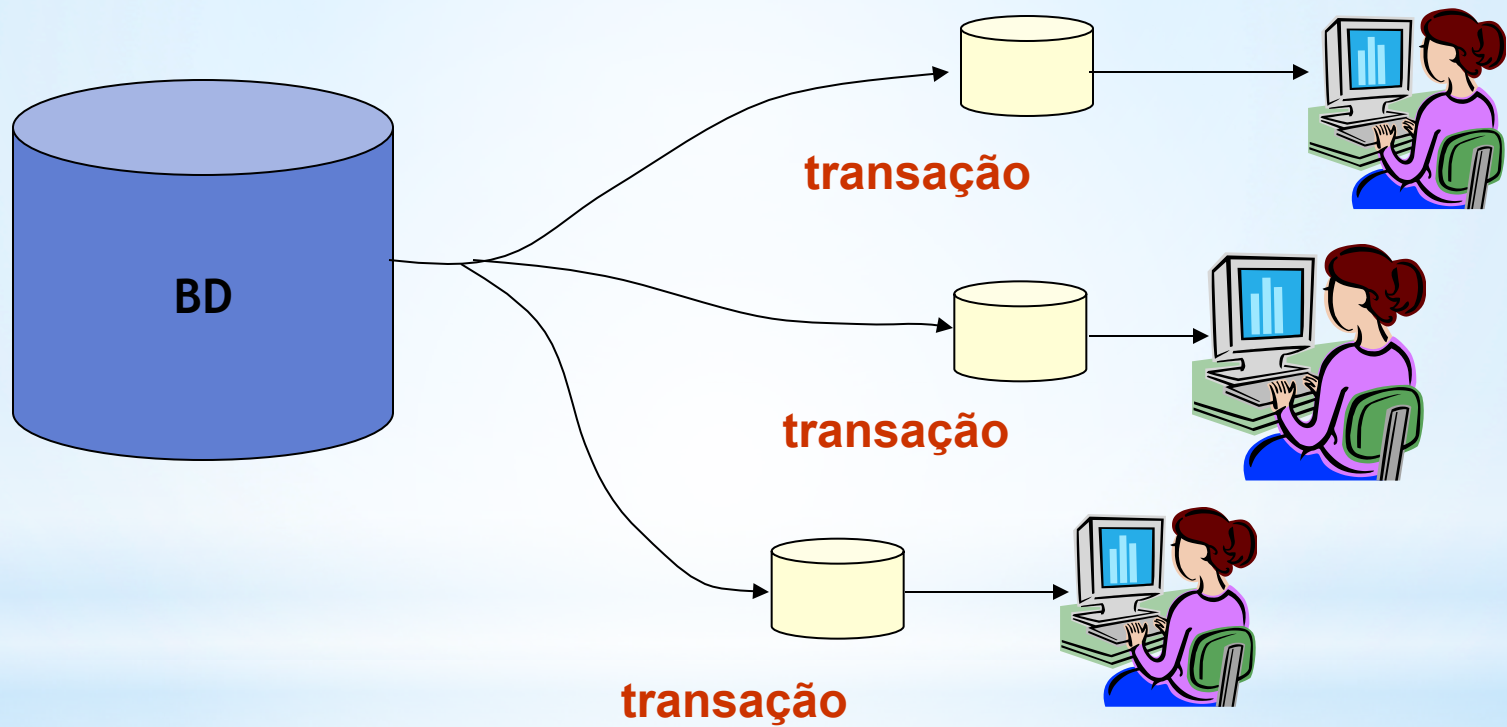
Introdução dos Bancos de Dados



Sistema de Gerenciamento de Banco de Dados

Problemas com o sistema de arquivos indexados

- * Redundância : a mesma informação armazenada com valores diferentes em diferentes lugares
- * Impacto das mudanças: uma mudança na informação armazenada em diferentes lugares pode ser feito em apenas algumas das localidades
- * Ineficiência para proceder mudanças e adaptações
- * Dificuldade de acesso: o acesso aos dados pode ser difícil para o usuário final, causando uma excessiva dependência dos programadores.



Transaction Manager

Garante as propriedades ACID

- Atomicidade (Atomicity): uma transação é indivisível; ou ela acontece ou não
- Consistência (Consistency): Antes e após uma transação, o estado do banco de dados é consistente
- Isolamento (Isolation): Se duas ou mais transações acontecem simultaneamente, seus efeitos devem ser isolados
- Durabilidade (Durability): Se uma transação se completa, seus efeitos não devem cessar mesmo que o sistema falhe imediatamente após

Transaction Manager

- Lock
 - Define o nível de isolamento
 - Página
 - Linha
- Logging
 - Armazenamento não volátil

- Commit
 - Durabilidade da transção

Exercício

Vamos construir um script simples composto de duas tabelas que denotam duas entidades: uma representando um departamento de empresa e a outra os seus empregados.

departamento

empregado

Exemplo

```
CREATE TABLE departamento
(
    id          INT,
    nome        VARCHAR(25),
    region      VARCHAR(10)
);
```

```
CREATE TABLE empregado
(
    id          INT,
    nome        VARCHAR(25),
    position    VARCHAR(25)
);
```

Seria possível colocar todos os comandos em um arquivo “plain text” (.txt) e rodar todo o arquivo com o comando **scripting** ou na linha de comando como mostrado a seguir:

Prompt> **mysql** *db_name file_name*

Rodando Scripts

The screenshot displays the MySQL Workbench interface. The main window shows a query editor with two SQL scripts. The first script creates a table named 'dep' with columns 'id' (INT), 'name' (VARCHAR(25)), and 'region' (VARCHAR(10)). The second script creates a table named 'emp' with columns 'id' (INT), 'name' (VARCHAR(25)), and 'salary' (DECIMAL(8,2)). The interface includes a Navigator on the left, a menu bar, and a toolbar. The bottom panel shows the 'Output' section with 'Action Output' selected, and a status bar at the bottom indicating 'Context Help' and 'Snippets'.

MySQL Workbench

pmr2490 x

File Edit View Query Database Server Tools Scripting Help

Navigator Query 1 scriptTest x

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

- sakila
- sys
- world
 - Tables
 - Views
 - Stored Procedures

Information

Schema: world

```
1 • create table dep
2   (
3     id      INT,
4     name    VARCHAR(25),
5     region  VARCHAR(10)
6   );
7 • create table emp
8   (
9     id      INT,
10    name    VARCHAR(25),
11    salary  DECIMAL(8,2)
12  );
```

SQL Additions

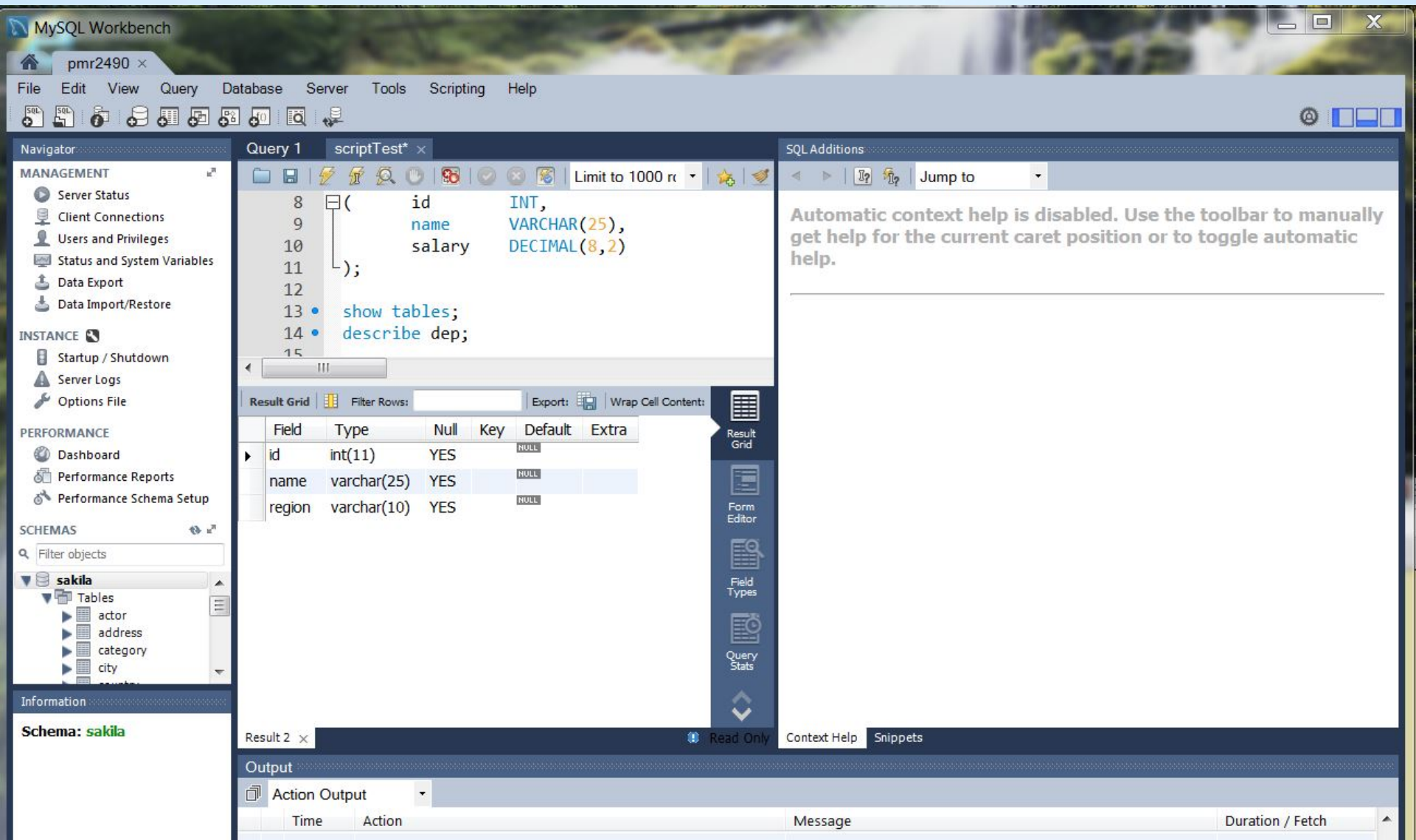
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Output

Action Output

Time	Action	Message	Duration / Fetch
------	--------	---------	------------------

Context Help Snippets



ALTER TABLE

ALTER TABLE <nome> ADD <def_coluna>

ALTER TABLE <nome> DROP <coluna>

É de fundamental importância que não existam tuplas iguais. Portanto cada uma delas deve ser identificada univocamente. A função do número de fornecedor (Sno) e do número de série da peça (Pno) é justamente esta.

Para enfatizar isto em um banco de dados se define estes campos como chave primária e neste caso estes não podem ter valor NULL.

 **Chave primária**

```
ALTER TABLE dept  
MODIFY id INT NOT NULL PRIMARY KEY;
```

FAÇA O MESMO COM O id DA TABELA DE EMPREGADOS

* Introduzindo chave
primária

MySQL Workbench

pmr2490 x

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

sakila

- Tables
 - actor
 - address
 - category
 - city

Information

Schema: sakila

Query 1 scriptTest* x

```

12
13 • show tables;
14 • describe dep;
15 • alter table dep
16 ❌ modify id INT NOT NULL primary key;
17 • describe dep;
18
  
```

Limit to 1000 r

Result Grid

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	
name	varchar(25)	YES		NULL	
region	varchar(10)	YES		NULL	

SQL Additions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid

Form Editor

Field Types

Query Stats

Result 3 x

Read Only Context Help Snippets

Output

Action Output

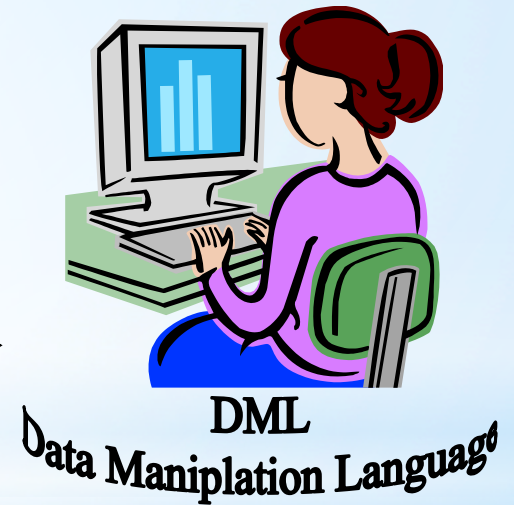
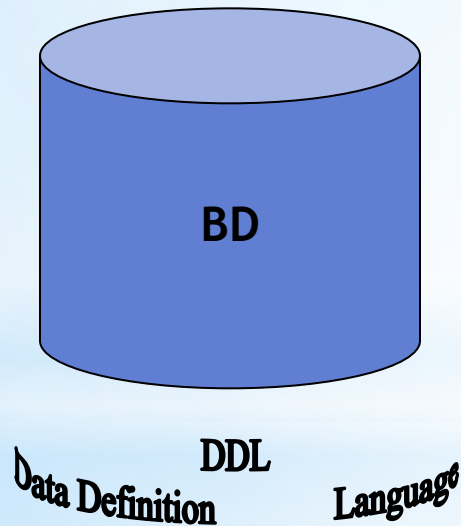
Time	Action	Message	Duration / Fetch
------	--------	---------	------------------

```
ALTER TABLE emp  
MODIFY id INT NOT NULL PRIMARY KEY;
```


Laboratório – Parte 3

DML – Data Manipulation Language

Tutorial, exercicios



Uma vez definido o esquema (pela DDL) as operações de manipulação são :

- Inserir dados nas tabelas
- modificar dados já inseridos
- apagar dados existentes para manutenção do BD (operação perigosa)

No caso do DML (Data Manipulation Language) vamos trabalhar com poucos comandos para:

- Selecionar dados na tabela para suprir processos ou ações sobre o BD;
- Inserir dados ou modificar os dados já inseridos.

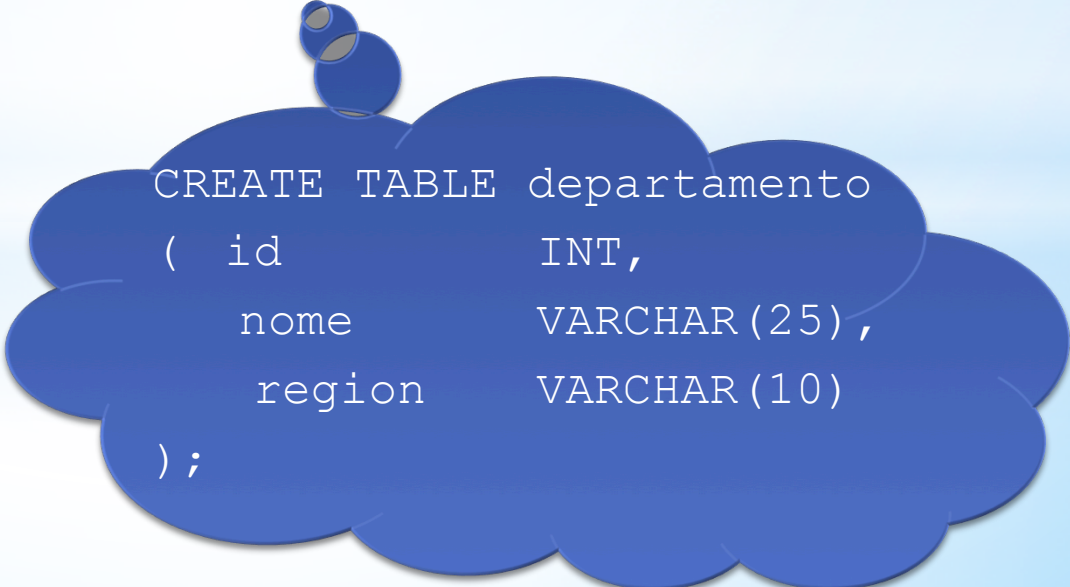
INSERT INTO

```
INSERT INTO <nome>  
  { (lista_colunas) }  
  VALUES ( <lista_valores> )
```

Exemplo

- Insere o Departamento de Compras

```
INSERT INTO departamento  
VALUES ( 1, 'Compras' , 'Leste' );
```



```
CREATE TABLE departamento  
( id          INT,  
  nome        VARCHAR(25),  
  region      VARCHAR(10)  
);
```

A linguagem de manipulação fala de um banco de dados cujo esquema já foi definido. Com ela se pode *popular* o banco com dados, procurar dados, alterar o que já foi armazenado, etc.

Todo comando da DML deve respeitar a estrutura lógica (o esquema) anteriormente definida (usando a DDL).

Exemplo

- Insere o Departamento Pessoal e o Financeiro

```
INSERT INTO departamento  
VALUES ( 2, 'Pessoal' , 'Oeste' );
```

```
INSERT INTO departamento  
VALUES ( 3, 'Financeiro' , 'Sul' );
```

Restrição

seleciona um subconjunto de linhas da tabela

Projeção

seleciona um conjunto de colunas

SUPPLIER:

SNO | SNAME | CITY

SNO	SNAME	CITY
1	Smith	London
2	Jones	Paris
3	Adams	Vienna
4	Blake	Rome

PART:

PNO | PNAME | PRICE

PNO	PNAME	PRICE
1	Screw	10
2	Nut	8
3	Bolt	15
4	Cam	25

```
SELECT sno, sname  
FROM supplier  
WHERE city= 'Paris' OR city= 'Rome' ;
```

SNO e SNAME de
fornecedor situado
em cidade onde se
fala lingua latina

```
SELECT *  
FROM part  
WHERE price > 10
```

Tuplas de peças cujo
preço é maior que 10

SELECT

```
SELECT { ALL | DISTINCT } <lista_itens>  
FROM <lista_tabelas>  
WHERE <expressão_seleção>
```

Exemplo

- Verifica os dados da tabela de departamentos

```
SELECT *
```

```
FROM departamento;
```

(só tem sentido fazer isso porque temos até aqui uma tabela com poucos dados)

Vamos agora enriquecer o nosso design do BD

- Primeiro insira no diagrama E-R anterior um relacionamento trabalha_em entre empregado e departamento (quais os atributos deste relacionamento?)
- em seguida insira o relacionamento gerente_de entre empregado e departamento de modo que cada departamento tem somente um gerente
- agora defina a entidade produto
- cada departamento é responsável pela venda de um conjunto de produtos e portanto existe o **relacionamento vende** entre departamento e produto
- mas nada disso tem sentido sem a entidade cliente que é atendido por um departamento em um ou mais produtos. Como será este relacionamento?

(note que agora devemos mudar a tabela de empregados de modo que os atributos sejam pertinentes somente à entidade. Atributos como salário – salary – são pertinentes à contratação ou ao relacionamento entre o empregado e o seu departamento?)

De posse do diagrama E-R complete as tabelas que faltam, inclusive relacionamentos, e insira dados (pelo menos quatro linhas em cada tabela para continuarmos com os exercicios).

Exemplos

- **Seleciona nome**

```
SELECT name FROM departamento;
```

- **Seleciona nome de todos os Departamentos da região “Oeste”**

```
SELECT nome FROM departamento  
WHERE region= ‘Oeste’
```

- **Seleciona o nome de todos os empregados da região “Leste”**

```
SELECT empregado.nome  
FROM empregado E, departamento D,  
trabalha_em T  
WHERE (D.region= ‘Leste’ AND D.id=T.d_id  
AND T_e_id=E.id);
```

UPDATE

UPDATE <tabela>

SET <lista_atribuições>

{WHERE <expressão_seleção>}

Exemplos

- **Duplica todos os salários**

```
UPDATE trabalha_em  
    SET position='aux_escritorio'
```

- **Modifica a região dos deptos de Compras**

```
UPDATE departamento  
    SET region= 'especial'  
    WHERE nome= 'Compras'
```

CONCEITO

Uma subquery é um SELECT inserido em uma cláusula de outro comando SQL

```
SELECT { ALL | DISTINCT } <lista_itens>  
FROM <lista_tabelas>  
WHERE <expressão_seleção>
```

```
<expressão_seleção> ::=  
  <expr> <operator> (SELECT ....)
```


CONCEITO(ii)

<expressão_seleção> ::=

<expr> <operator> (SELECT)

<operator> ::= [> | = | >= | < | <> | <=] |

[IN | NOT IN]

Exemplo(i): Subquery retorna uma única linha

Obter o nome de todos os departamentos que atendem a um cliente chamado “Jarbas”

```
SELECT D.nome
FROM departamento D
WHERE D.id IN
    (SELECT vende.d_id FROM vende, cliente
     WHERE cliente.nome = 'Jarbas' AND
     cliente.cliente_id=vende.cliente.id)
```

Exemplo(ii): Subquery retorna várias linhas

Obter o nome de todos os funcionários que trabalham nos departamentos da região Sul

```
SELECT empregado.nome
FROM empregado, trabalha_em
WHERE empregado_id=trabalha_em.e_id
AND trabalha_em.d_id IN
    (SELECT id FROM departamento
     WHERE region= 'Sul' )
```

Operadores de agregação

- Count
- AVG
- MIN
- MAX
- SUM

Exemplo

- **Determina o número de funcionários**

```
SELECT count(*) FROM empregado;
```

- **Determina o salário médio do setor Pessoal**

```
SELECT AVG(T.salary)
FROM trabalha_em T, departamento D
WHERE D.name= 'Pessoal' AND D.id=T.d_id;
```


Agrupamento

GROUP BY <critério_agrupamento>

Formação de grupos

Uso: junto com SELECT

Seja o banco composto pelas seguintes tabelas,

```
SQL> create table dept
2 ( id          INT(7),
3   name        VARCHAR(25),
4   region_id  SMALLINT);
```

```
SQL> create table s_emp
2 ( id          INT(7),
3   name        CHAR(25),
4   dept        INT(7),
5   salary      DECIMAL(8,2));
```

Este banco foi feito só para exercicios e estará disponivel no site (o script)



Oracle SQL*Plus



File Edit Search Options Help

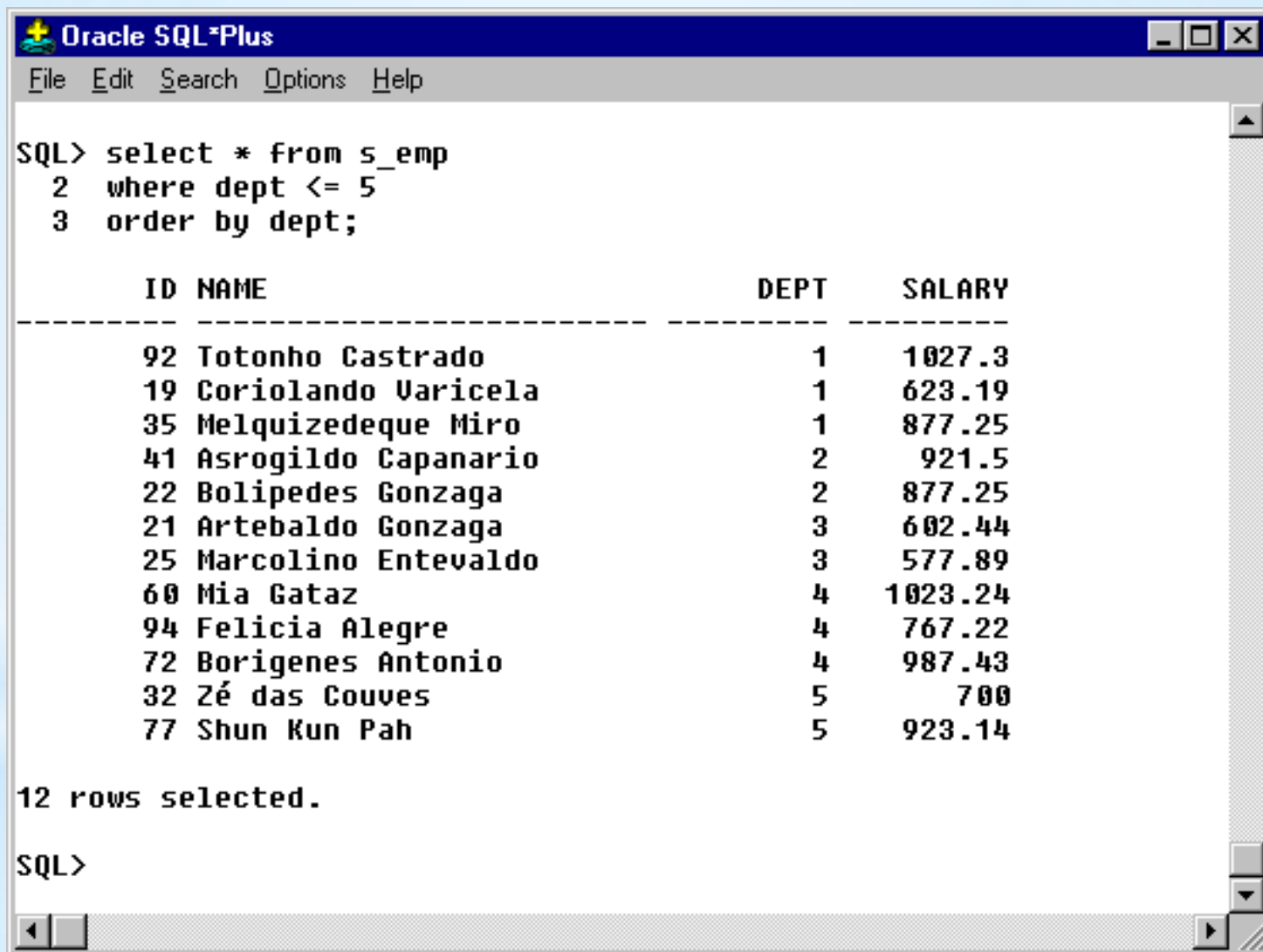
SQL> select * from dpt;

ID	NAME	REGION_ID
1	compras	1
2	compras	2
3	compras	5
4	peessoal	2
5	contas a pagar	1
6	marketing	2
7	inovacao tecnologica	3
8	inovacao tecnologica	1
9	meio ambiente	1
10	pesquisas	2
11	pesquisas	1
12	pesquisas	3

12 rows selected.



Exemplo : (agrupando empregados por dept)



The screenshot shows a window titled "Oracle SQL*Plus" with a menu bar containing "File", "Edit", "Search", "Options", and "Help". The main area displays the following SQL query:

```
SQL> select * from s_emp
 2  where dept <= 5
 3  order by dept;
```

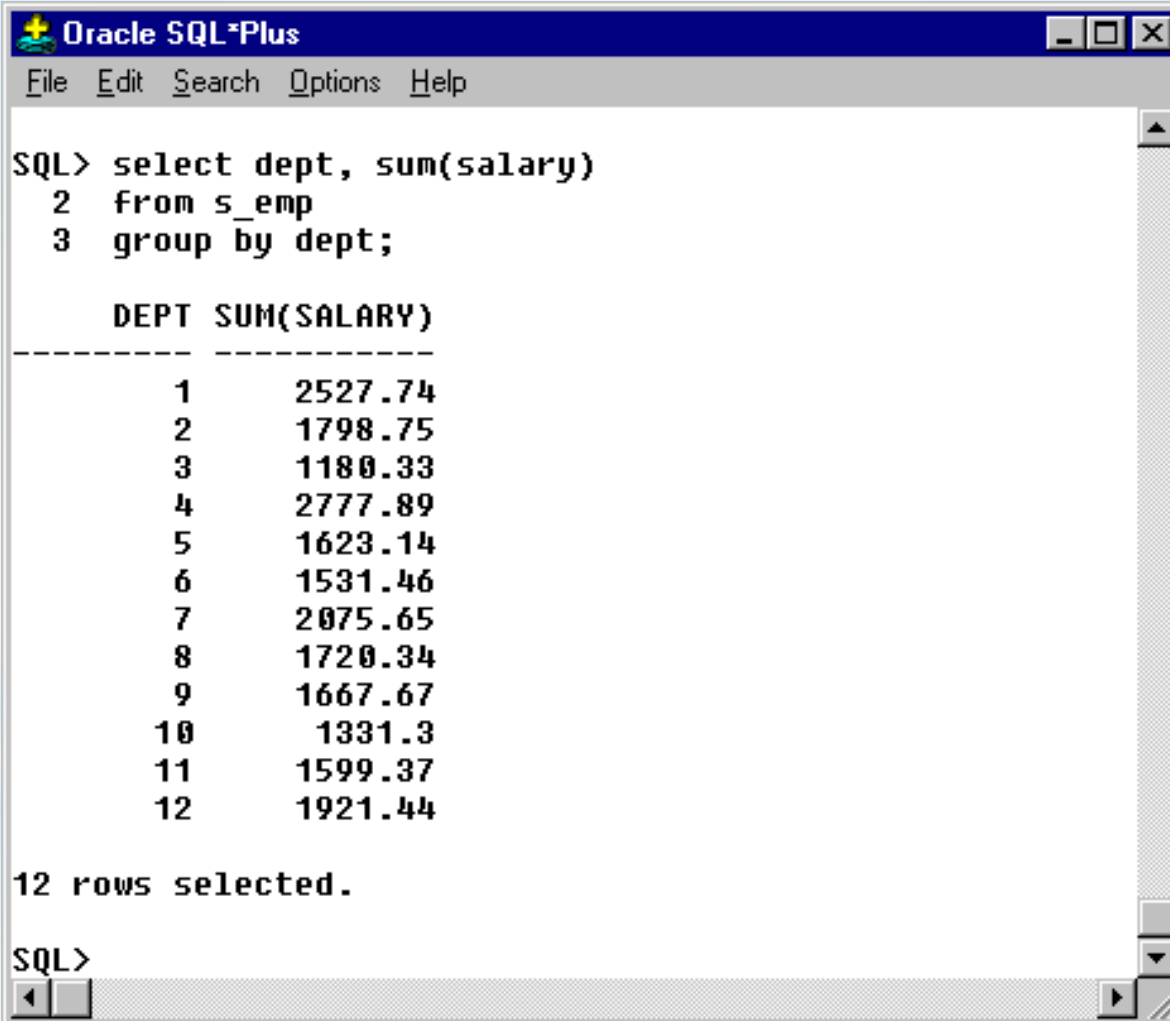
The results are displayed in a table with columns ID, NAME, DEPT, and SALARY. The data is sorted by department number. Below the table, it indicates "12 rows selected." and the prompt "SQL>" is visible at the bottom.

ID	NAME	DEPT	SALARY
92	Totonho Castrado	1	1027.3
19	Coriolando Varicela	1	623.19
35	Melquizedeque Miro	1	877.25
41	Asrogildo Capanario	2	921.5
22	Bolipedes Gonzaga	2	877.25
21	Artebaldo Gonzaga	3	602.44
25	Marcolino Entevaldo	3	577.89
60	Mia Gataz	4	1023.24
94	Felicia Alegre	4	767.22
72	Borigenes Antonio	4	987.43
32	Zé das Couves	5	700
77	Shun Kun Pah	5	923.14

12 rows selected.

SQL>

Exercicio : Calcular a soma dos salarios em cada departamento



The screenshot shows a window titled "Oracle SQL*Plus" with a menu bar containing "File", "Edit", "Search", "Options", and "Help". The main area displays the following SQL query and its output:

```
SQL> select dept, sum(salary)
2  from s_emp
3  group by dept;
```

DEPT	SUM(SALARY)
1	2527.74
2	1798.75
3	1180.33
4	2777.89
5	1623.14
6	1531.46
7	2075.65
8	1720.34
9	1667.67
10	1331.3
11	1599.37
12	1921.44

12 rows selected.

```
SQL>
```


Exercício

Como acontece frequentemente na prática, a empresa decidiu agora fechar os 3 departamentos de pesquisa (id's 10, 11 e 12) e os dois de inovação tecnológica (id's 7 e 8) em um novo departamento de inovação situado na região 3, transferindo os empregados do antigo departamento de id 7 para o marketing e os demais, todos para o novo departamento que terá o id 13. Isto seria uma “manutenção” no banco para refletir mudanças normais da empresa.

Faça os respectivos queries para executar este processo. Em que o design do banco faz este processo mais complicado? Seria melhor se o modelo conceitual fosse diferente? Como seria então?

GROUP BY Clause

Use the GROUP BY clause to group selected rows and return a single row of summary information. Oracle collects each group of rows based on the values of the expression(s) specified in the GROUP BY clause.

Exercicio : Como seria o query para listar o id do departamento, a soma dos salarios dos seus empregados e o salario máximo auferido por um deles?

```
Oracle SQL*Plus
File Edit Search Options Help

SQL> select dept, sum(salary), max(salary)
 2  from s_emp
 3  group by dept;

  DEPT  SUM(SALARY)  MAX(SALARY)
-----
      1    2527.74    1027.3
      2    1798.75     921.5
      3    1180.33     602.44
      4    2777.89    1023.24
      5    1623.14     923.14
      6    1531.46     877.25
      7    2075.65    1033.24
      8    1720.34     902.44
      9    1667.67     989.33
     10     1331.3      677.1
     11    1599.37    1032.12
     12    1921.44    1032.14

12 rows selected.

SQL> |
```

Exemplo

- Calcular o total de salários por departamento

```
SELECT dept, SUM(salary) FROM s_emp  
GROUP BY dept
```


Pertinência

HAVING <critério_pertinência>

Seleção baseada em alguma propriedade do grupo (ou seja, após o cálculo do GROUP BY)

Uso: junto com SELECT

Exemplo

- Calcular o total de salários por departamento de todos os departamentos com folha superior ao de um dado departamento (no caso, o 32)

```
SELECT dept, SUM(salary) FROM s_emp
GROUP BY dept
HAVING AVG(salary) >
    (SELECT AVG(salary) FROM s_emp
     WHERE dept = 32)
```

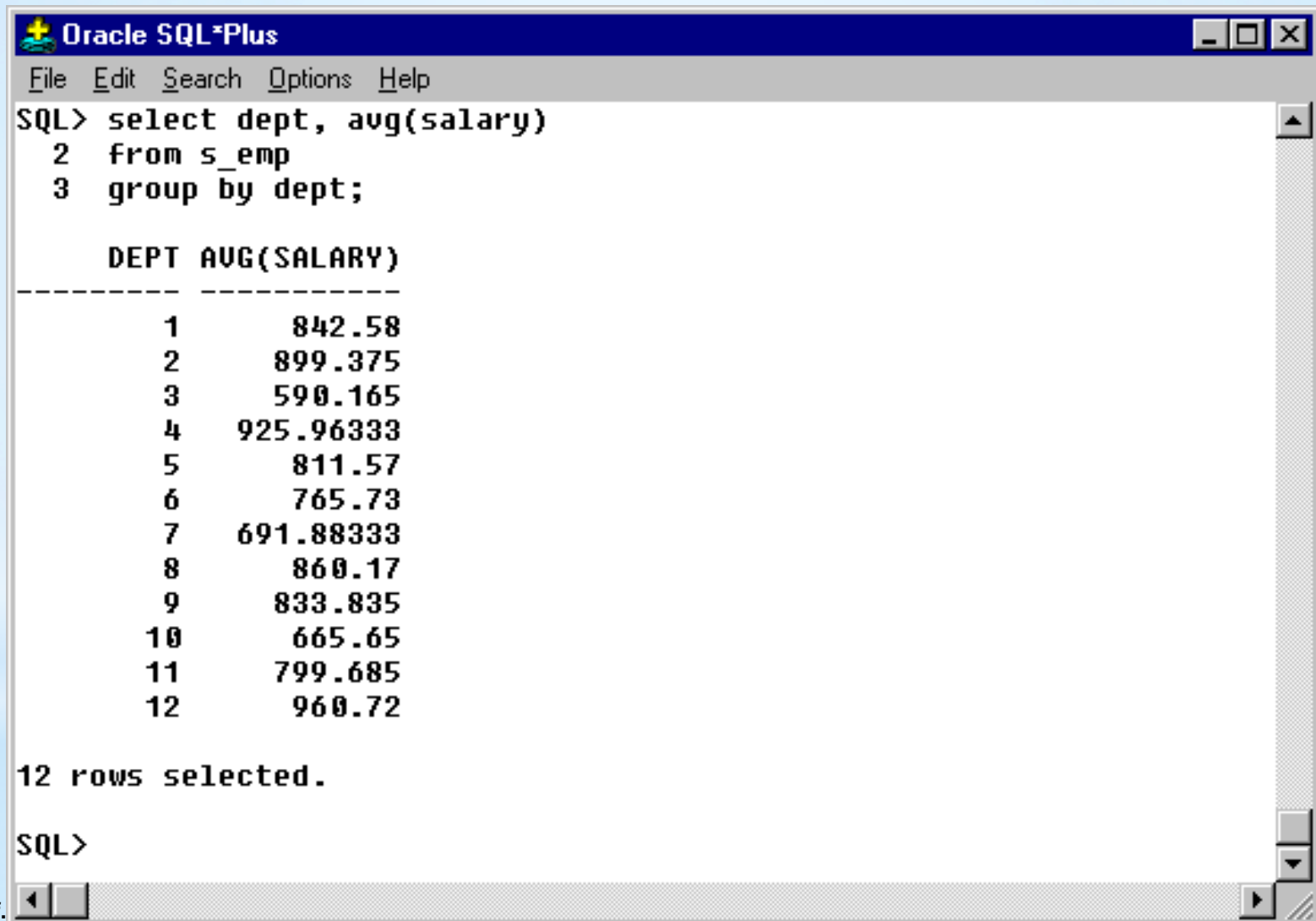
```
Oracle SQL*Plus
File Edit Search Options Help
SQL> select dept, sum(salary)
  2  from s_emp
  3  group by dept
  4  having avg(salary) > (select avg(salary) from s_emp
  5                        where dept=10);

  DEPT  SUM(SALARY)
-----
      1      2527.74
      2      1798.75
      4      2777.89
      5      1623.14
      6      1531.46
      7      2075.65
      8      1720.34
      9      1667.67
     11      1599.37
     12      1921.44

10 rows selected.

SQL> |
```

Conferindo ...



The screenshot shows the Oracle SQL*Plus interface. The title bar reads "Oracle SQL*Plus". The menu bar includes "File", "Edit", "Search", "Options", and "Help". The command prompt shows the following SQL query:

```
SQL> select dept, avg(salary)
2  from s_emp
3  group by dept;
```

The output is a table with two columns: "DEPT" and "AVG(SALARY)". The data is as follows:

DEPT	AVG(SALARY)
1	842.58
2	899.375
3	590.165
4	925.96333
5	811.57
6	765.73
7	691.88333
8	860.17
9	833.835
10	665.65
11	799.685
12	960.72

Below the table, the text "12 rows selected." is displayed. The prompt "SQL>" is visible at the bottom of the window.

Exemplo

F#	FNOME	STATUS	CIDADE
F1	Smith	20	Londres
F2	Jones	10	Paris
F3	Blake	30	Paris
F4	Clark	20	Londres
F5	Adams	30	Atenas

F

P#	PNOME	COR	PESO	CIDADE
P1	Porca	Vermelho	12.0	Londres
P2	Pino	Verde	17.0	Paris
P3	Parafuso	Axul	17.0	Roma
P4	Parafuso	Vermelho	14,0	Londres
P5	Came	Axul	12.0	Paris
P6	Tubo	Vermelho	19.0	Londres

P

J#	JNOME	CIDADE
J1	Classificador	Paris
J2	Monitor	Roma
J3	OCR	Atenas
J4	Console	Atenas
J5	RAID	Londres
J6	EDS	Oslo
J7	Fita	Londres

J

F#	P#	J#	QTDE
F1	P1	J1	200
F1	P1	J4	700
F2	P3	J1	400
F2	P3	J2	200
F2	P3	J3	200
F2	P3	J4	500
F2	P3	J5	600
F2	P3	J6	400
F2	P3	J7	800
F2	P5	J2	100
F3	P3	J1	200
F3	P4	J2	500
F4	P6	J6	300
F4	P6	J6	300
F5	P2	J2	200
F5	P2	J4	100
F5	P5	J5	500
F5	P5	J7	100
F5	P6	J2	200
F5	P1	J4	100
F5	P3	J4	200
F5	P4	J4	800
F5	P5	J4	400
F5	P6	J4	500

FPJ

1. Executar as seguintes queries:

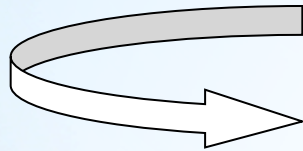
1. Dados um projeto, quantos e quais produtos foram pedidos para este projeto e de qual fornecedor?

2. Entre os projetos que requisitaram a peça P3 qual deles estará sendo atendido pelo fornecedor melhor ranqueado (maior valor de status) ?

3. Qual projeto estaria sendo atendido pelo fornecedor que tem sede no mesmo local onde está armazenado o produto ?

4. Faça uma lista de fornecedor, peça e a quantidade que este deve fornecer para os diversos projetos.

```
SQL> create table dept
 2 ( id NUMBER(7),
 3   name VARCHAR(25),
 4   region_id NUMBER(7));
```



```
SQL> alter table dept
 2   add primary key (id);
```

* Utilizando chaves ...

(AQUI O ESQUEMA FOI FEITO EM ORACLE, PORTANTO OS TIPOS DE DADOS SÃO UM POUCO DIFERENTES DAQUELES VISTOS EM SALA)

```
SQL> desc s_emp
```

Name	Null?	Type
ID		NUMBER(7)
NAME		VARCHAR2(25)
DEPT		NUMBER(7)
SALARY		NUMBER(8,2)

```
SQL> alter table s_emp  
2 add primary key (id);
```

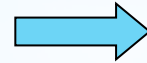
```
Table altered.
```

```
SQL> desc s_emp
```

Name	Null?	Type
ID	NOT NULL	NUMBER(7)
NAME		VARCHAR2(25)
DEPT		NUMBER(7)
SALARY		NUMBER(8,2)

```
SQL>
```

PRIMARY KEY



NOT NULL

NOT NULL



PRIMARY KEY

```
Oracle SQL*Plus
File Edit Search Options Help

SQL> desc s_emp
Name                               Null?    Type
-----
ID                                  NOT NULL NUMBER(7)
NAME                                UARCHAR2(25)
DEPT                                 NUMBER(7)
SALARY                              NUMBER(8,2)

SQL> alter table s_emp
  2 modify name not null;

Table altered.

SQL> desc s_emp
Name                               Null?    Type
-----
ID                                  NOT NULL NUMBER(7)
NAME                                NOT NULL UARCHAR2(25)
DEPT                                 NUMBER(7)
SALARY                              NUMBER(8,2)

SQL>
```




departamento

REFERENCIAL
INTEGRITY



empregado

```
Oracle SQL*Plus
File Edit Search Options Help

SQL> alter table s_emp
  2  add foreign key (dept) references scott.dpt(id);

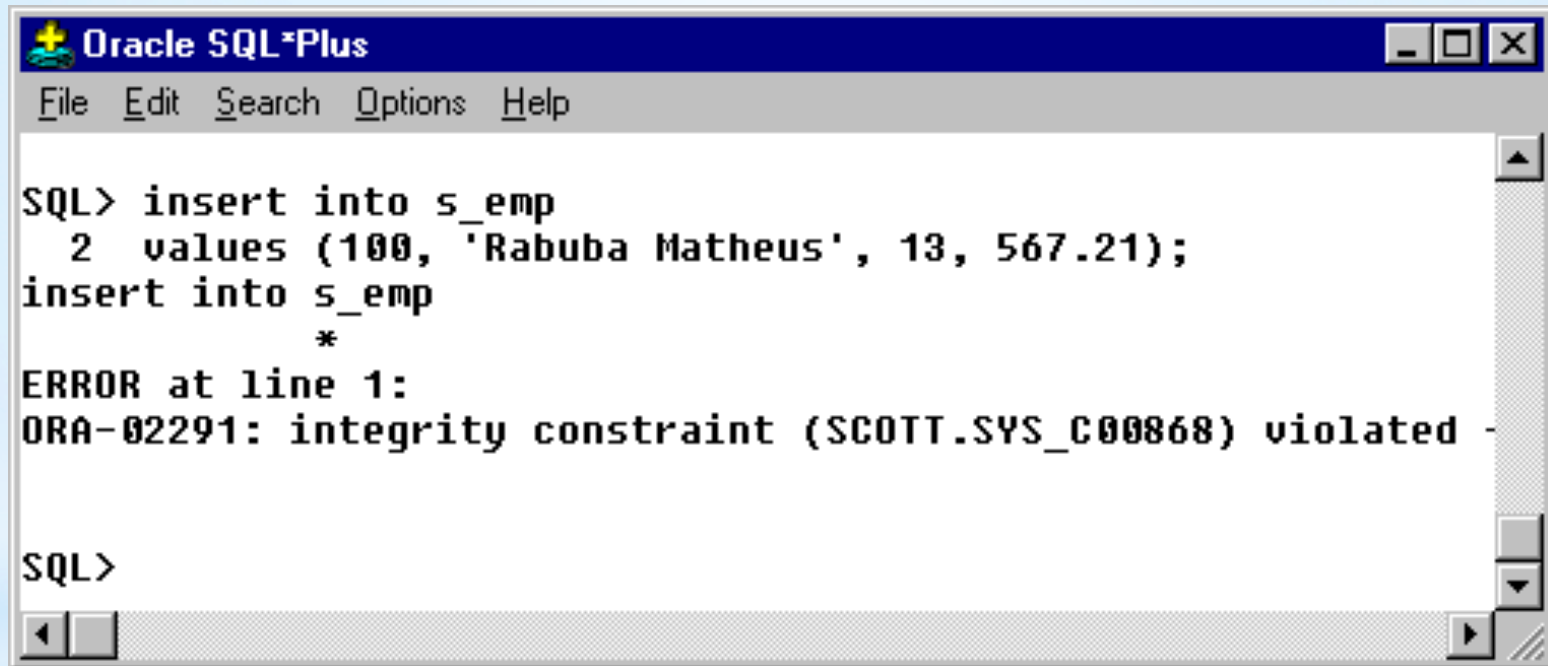
Table altered.

SQL> desc s_emp
Name                                Null?    Type
-----
ID                                    NOT NULL NUMBER(7)
NAME                                  NOT NULL VARCHAR2(25)
DEPT                                   NUMBER(7)
SALARY                                NUMBER(8,2)

SQL> |
```

* Utilizando chave estrangeira

Vamos inserir uma funcionaria em um departamento que ainda NÃO EXISTE ...

A screenshot of the Oracle SQL*Plus application window. The title bar reads "Oracle SQL*Plus" and the menu bar includes "File", "Edit", "Search", "Options", and "Help". The main text area shows the following SQL command and its execution result:

```
SQL> insert into s_emp  
  2 values (100, 'Rabuba Matheus', 13, 567.21);  
insert into s_emp  
      *  
ERROR at line 1:  
ORA-02291: integrity constraint (SCOTT.SYS_C00868) violated
```

The prompt "SQL>" is visible at the bottom left of the window.

Modifique as suas tabelas de modo a inserir chaves primárias e atender aos requisitos de integridade referencial existentes.

 **EXERCICIO**

Creating Indexes

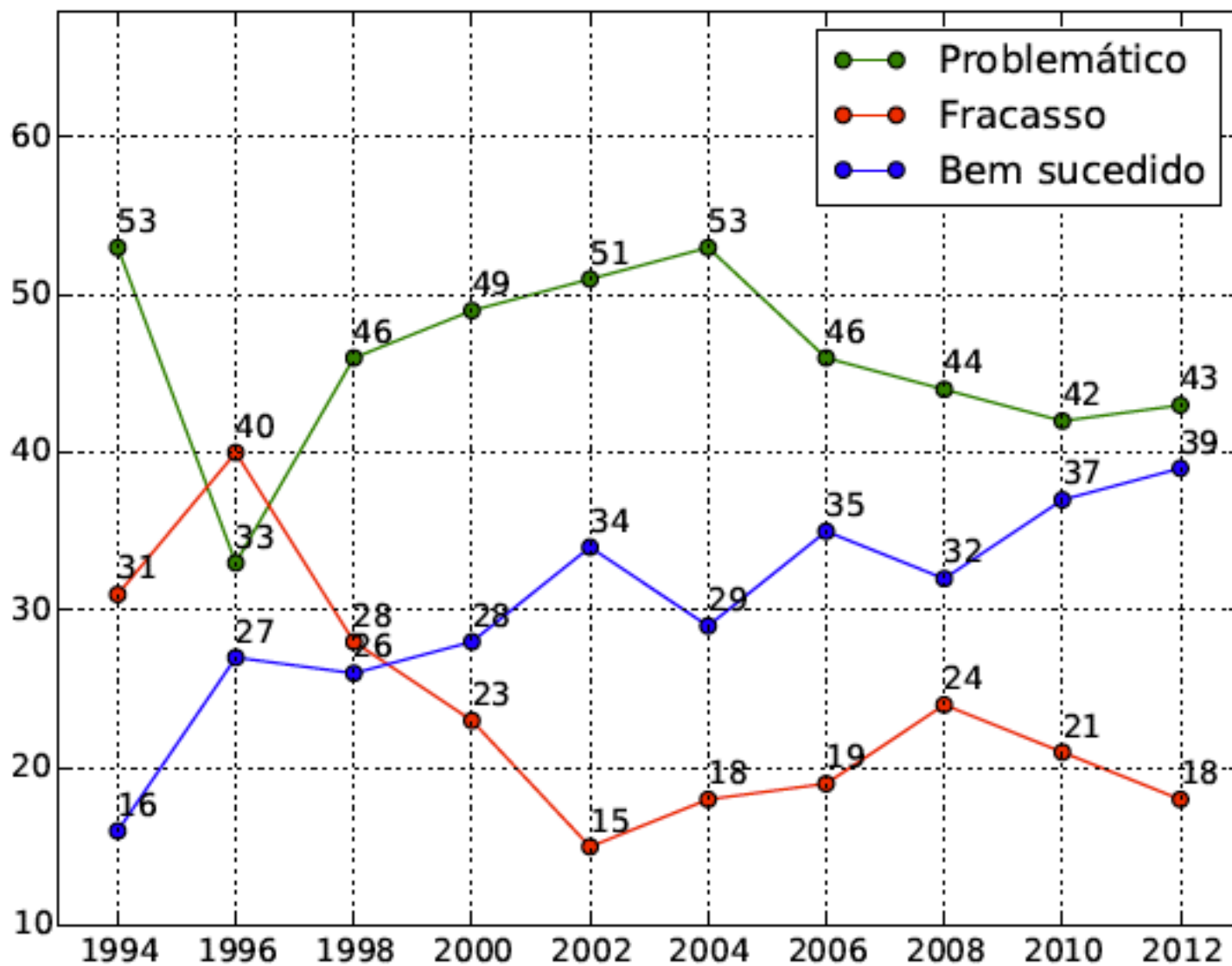
An index is an ordered list of all the values that reside in a group of one or more columns at a given time. Such a list makes queries that test the values in those columns vastly more efficient. However, indexes

take up data storage space and must be changed whenever the data is changed. Therefore, you should make a cost-benefit analysis in each case to determine whether and how indexes should be used. Oracle can use indexes to improve performance when:

- searching for rows with specified index column values
- accessing tables in index column order

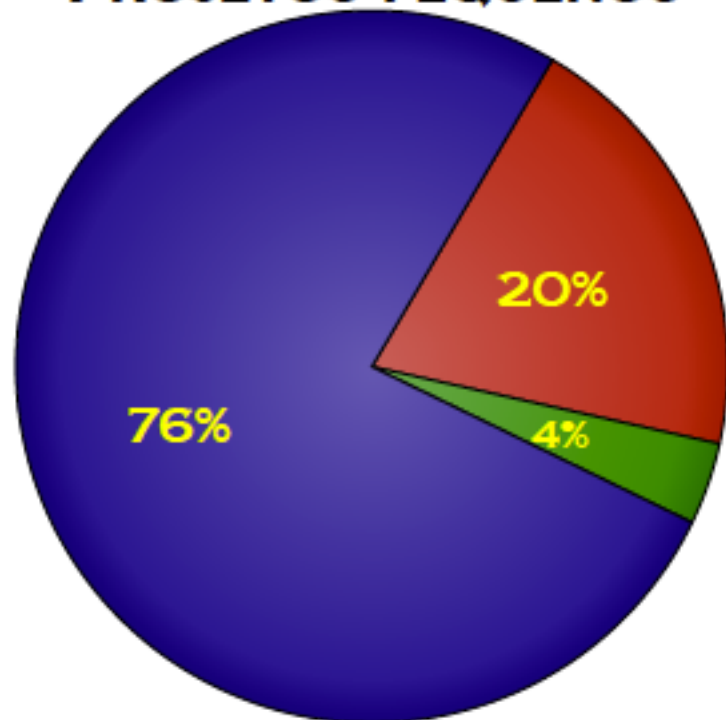
Requisitos: Uma fase essencial de qualquer projeto

José Reinaldo Silva

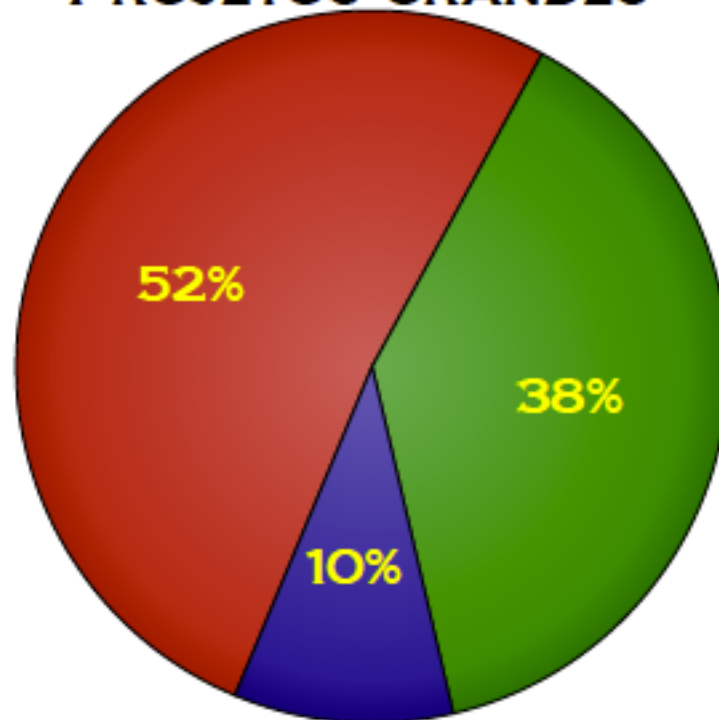


AVALIAÇÃO SEGUNDO CHAOS SOBRE O SUCESSO DE PEQUENOS E GRANDES PROJETOS (2012)

PROJETOS PEQUENOS



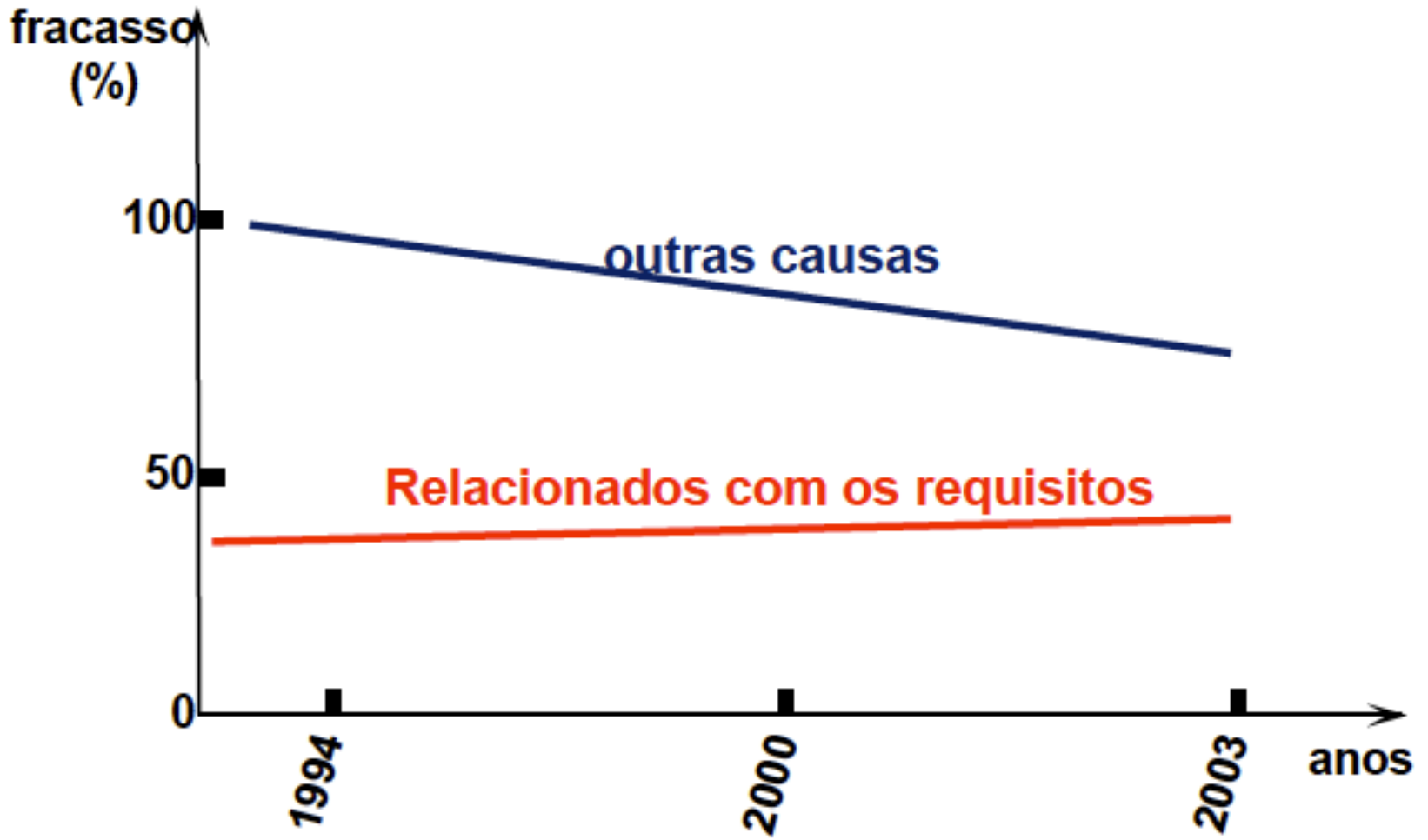
PROJETOS GRANDES



- BEM SUCEDIDO
- FRACASSO
- PROBLEMÁTICOS

Projetos pequenos: o custo de desenvolvimento não excede US \$1 milhão.

Projetos grandes: o custo de desenvolvimento é maior a US \$10 milhões.



[J. Maresco, IBM developersWork, 2007]

fracasso (%)

100

50

0

1994

2000

2003

anos

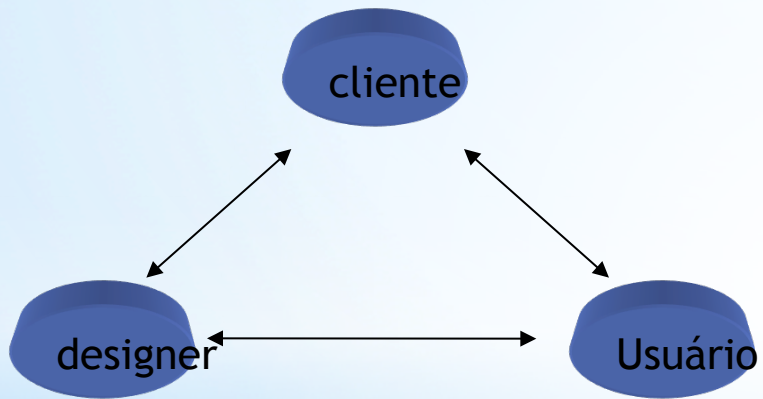
Requirements



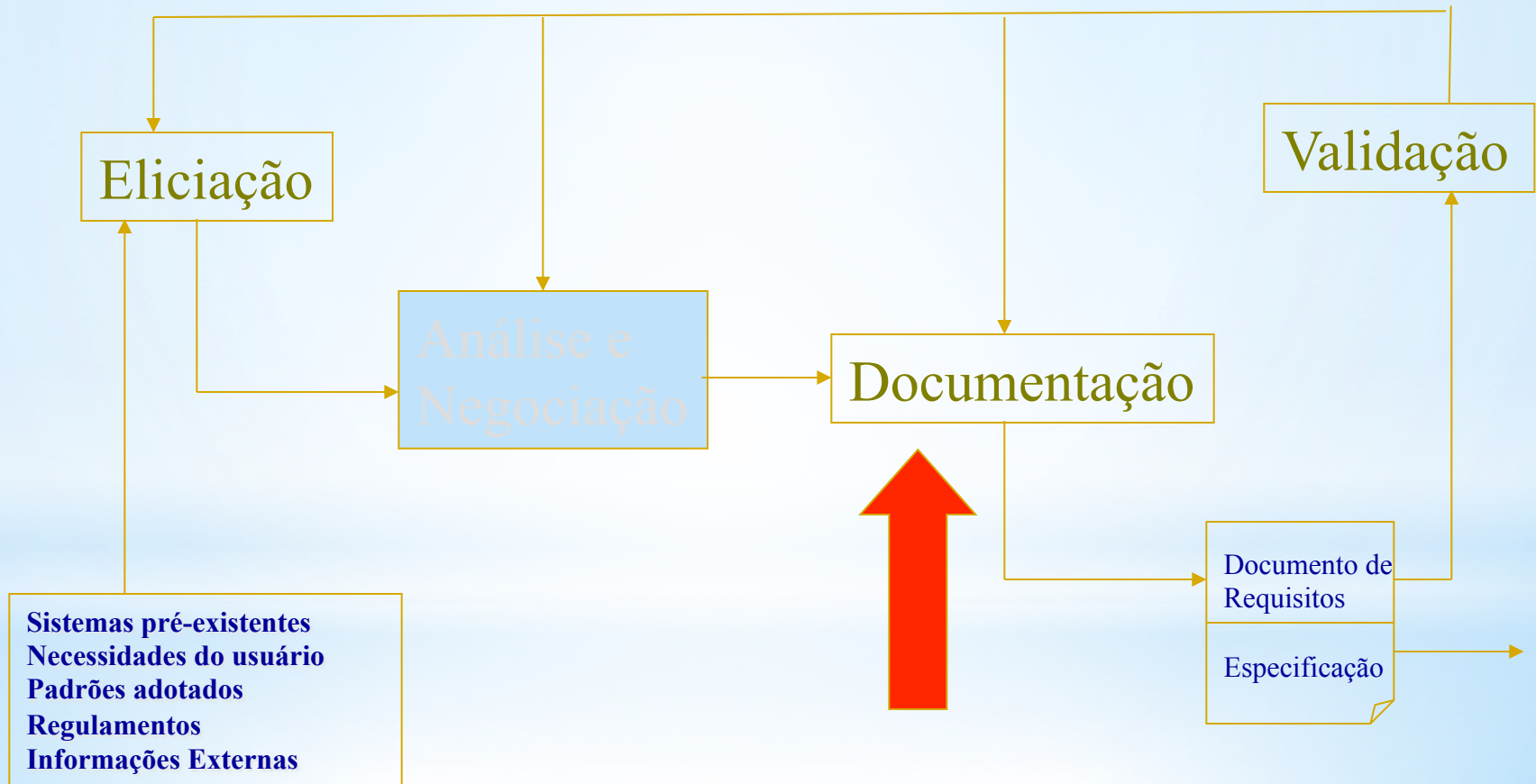
Relacionados com os requisitos



Classes de pontos de vista e classes de agentes



O ciclo de avaliação de requisitos



* Métodos para análise de requisitos

Processo de engenharia de requisitos, composto da eliciação, análise, validação e documentação, é feito segundo métodos que de fato são propostas de sistematizar a modelagem de sistemas - especialmente nesta fase preliminar.

Alguns destes métodos são a base da pesquisa nesta área e ficaram conhecidos por suas características básicas.

* Métodos Básicos

Os métodos podem ser caracterizados pelo respectivo esquema de representação:

1. Data-flow models - diagramas de fluxo de dados
2. Compositional models - baseados em diagramas Entidade-Relação
3. Classification models - baseado em diagramas de objeto
4. Stimulus-response models - baseados em diagramas estado-transição
5. Process models - diagramas de processo, redes de Petri, álgebra de processos, statecharts.

Para a próxima aula:

Vocês devem ter postado no sistema Moodle-STOA os diagramas de caso-de-uso já que não viram estes diagramas antes deste curso. Devemos agora discutir estes diagramas para saber qual é de fato um diagrama de caso de uso e inserir os diagramas de sequencia e classe. O Marcos deve ter falado (ou vai fazer isso esta semana) sobre isso na aula teórica.

Vocês receberão algum feedback pelo sistema Moodle sobre o que fizeram até agora e devem se reunir para discutir o que vão fazer em seguida.