

Bases de Dados

José Machado
Departamento de Informática
Escola de Engenharia
Universidade do Minho

jmac@di.uminho.pt

2010

Sistemas de Informação

Definição

Numa perspectiva estrutural:

um Sistema de Informação (SI) é um agrupamento de pessoas, processos, dados, modelos, tecnologia e linguagens parcialmente formalizadas, formando uma estrutura coesa, servindo algum propósito ou função organizacional.

Numa perspectiva funcional:

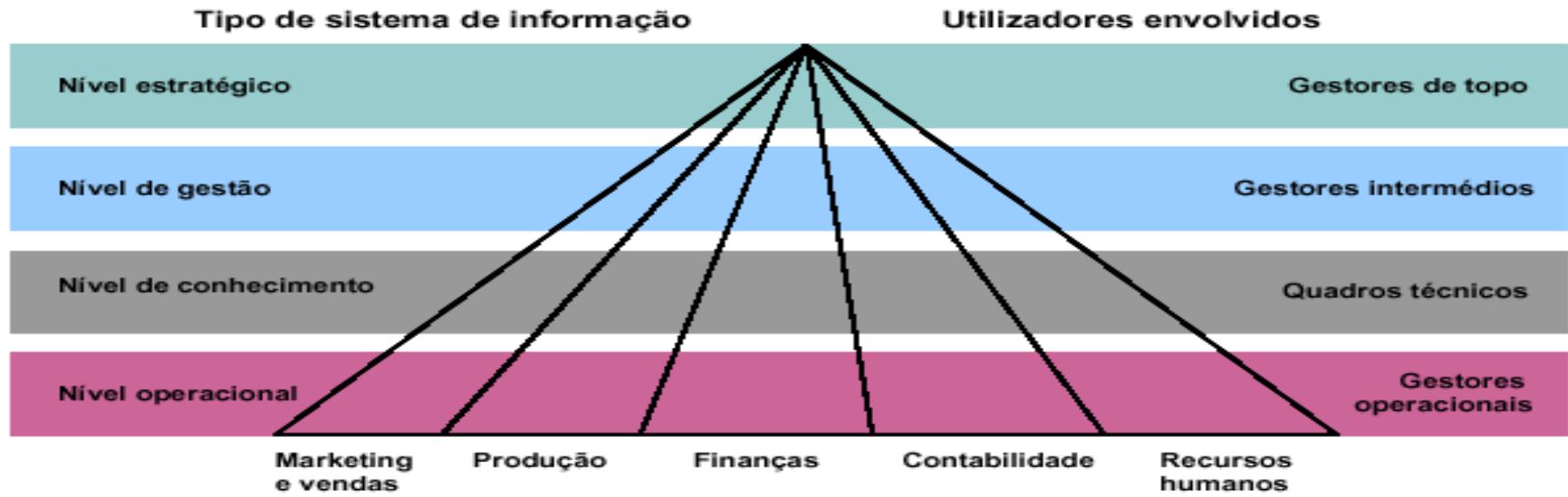
um SI é um meio implementado tecnologicamente para o registo, armazenamento, e disseminação de expressões linguísticas, assim como para o apoio à realização de inferências. O SI facilita a criação e troca de significados que servem propósitos definidos socialmente tais como o controlo, o dar sentido e a argumentação.



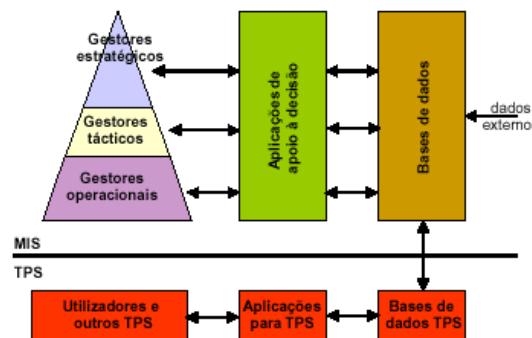
Dados, Informação e Conhecimento



Tipos de Sistemas de Informação



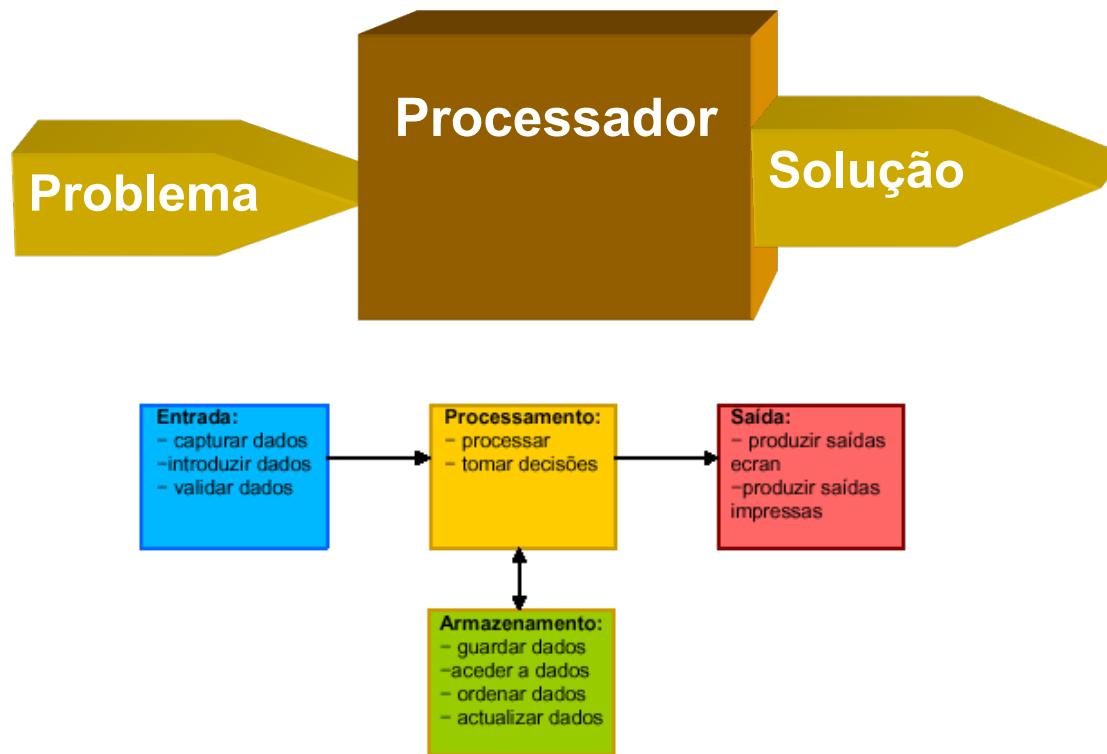
Fonte: Laudon e Laudon, 2000



Fonte: R. Nickerson, 2000



Resolução de Problemas

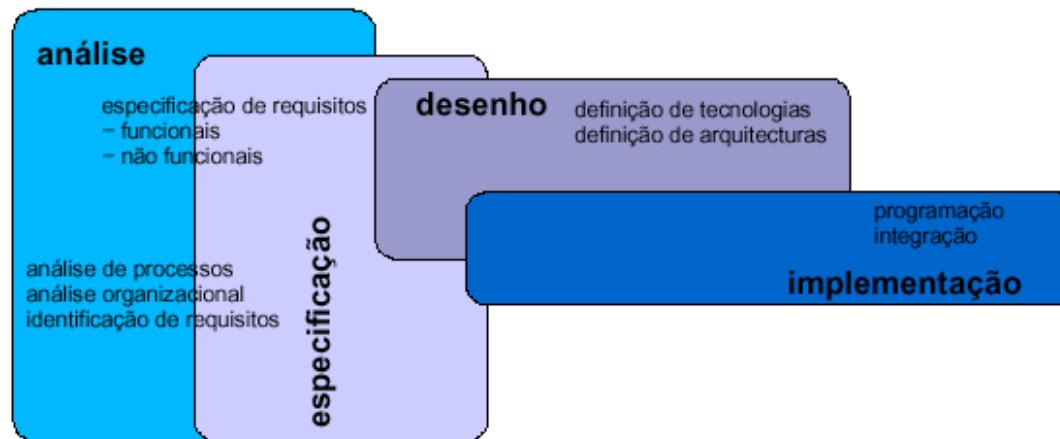


Fonte: R. Nickerson, 2000



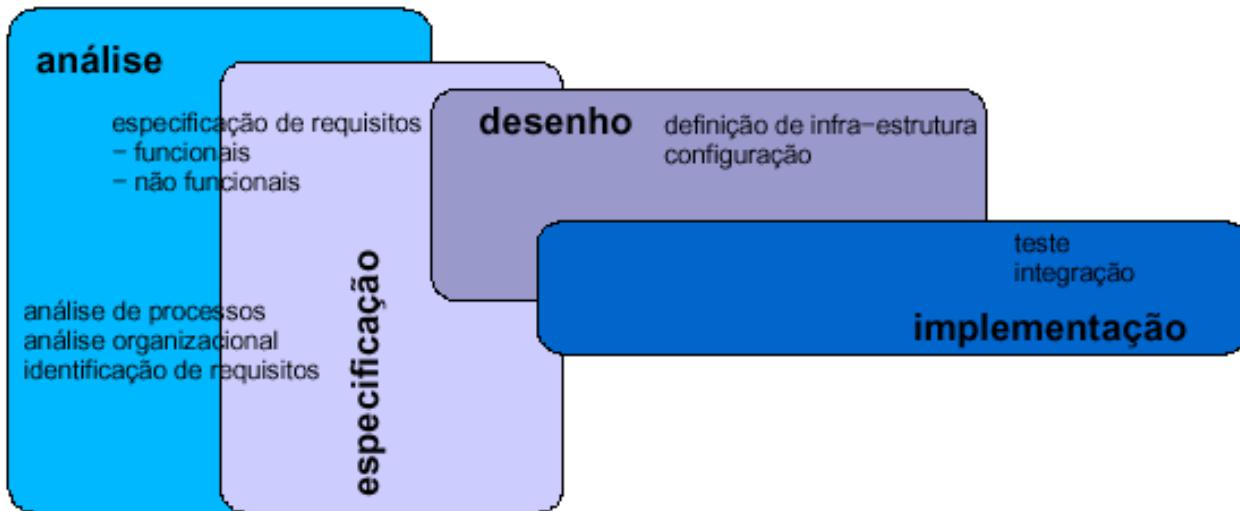
SI à medida vs SI adaptado

ideia, necessidade, oportunidade



sistema em funcionamento

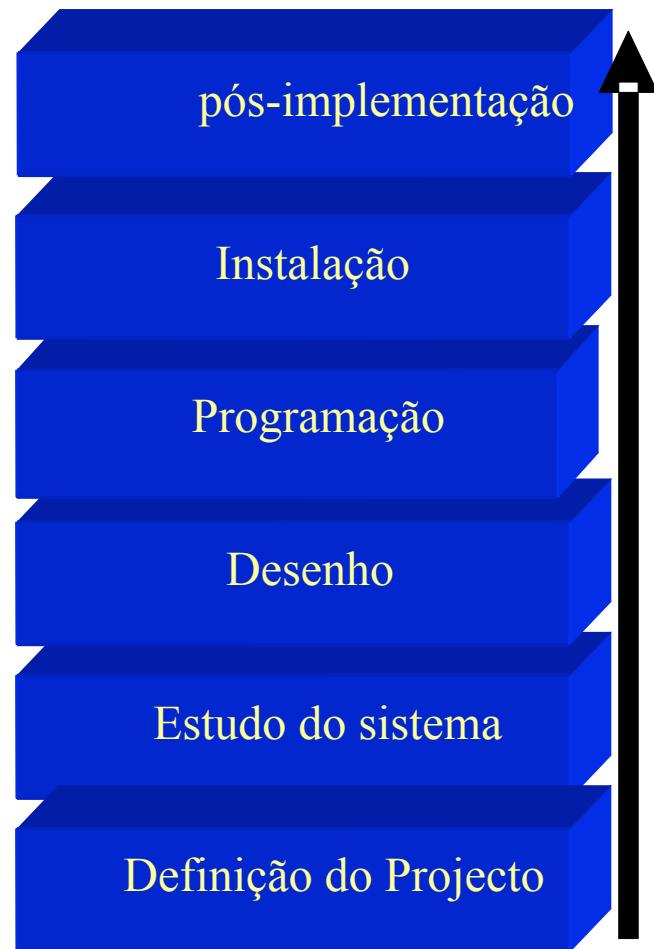
ideia, necessidade, oportunidade



sistema em funcionamento



Ciclo de Vida de um SI



Auditoria pós-implementação

Testes de desempenho

Programa

Especificação

Proposta de Sistema

Proposta de projecto



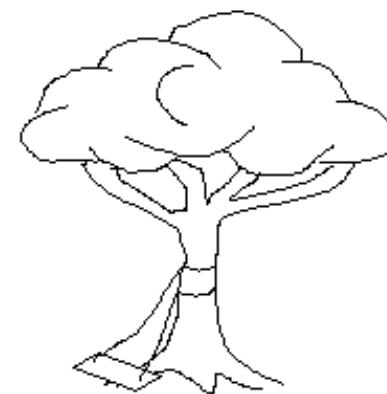
O Custo do Erro



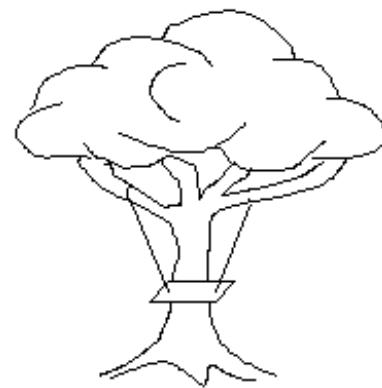
O que o cliente pede



O que o contrato prevê



O que o analista previu



O que o programador escreveu



O que, de facto, foi feito



O que era pretendido

O Modelo Relacional

- Resultado do trabalho de E.F.Codd na IBM durante a década de 70.
- Oracle Server da Oracle Corp.; Informix SE e RDS e DB2 da IBM, SQL Server da Microsoft, Sybase SQL da Sybase Inc. entre outros, são os mais vendidos dos grandes fabricantes que aderiram aos SGBDs seguindo o modelo relacional.



As 12 Regras de Codd

- 1) Numa base de dados relacional, todos os dados, incluindo o próprio dicionário de dados, são representados de uma só forma, em tabelas bidimensionais.
- 2) Cada elemento de dados fica bem determinado pela combinação do nome da tabela onde está armazenado, valor da chave primária e respectiva coluna (atributo).
- 3) Os valores nulos são suportados para representar informação não disponível ou não aplicável, independentemente do domínio dos respectivos atributos.
- 4) Os metadados são representados e acedidos da mesma forma que os próprios dados.
- 5) Apesar de um sistema relacional poder suportar várias linguagens, deverá existir pelo menos uma linguagem com as seguintes características:
 - Manipulação de dados, com possibilidade de utilização interactiva ou em programas de aplicação.
 - Definição de dados.
 - Definição de views.
 - Definição de restrições de integridade.
 - Definição de acessos (autorizações).
 - Manipulação de transacções (commit, rollback, etc.).
- 6) Numa view, todos os dados actualizáveis que forem modificados, devem ver essas modificações traduzidas nas tabelas base.



As 12 Regras de Codd

- 7) Há a capacidade de tratar uma tabela (base ou virtual) como se fosse um simples operando (ou seja, utilização de uma linguagem set-oriented), tanto em operações de consulta como de actualização.
- 8) Alterações na organização física dos ficheiros da base de dados ou nos métodos de acesso a esses ficheiros (nível interno) não devem afectar o nível conceptual – independência física.
- 9) Alterações no esquema da base de dados (nível conceptual), que não envolvam remoções de elementos, não devem afectar o nível externo – independência lógica.
- 10) As restrições de integridade devem poder ser especificadas numa linguagem relacional, independentemente dos programas de aplicação, e armazenadas no dicionário de dados.
- 11) O facto de uma base de dados estar centralizada numa máquina, ou distribuída por várias máquinas, não deve repercutir-se ao nível da manipulação de dados.
- 12) Se existir no sistema uma linguagem de mais baixo nível (tipo record-oriented), ela não deverá permitir ultrapassar as restrições de integridade e segurança.

Não existe nenhuma implementação do modelo relacional que, à luz das doze regras de Codd, possa ser considerada completamente relacional.



Sistemas de Bases de Dados

- organizam grandes quantidades de informação.
- colecção de relações, que se encontram relacionadas através de atributos comuns.
- uma linguagem simples, para fazer a manutenção da base de dados (através de inserções, modificações ou remoções de informação) e para responder a questões.
- uma linguagem estruturada que facilita a consulta aos dados organizados na base de dados (e.g., SQL).

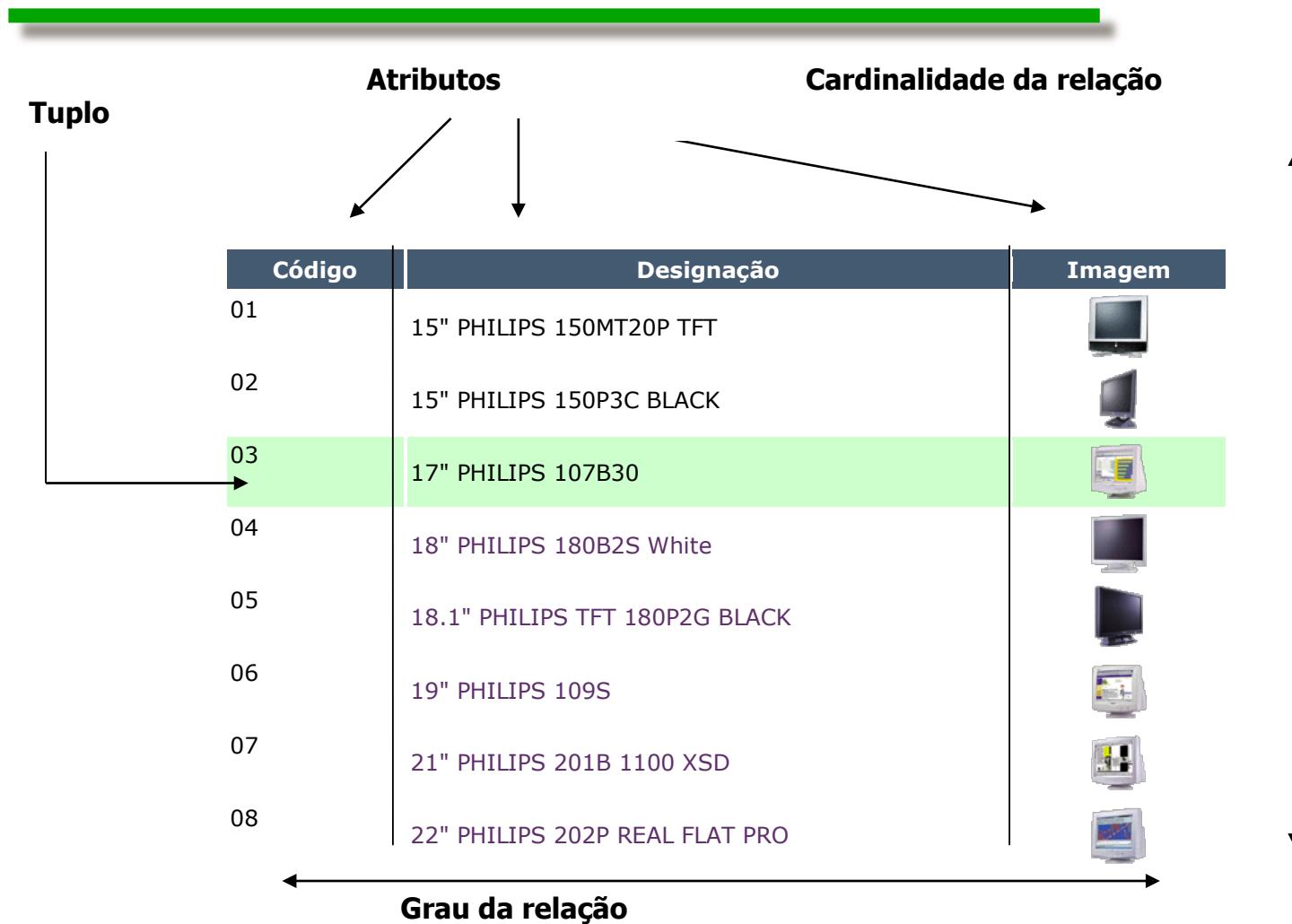


Objectos de um BD

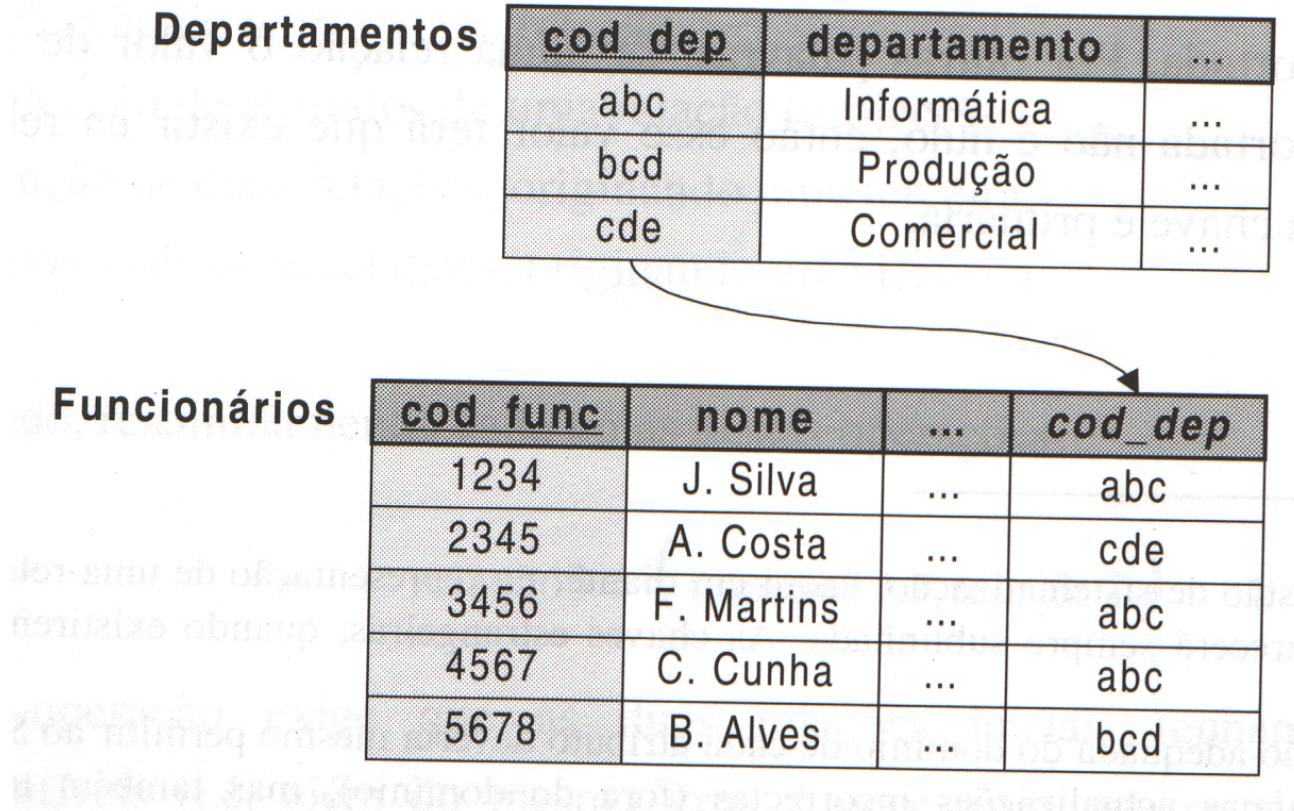
- Um sistema de bases de dados relacionais contêm um ou mais objectos chamados tabelas.
- Os dados ou informação são armazenados nessas tabelas.
- As tabelas são apenas identificadas pelo nome e contêm colunas e linhas.
- As colunas contêm o nome da coluna, o tipo de dado, e qualquer outro atributo para a coluna.
- As linhas contêm os registos ou dados para as colunas.



A tabela : Produtos



Relacionamentos



Conceitos Básicos

- **relação (tabela)** constitui uma estrutura bidimensional.
- **atributos** da relação - 1 ou mais colunas.
- **tuplos** da relação – 0 ou mais linhas.

O *esquema conceptual* de uma relação é o conjunto dos seus atributos, que traduzem o tipo de dados a armazenar.



Chaves

- **Chave candidata** – subconjunto de atributos de uma relação que , em conjunto, identificam univocamente qualquer tuplo, e que não pode ser reduzido sem perder essa qualidade.
- **Chave primária** – chave seleccionada de entre as diversas chaves candidatas, para efectivamente identificar cada tuplo.
- **Chave estrangeira** – atributo, ou conjunto de atributos, de uma relação que é chave primária numa outra relação.



Entidades e Relacionamentos

Entidades – uma entidade será representativa de uma classe de objectos sobre os quais se quer guardar informação.

Ex: – numa clínica as entidades poderão ser: médicos, pacientes, especialidades, etc.

As entidades correspondem, ao nível do modelo relacional, a *relações*. Cada instância de uma entidade será caracterizada pelos valores de um conjunto de atributos. Ex: um médico tem o seu *nome, morada, especialidade* etc.



Entidades e Relacionamentos

Entre entidades estabelecem-se um conjunto de relações:

- Relações de 1 para 1.
- Relações de 1 para muitos (1 para ∞).
- Relações de muitos para muitos (∞ para ∞).

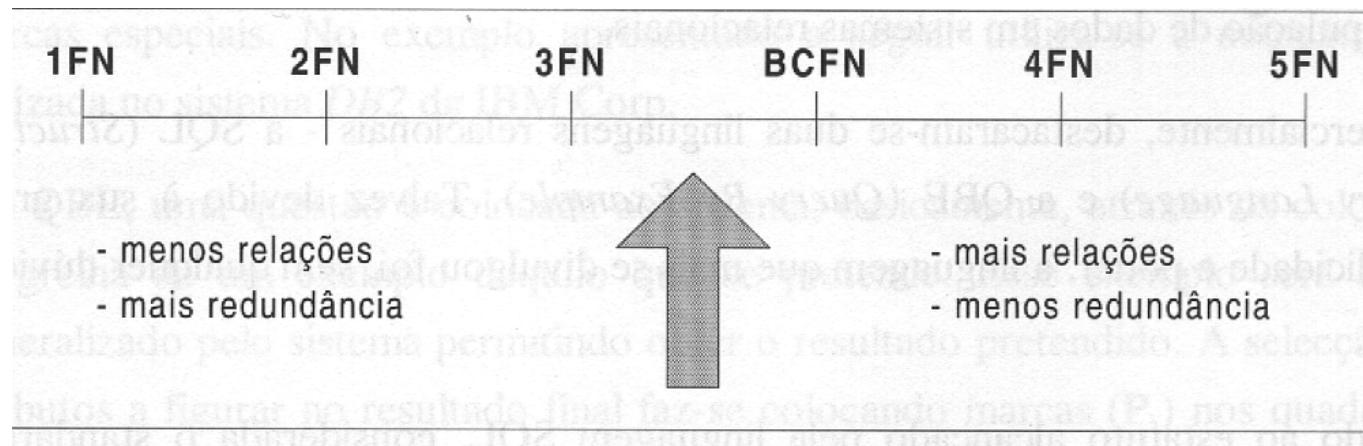
Estas últimas levam à criação de novas entidades no processo de normalização.



Normalização da Informação

A normalização é um processo sistemático, definido por um conjunto de regras bem definidas, que visa eliminar fontes de redundância nos dados:

- Problemas de manutenção;
- Custos de espaço de armazenamento;
- Problemas de desempenho.



Exemplo

Consideremos a ficha de consulta de uma clínica médica:

Cod.Medico: _____ Nome: _____ Especialidade : _____ --- _____							
Morada Med. : _____ Contacto: _____ Idade : _____							
CONSULTAS:							
Data	Hora	N.Benif	Paciente	Idade	Morada	Cod.Postal	Preço



Normalização

Considere as seguintes restrições:

- Um médico só consulta uma especialidade.
- O preço da consulta depende da especialidade.
- Durante a consulta podem ser feitos tratamentos ao paciente que serão pagos em conjunto com a consulta.

A clínica tem 30 médicos a consultar 10 especialidades, para um universo de 10000 pacientes, tendo cerca de 50 consultas por dia.



Normalização : 1^a FN

- O primeiro passo do processo de normalização (1FN), visa eliminar grupos de valores repetidos para um dado atributo, ou conjunto de atributos num dado tuplo.
- Analisando o exemplo, verifica-se que para a entidade médico podem existir várias consultas.
- A solução passa por decompor a relação inicial criando novas relações, tantas quantas o número de grupos que se repetem.



Normalização : 1^a FN

- A nova relação teria como atributos a chave primária da relação de onde é originária, bem como os atributos que fazem parte do grupo que se repete.
- O esquema conceptual da base de dados na 1FN é o seguinte:
 - Médico:(cod_med,nome,morada,contacto,idade,cod_esp,especialidade,preço)
 - Consulta(cod_med,data,hora,n_ben,paciente,idade,morada,cod_post,localidade,preço)
- Note-se que a chave da tabela criada é *quadrupla*, incluindo a chave da tabela inicial e os atributos *data*, *hora* e *n_ben*.



Dependências

- Todo o processo necessário à normalização está traduzido nas dependências existentes entre os dados, existindo três tipos de dependências.
- **Funcionais** – existe uma dependência funcional $X \rightarrow Y$ entre dois conjuntos de atributos X e Y , se uma instância de valores de X determina ou identifica univocamente uma instância de valores dos atributos Y . (este conceito vai ser utilizado para estabelecer a 2^a FN, a 3^a FN e a *Boyce-Codd FN*).
- **Multivalor** – numa relação $R(X,Y,Z)$ diz-se que existe uma dependência multivalor $X \twoheadrightarrow Y$ (X multidetermina Y) se, para cada par de tuplos de R contendo os mesmos valores de X também existe em R um par de tuplos correspondentes à troca de Y no par original (é com base nesta dependência que se desenvolve a 4º FN).
- **Junção** - existe uma dependência de junção se, dadas algumas projecções sobre essa relação, apenas se reconstroí a relação inicial através de algumas junções bem específicas mas não de todas (é com base nesta dependência que se desenvolve a 5º FN).



Dependências Funcionais

Existe uma dependência funcional $X \rightarrow Y$ entre 2 conjuntos de atributos X e Y, se uma instância de valores dos atributos de X determina ou identifica univocamente uma instância de valores dos atributos de Y. Não existem 2 instâncias distintas de Y para uma mesma instância de X.

Se $X \rightarrow Y$ então existe uma dependência funcional entre X e Y em que X determina Y ou Y depende de X. Por exemplo:

$\text{NºFuncionário} \rightarrow \text{Nome_funcionário, Departamento}$
 $(\text{NºFactura, Cod_produto}) \rightarrow \text{Qtd_vendida, Preço_venda}$

Por definição se $X \rightarrow Y$ então a correspondência entre X e Y é do tipo 1:1 ou M:1



Dependências Funcionais

Axiomas de Armstrong :

Reflexividade (se $X \subseteq Y$ então $X \rightarrow Y$)

Aumentatividade (se $X \rightarrow Y$ então $XZ \rightarrow YZ$)

Transitividade (se $X \rightarrow Y$ e $Y \rightarrow Z$ então $X \rightarrow Z$)

Regras de Inferência :

Decomposição (se $X \rightarrow YZ$ então $X \rightarrow Y$ e $X \rightarrow Z$)

União (se $X \rightarrow Y$ e $X \rightarrow Z$ então $X \rightarrow YZ$)

Pseudotransitividade (se $X \rightarrow Y$ e $YW \rightarrow Z$ então $XW \rightarrow Z$)

O conceito de dependência funcional é um caso especial de uma outra, a dependência multi-valor, que apenas se prova se a relação tiver pelo menos 3 atributos. A 4^a forma normal usa este tipo de dependência.



2^a Forma Normal

- Uma relação na 2FN é uma relação em que todos os atributos não pertencentes a qualquer chave candidata, devem depender da totalidade da chave.
- Retomando o exemplo verificamos que na relação *Consulta* existem duas dependências funcionais:
 - Consulta(cod_med,data,hora,n_ben,paciente,idade,morada,cod_post,localidade,preço)
 - cod_med,data,hora,n_ben → preço
 - n_ben → paciente,idade,morada,cod_post,localidade
- Esta ultima dependência está em desacordo com definição da 2ºFN.



2^a Forma Normal

- A solução passa por decompor a relação de acordo com as dependências funcionais anteriores:
 - Consulta(cod_med,data,hora,n_bem,preço)
 - Paciente(n_ben,paciente,idade,morada,cod_post,localidade)
- O problema detectado está agora resolvido, a base de dados está na 2^a FN.



3^a Forma Normal

- Uma relação na 3^aFN é uma relação em que não existem dependências funcionais entre atributos não chave (dependências transitivas).
- Retomando o exemplo verificamos que na relação *Médicos* existem duas dependências funcionais:
 - Médico:(cod_med,nome,morada,contacto,idade,cod_esp,especialidade,preço)
 - cod_med → nome,morada,contacto,idade,cod_esp
 - cod_esp → especialidade,preço
- Esta ultima dependência está em desacordo com definição da 3^aFN.



3^a Forma Normal

- A solução passa por decompor a relação de acordo com as dependências funcionais anteriores:
 - Medicos(cod_med, nome,morada,contacto,idade,cod_esp_)
 - Especialidades(cod_esp, especialidade,preço)
- na relação *Pacientes* existem duas dependências funcionais:
 - Paciente(n_ben,paciente,idade,morada,cod_post,localidade)
 - n_ben → paciente,idade,morada,cod_pos
 - cod_post → localidade

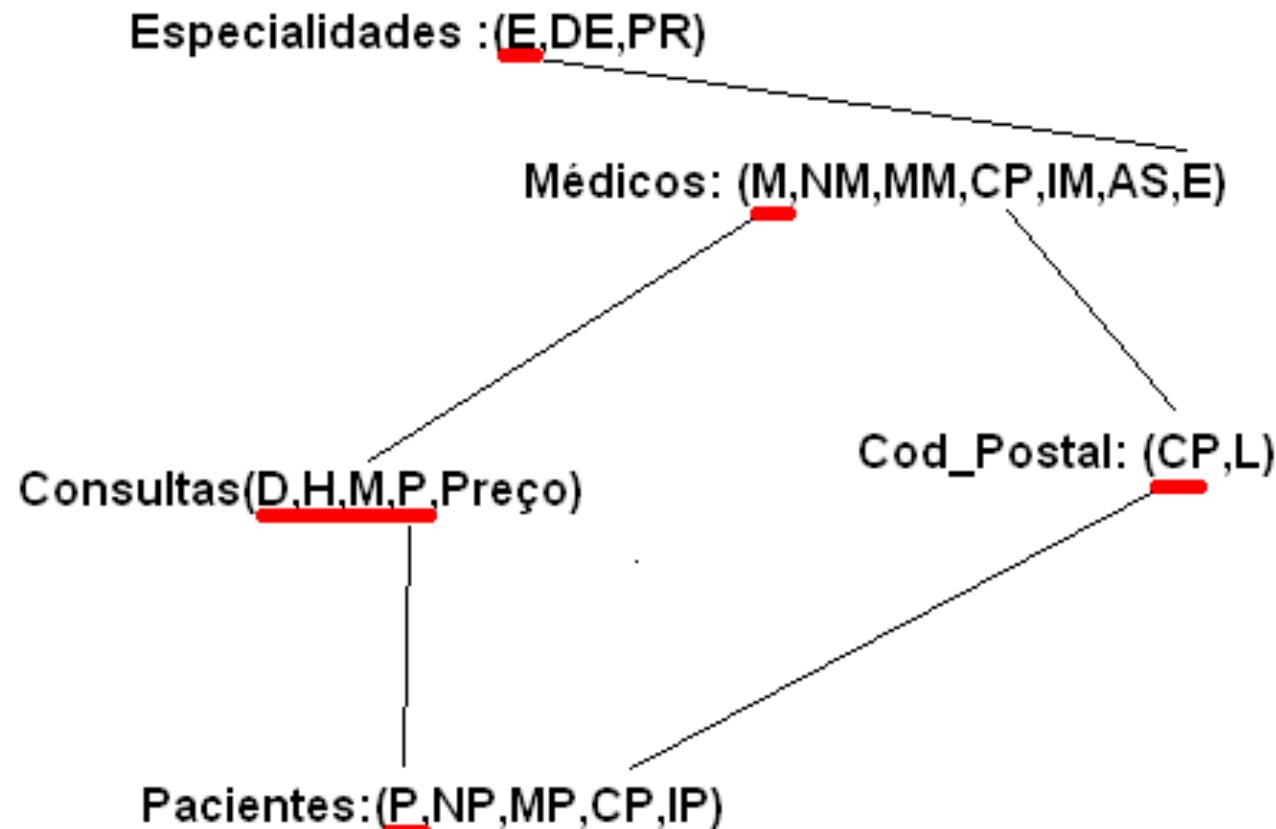


3^a Forma Normal

- A solução passa novamente por decompor a relação de acordo com as dependências funcionais anteriores:
 - Pacientes(n_ben,paciente,idade,morada,cod_post)
 - Cod_Postal(cod_post,localidade)
- Os problemas detectados estão agora resolvidos, a base de dados está na 3^a FN.



Exemplo : Uma Clínica



Normalização

O exemplo seguinte ilustra o ganho em volume de informação com o processo de normalização.

É dado o registo R0 com a informação não normalizada:

R0: médicos(cm, nm, md, im, as, ce, de, pr, cd, nd, md, id, da, ho)

Será usado um exemplo considerando a cardinalidade de cada conjunto de entidade:

- 50 médicos
- 10000 pacientes
- 6 consultas diárias por médico
- 10 especialidades
- 1000 dias



Normalização

Col.	Tamanho (em bytes)		Col.	Tamanho (em bytes)
cm	2		cd	2
nm	40		nd	40
mm	50		md	50
im	2		id	2
as	2		da	8
ce	2		ho	5
de	40			
pr	4			



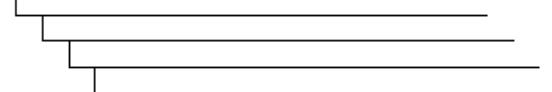
Normalização

Tamanho inicial do registo : 249 bytes

Cardinalidade : $50 * 6 * 1000 = 300\,000$ registos

Tamanho : $300\,000 * 249 = 74\,700\,000$ b = **71.24 Mb**

médicos(cm,nm,md,im,as,ce,de,pr,cd,nd,md,id,da,ho)



1º FN

médicos(cm,nm,mm,im,as,ce,de,pr)

consultas(cm,cd,nd,md,id,da,ho)



2º FN

médicos(cm,nm,mm,im,as,ce,de,pr)

consultas(cm,cd,da,ho)

pacientes(cd,nd,md,id)

3º FN

médicos(cm,nm,mm,im,as,ce)

consultas(cm,cd,da,ho)

pacientes(cd,nd,md,id)

especialidades(ce,de,pr)



1º FN

consultas(cm,cd,nd,md,id,da,ho)

50 médicos

10000 pacientes

6 consultas diárias por médico

10 especialidades

1000 dias

Tamanho de registo de médicos : 142 b

Cardinalidade : 50

Tamanho : $50 * 142 = 7\,100\text{ b}$

Tamanho de registo de consultas : 109 b

Cardinalidade : $50 * 6 * 1000 = 300\,000$

Tamanho : $300\,000 * 109 = 32\,700\,000\text{ b}$

Total : 31.19 Mb



2º FN

médicos(cm,nm,mm,im,as,ce,de,pr)

consultas(cm,cd,da,ho)

pacientes(cd,nd,md,id)

50 médicos, 10000 pacientes, 6 consultas diárias por
médico, 10 especialidades e 1000 dias

Tamanho de registo de médicos : 142 b

Cardinalidade : 50

Tamanho : $50 * 142 = 7\,100$ b

Tamanho de registo de consultas : 17 b

Cardinalidade : $50 * 6 * 1000 = 300\,000$

Tamanho : $300\,000 * 17 = 5\,100\,000$ b

Tamanho de registo de pacientes : 94 b

Cardinalidade : 10 000

Tamanho : $10\,000 * 94 = 940\,000$ b

Total : 5.77 Mb



3º FN

médicos(cm,nm,mm,im,as,ce)

consultas(cm,cd,da,ho)

pacientes(cd,nd,md,id)

especialidades(ce,de,pr)

50 médicos, 10000 pacientes, 6 consultas diárias por médico, 10 especialidades e 1000 dias

Tamanho de registo de médicos : 98 b

Cardinalidade : 50

Tamanho : $50 * 98 = 4\,900$ b

Tamanho de registo de consultas : 17 b

Cardinalidade : $50 * 6 * 1000 = 300\,000$

Tamanho : $300\,000 * 17 = 5\,100\,000$ b

Tamanho de registo de pacientes : 94 b

Cardinalidade : 10 000

Tamanho : $10\,000 * 94 = 940\,000$ b

Tamanho de registo de especialidades : 46 b

Cardinalidade : 10

Tamanho : $10 * 46 = 460$ b

Total : 5.77 Mb



Resultados

Fase	Espaço (em Bytes)	Espaço (em Mbytes)	Ganho Total	Ganho entre fases
Início	74 700 000	71.24		
1ª FN	32 707 100	31.19	56.22%	56.22%
2ª FN	6 047 100	5.77	91.90%	81.51%
3ª FN	6 045 360	5.76	91.91%	0.03%

50 médicos, 10000 pacientes, 6 consultas diárias por médico, 1000 dias



Forma Normal de Boyce Codd

$(\text{paciente}, \text{cod_serv}) \rightarrow \text{medico}$

Representada na relação: **R**(paciente, cod_serv, medico)

No entanto um médico pertence a um só serviço, dependência que não está a ser atendida. Uma solução poderia passar por decompor **R** em duas relações:

R1(paciente, medico) e **R2**(medico, cod_serv).

Estas estão sim na **BCFN**.

Esta solução introduz outro problema: é possível registrar dois médicos do mesmo serviço para o mesmo paciente!

(Esta dependência deverá ser tratada ao nível aplicacional!)

Ou então manter R juntamente com R2



Forma Normal de Boyce Codd

Definição : Uma relação está na BCNF se todos os atributos são funcionalmente dependentes da chave, de toda a chave e nada mais de que a chave.

- A 3^a FN é aquela em que, na maioria dos casos, termina o processo de normalização, em alguns casos a 3^aFN ainda transporta algumas anomalias que são resolvidas pela BCNF.
- *Conderemos que no sentido de prestar um melhor serviço aos pacientes, para cada serviço o paciente é sempre atendido pelo mesmo médico.* Daqui temos a seguinte dependência funcional.



Dependências Multivalor

Por definição, dada uma relação $R(X,Y,Z)$ diz-se que existe uma dependência multivalor $X \rightarrow\!\!\!> Y$ (X multidetermina Y) se, para cada par de tuplos de R contendo os mesmos valores de X também existe em R um par de tuplos correspondentes à troca dos valores de Y no par original.

Por simples análise dos tuplos presentes numa relação não é possível concluir da existência de uma dependência multivalor, a não ser que estas representem todos os casos admissíveis de relacionamento entre os dados. Contudo, a mesma análise permite concluir da inexistência de uma dependência multivalor.



Dependências Multivalor

X	Y	Z
x1	y1	z1
x1	y1	z2
x1	y2	z1
x1	y2	z2
x3	y1	z1
x4	y3	z2

Existem 2 dependências multivalor $X \rightarrow\!\!> Y$ e $X \rightarrow\!\!> Z$, contudo basta que se retire, por exemplo, o primeiro tuplo da relação para que deixem de se verificar as duas dependências multivalor.

Por observação das instâncias de uma relação pode-se concluir da não existência de uma dependência multivalor. Essa observação já não é suficiente para concluir da sua existência. Para isso, terão de se conhecer essas regras que governam a manipulação dessa relação. Parece evidente que só em situações muito específicas é que surgem dependências multivalor.

Ex: R(Agente, Produto, Zona) regra : Todos os agentes vendem todos os produtos que representam em todas as zonas em que actuam

(Agente \rightarrow Produto e Agente \rightarrow Zona)

Solução : R1(Agente, Produto) e R2(Agente, Zona)



Dependências de Junção

X	Y	Z
x1	y1	z1
x1	y1	z2
x1	y2	z2
x2	y3	z2
x2	y4	z2
x2	y4	z4
x2	y5	z4
x3	y2	z5

Projectando em (X,Y), (X,Z) e (Y,Z) não é possível reconstruir a relação R por junção de qualquer destas relações.

Apenas se consegue reconstruir R por junção das 3 projecções. A relação não possui uma dependência de junção (5^a Forma Normal)



O Comando **SELECT**

O comando **SELECT** é usado para interrogar a base de dados e recuperar os dados selecionados e que verificam as condições especificadas.

```
SELECT "coluna1"[,"coluna2",etc]  
      FROM "nometabela"  
      [WHERE "condição"];
```

[] = *opcional*

Os nomes das colunas que seguem a palavra SELECT determinam as colunas a apresentar como resultado. Pode-se selecionar um número variável de colunas ou usar um metacaracter "" para selecionar todas as colunas.

*O nome da tabela após a palavra **FROM** especifica a tabela a interrogar.

*A cláusula **WHERE** (opcional) especifica que valores ou linhas devem ser apresentados, baseando-se na condição descrita após a palavra **WHERE**.



Operadores

Os seguintes operadores podem ser numa cláusula **WHERE** :

- = Igual
- > Maior que
- < Menor que
- >= Maior ou igual
- <= Menor ou igual
- <> Diferente
- LIKE



LIKE

Like é um operador muito versátil que permite selecionar todas as linhas que são parecidas com uma dada sequência. O símbolo de percentagem "%" pode ser usado como um metacaracter que substituí qualquer sequência de caracteres.

```
SELECT primeiro, ultimo, localidade  
FROM empregado  
WHERE primeiro LIKE 'Lu%';
```

O comando seleciona qualquer atribuído primeiro que comece por "Er". Os valores de strings devem estar entre "".



LIKE

```
SELECT primeiro, ultimo  
FROM empregado  
WHERE ultimo LIKE '%s';
```

O comando seleciona qualquer atribuído *primeiro* que termine com "s".

```
SELECT * FROM empregado  
WHERE primeiro = 'Luis';
```

Este comando apenas seleciona a linha cuja atributo *primeiro* é igual a Luis.



EXEMPLO

SELECT primeiro, ultimo, localidade FROM empregado;

*SELECT ultimo, localidade, idade FROM empregado
WHERE idade > 30;*

*SELECT primeiro, ultimo, localidade, estado FROM empregado
WHERE primeiro LIKE 'J%';*

*SELECT * FROM empregado;*

*SELECT primeiro, ultimo, FROM empregado
WHERE último LIKE '%os';*

*SELECT primeiro, ultimo, idade FROM empregado
WHERE último LIKE '%oui%';*

*SELECT * FROM empregado WHERE primeiro = 'Luis';*



O Comando CREATE TABLE

O comando “CREATE TABLE” :

```
CREATE TABLE "nometabela"
("coluna1" "data type",
"coluna2" "data type",
"coluna3" "data type");
```

no caso de usar restrições opcionais:

```
CREATE TABLE "nometabela"
("coluna1" "data type" [restrição],
"coluna2" "data type" [restrição],
"coluna3" "data type" [restrição]);
[ ] = opcional
```

Exemplo:

```
CREATE TABLE empregado
(primeiro varchar(15),
ultimo varchar(20),
idade número(3),
morada varchar(30),
localidade varchar(20),
estado varchar(20));
```



Tipos de Dados

Os tipos de dados especificam o tipo dos valores a armazenar numa coluna. Se uma coluna chamada "Último_Nome" vai suportar nomes, então o tipo deve ser "varchar" (caracter de comprimento variável).

char(tamanho) : String de comprimento fixo. O tamanho esta especificado entre parênteses. Max 255 bytes.

varchar(tamanho) : String de comprimento variável. O tamanho máximo está especificado entre parênteses.

número(tamanho) : Valor numérico inteiro com um número máximo de dígitos “tamanho”

data : Valor de data

número(tamanho,d) : Valor numérico com um número máximo de dígitos “tamanho” e um número nmáximo de casas decimais “d”.



Restrições

Quando as tabelas são criadas, é normal associar a uma ou mais colunas algumas restrições.

Uma restrição é basicamente uma regra associada a uma coluna que deve ser respeitada por todos os dados que forem armazenados nessa coluna.

Por exemplo, a restrição "unique" especifica que não é possível dois registo diferentes terem essa coluna com o mesmo valor.

As outras duas restrições mais populares são o "not null" que especifica que uma coluna não pode ter o valor nulo, e "primary key".

A restrição "primary key" define uma identificação única para cada registo.



O Comando DROP TABLE

O comando DROP TABLE é usado para remover a tabela e a respectiva informação:

DROP TABLE "nometabela"

Exemplo:

DROP TABLE empregado;

Exercício:

Elimine a tabela empregado.



Exemplos

```
drop table student cascade constraints;  
drop table faculty cascade constraints;  
drop table class cascade constraints;  
drop table enrolled cascade constraints;  
drop table emp cascade constraints;  
drop table works cascade constraints;  
drop table dept cascade constraints;  
drop table flights cascade constraints;  
drop table aircraft cascade constraints;  
drop table certified cascade constraints;  
drop table employees cascade constraints;  
drop table suppliers cascade constraints;  
drop table parts cascade constraints;  
drop table catalog cascade constraints;  
drop table sailors cascade constraints;
```



Exemplos

```

create table student(
  snum number(9,0) primary key,
  sname varchar2(30),
  major varchar2(25),
  standing varchar2(2),
  age number(3,0) );
create table faculty(...);
create table class(
  name varchar2(40) primary key,
  meets_at varchar2(20),
  room varchar2(10),
  fid number(9,0),
  foreign key(fid) references faculty );
create table enrolled(...);
create table emp(...);
create table dept(...);
create table works(...);

```

```

create table flights(
  flno number(4,0) primary key,
  origin varchar2(20),
  destination varchar2(20),
  distance number(6,0),
  departs date,
  arrives date,
  price number(7,2) );
create table aircraft(...);
create table employees(...);
create table certified(...);
create table suppliers(...);
create table parts(...);
create table catalog(...);
create table sailors(
  sid number(9,0) primary key,
  sname varchar2(30),
  rating number(2,0),
  age number(4,1) );

```



O Comando INSERT

O comando **INSERT** é usado para inserir ou adicionar dados a uma tabela.

```
INSERT INTO "nometabela"  
[(primeira_coluna,...última_coluna)]  
VALUES (primeira_valor,...última_valor);  
[ ] = opcional
```

Exemplo:

```
INSERT INTO empregado  
(primeiro, ultimo, idade, morada, localidade, estado)  
VALUES ('Luis', 'Duque', 45, 'Rua dos Bragas, 20', 'Braga', 'Minho');
```



O Comando UPDATE

O comando **UPDATE** é usado para actualizar regtos:

UPDATE "nometabela"

SET "nomecoluna" = "novovalor"[,"proxcoluna" = "novovalor2"...]

WHERE "nomecoluna" OPERADOR "valor" [and|or "coluna"

OPERADOR "valor"];

[] = opcional

Exemplos:

UPDATE agenda SET indicativo = "253"

WHERE localidade = "Braga";

UPDATE agenda

SET ultimo_nome = 'Ferreira', indicativo="253"

WHERE ultimo_nome = 'Fereira';

UPDATE empregado

SET idade = idade+1

WHERE primeiro_nome='Maria' and último_nome='Pereira';

1. Joana Ferreira casou-se com Paulo Gomes. O seu último nome mudou para Ferreira Gomes.
2. Daniel Santos faz anos hoje. Vai ficar um ano mais velho.
3. Todas as secretárias serão agora intituladas “Assistentes Administrativas”.
4. Todos aqueles que ganham menos de 30000 vão ter um aumento de 3500.
5. Todos aqueles que ganham mais de 33500 irão ter um aumento de 4500.
6. Todos aqueles que auferem o título de "Programador II" vão ser promovidos a "Programador III".
7. Todos aqueles que auferem o título de "Programador" vão ser promovidos a "Programador II".



O Comando DELETE

O comando **DELETE** é usado para apagar / eliminar registos:

```
DELETE FROM "nometabela"
WHERE "nomecoluna" OPERADOR "valor" [and|or "coluna"
OPERADOR "valor"];
[ ] = opcional
```

Exemplos:

```
DELETE FROM empregado;
```

Nota: **todos os registos são removidos!**

```
DELETE FROM empregado
WHERE últimonome = 'Neves';
```

```
DELETE FROM empregado
WHERE primeironome = 'Miguel' or primeironome = 'Manuel';
```

Exercício (use o comando SELECT para verificar as eliminações):

1. A Joana Ferreira Gomes foi embora;
2. É tempo de poupar. Os empregados que ganham mais de 7000 vão ser despedidos



O SELECT (conceitos avançados)

O comando SELECT tem a seguinte sintaxe:

```
SELECT [ALL | DISTINCT] coluna1  
[,coluna2]  
FROM table1[,table2]  
[WHERE "condições"]  
[[GROUP BY "coluna-lista"]]  
[HAVING "condições"]  
[ORDER BY "coluna-lista" [ASC |  
DESC] ]
```

As palavras reservadas **ALL** e **DISTINCT** são usadas para SELEC(T)ionar todos (ALL) ou os valores sem repetições.

Por exemplo:

```
SELECT DISTINCT idade  
FROM empregado_info;
```



GROUP BY

```
SELECT coluna-lista, SUM(coluna2)
FROM "lista-de-tabelas"
GROUP BY "coluna-lista";
```

```
SELECT max(Vencimento), dept
FROM empregado
GROUP BY dept;
```

```
SELECT quantidade, max(preço)
FROM artigos_encomendados
GROUP BY quantidade;
```



HAVING

```
SELECT coluna1, SUM(coluna2)
FROM "lista-de-tabelas"
GROUP BY "colunal"
HAVING "condição";
```

```
SELECT dept, avg(Vencimento)
FROM empregado
GROUP BY dept;
```

```
SELECT dept, avg(Vencimento)
FROM empregado
GROUP BY dept
HAVING avg(Vencimento) > 20000;
```



ORDER BY

```
SELECT coluna1, SUM(coluna2)
FROM "lista-de-tabelas"
ORDER BY "coluna-lista" [ASC | DESC];
[ ] = opcional
ASC = Ordem ascendente - defeito
DESC = Ordem descendente
```

Exemplo:

```
SELECT empregado_id, dept,
nome, idade, Vencimento
FROM empregado_info
WHERE dept = 'Vendas'
ORDER BY vencimento;
```

```
SELECT empregado_id, dept,
nome, idade, Vencimento
FROM empregado_info
WHERE dept = 'Vendas'
ORDER BY vencimento, idade
DESC;
```



Funções de Agregação

MIN	retorna o valor mais pequeno duma dada coluna
MAX	retorna o valor maior duma dada coluna
AVG	Retorna a média de todos os valores numéricos duma coluna
COUNT	Retorna a cardinalidade do conjunto de todos os registos
COUNT(*)	Retorna a cardinalidade da tabela

```
SELECT AVG(Vencimento)
FROM empregado;
```

```
SELECT AVG(Vencimento)
FROM empregado
WHERE Cargo = 'Programador';
```

```
SELECT Count(*)
FROM empregados;
```



Condições e Operadores Lógicos

Exemplo:

```
SELECT coluna1, SUM (coluna2)
FROM "lista-de-
tabelas"
WHERE "condição1"
AND "condição2";
```

```
SELECT empregadoid, primeironome, últimonome, Cargo,
Vencimento
FROM empregado_info
WHERE vencimento >= 50000.00 AND cargo =
'Programador';
```

```
SELECT empregadoid, primeironome, ultimonome, cargo,
vencimento
FROM empregado_info
WHERE (vencimento >= 50000.00) AND (cargo =
'Programador');
```

```
SELECT primeironome, últimonome, Cargo, vencimento
FROM empregado_info
WHERE (cargo = 'Vendas') OR (cargo = 'Programador');
```



IN e BETWEEN

Exemplo:

```
SELECT coluna1, SUM(coluna2)
FROM "lista-de-tabelas" WHERE
coluna3 IN (lista-de-valores);
```

```
SELECT coluna1, SUM(coluna2)
FROM "lista-de-tabelas" WHERE
coluna3 BETWEEN valor1 AND
valor2;
```

```
SELECT empregadoid, ultimonome, Vencimento
FROM empregado_info
WHERE ultimonome IN ('Hernandez', 'Jones', 'Roberts',
'Ruiz');
```

```
SELECT empregadoid, ultimonome, Vencimento
FROM empregado_info
WHERE ultimonome = 'Hernandez' OR ultimonome =
'Jones' OR ultimonome = 'Roberts' OR ultimonome =
'Ruiz';
```

```
SELECT empregadoid, idade, ultimonome, Vencimento
FROM empregado_info
WHERE idade BETWEEN 30 AND 40;
```

```
SELECT empregadoid, idade, ultimonome, Vencimento
FROM empregado_info
WHERE idade >= 30 AND idade <= 40;
```



FUNÇÕES

+	Adição	ABS(x)	retorna o valor absoluto de x
-	Subtração	SIGN(x)	retorna o sinal de x (-1, 0, ou 1) (negativo, zero, ou positivo)
*	Multiplicação	MOD(x,y)	módulo - retorna o resto da divisão inteira de x por (x%y)
/	Divisão	FLOOR(x)	retorna o maior inteiro que é menor ou igual a x
		CEILING(x) or CEIL(x)	retorna o menor inteiro que é maior ou igual a x
%	Módulo	POWER(x,y)	retorna o valor de x elevado à potência de y
		ROUND(x)	retorna o valor de x arredonda ao inteiro mais próximo
		ROUND(x,d)	retorna o valor de x arredondado o número com d casas decimais mais próximo
		SQRT(x)	retorna a raíz quadrada de x



EXEMPLO

```
SELECT round(vencimento), primeironome FROM empregado_info
```

Este comando seleciona o vencimento arredondado ao inteiro mais próximo e o primeironome do empregado.



JUNÇÃO - Introdução

Todas as questões colocadas até agora tem uma ligeira limitação. O comando SELECT aplica-se a apenas uma tabela. É tempo de introduzir agora uma das características mais importantes das bases de dados relacionais – o “join”, o operador que torna os sistemas de bases de dados relacionais “relacionais”.

As junções permitem ligar dados de duas ou mais tabelas, de forma a obter uma única tabela como resultado. Uma “junção” reconhece-se num SELECT que houver mais de uma tabela referida na cláusula FROM.

Exemplo:

```
SELECT "lista-de-colunas"  
FROM table1,table2  
WHERE "condição(ões)_pesquisa"
```



Agora, sempre que um cliente já registado efectuar uma compra, apenas a segunda tabela “vendas” necessita de ser actualizada. Os dados redundantes foram assim eliminados e a base de dados está normalizada. Nota-se que ambas as tabelas tem uma coluna comum "cliente_número" coluna. Essa coluna, contem o número único do cliente será usada para juntar as duas tabelas. Usando as duas tabelas, se se quiser saber o nome dos clientes e os artigos que compraram, o seguinte comando pode resolver essa questão:

```
SELECT cliente_info.primeironome, cliente_info.ultimonome, vendas.artigo
FROM cliente_info, vendas
WHERE cliente_info.cliente_numero = vendas.cliente_numero;
```

Esta junção particular é conhecida por "Inner Join" ou "Equijoin". É o tipo mais conhecida de junção. Nota-se que o identificador de cada coluna é precedido pelo nome da tabela e um ponto. Não é obrigatório, mas é um bom hábito fazê-lo para que em caso de ambiguidade se saiba o significado das colunas e a que tabelas elas pertencem. Por exemplo é obrigatório quando as colunas comum têm o mesmo nome.

```
SELECT cliente_info.primeironome, cliente_info.ultimonome, vendas.artigo
FROM cliente_info INNER JOIN vendas
ON cliente_info.cliente_número = vendas.cliente_numero;
```



O modelo relacional

Uma base de dados relaciona um conjunto de tabelas na forma R (A_1, A_2, \dots, A_r), com grau r.

A cardinalidade é o número de elementos de R.

O conjunto dos possíveis valores de um atributo é o domínio $\text{dom}(A_i) = D_i$ (e.g., o domínio dum código é o conjunto dos números positivos e o domínio da idade de um trabalhador activo é o conjunto de números inteiros $\{18, \dots, 65\}$).

Regra 1: não pode haver 2 tuplos idênticos na mesma relação.

Regra 2: não existe uma ordem pré-definida numa relação.



Álgebra Relacional

UNIÃO

$$\begin{array}{|c|c|c|} \hline A & B & C \\ \hline a1 & b2 & c1 \\ \hline a5 & b1 & c2 \\ \hline a2 & b4 & c4 \\ \hline a3 & b3 & c3 \\ \hline \end{array} \cup
 \begin{array}{|c|c|c|} \hline A & B & C \\ \hline a2 & b3 & c2 \\ \hline a1 & b2 & c1 \\ \hline a2 & b4 & c4 \\ \hline \end{array}
 =
 \begin{array}{|c|c|c|} \hline A & B & C \\ \hline a1 & b2 & c1 \\ \hline a5 & b1 & c2 \\ \hline a2 & b4 & c4 \\ \hline a3 & b3 & c3 \\ \hline a2 & b3 & c2 \\ \hline \end{array}$$

INTERSECÇÃO

$$\begin{array}{|c|c|c|} \hline A & B & C \\ \hline a1 & b2 & c1 \\ \hline a5 & b1 & c2 \\ \hline a2 & b4 & c4 \\ \hline \end{array} \cap
 \begin{array}{|c|c|c|} \hline A & B & C \\ \hline a2 & b3 & c2 \\ \hline a1 & b2 & c1 \\ \hline a2 & b4 & c4 \\ \hline \end{array}
 =
 \begin{array}{|c|c|c|} \hline A & B & C \\ \hline a1 & b2 & c1 \\ \hline a2 & b4 & c4 \\ \hline \end{array}$$

DIFERENÇA

$$\begin{array}{|c|c|c|} \hline A & B & C \\ \hline a1 & b2 & c1 \\ \hline a5 & b1 & c2 \\ \hline a2 & b4 & c4 \\ \hline \end{array} -
 \begin{array}{|c|c|c|} \hline A & B & C \\ \hline a2 & b3 & c2 \\ \hline a1 & b2 & c1 \\ \hline a2 & b4 & c4 \\ \hline \end{array}
 =
 \begin{array}{|c|c|c|} \hline A & B & C \\ \hline a5 & b1 & c2 \\ \hline \end{array}$$


Álgebra Relacional

PRODUTO CARTESIANO

$$\begin{array}{|c|c|} \hline \text{A} & \text{B} \\ \hline \text{a1} & \text{b2} \\ \hline \text{a5} & \text{b1} \\ \hline \text{a2} & \text{b4} \\ \hline \end{array} \times \begin{array}{|c|c|} \hline \text{C} & \text{D} \\ \hline \text{c2} & \text{d3} \\ \hline \text{c1} & \text{d2} \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline \text{A} & \text{B} & \text{C} & \text{D} \\ \hline \text{a1} & \text{b2} & \text{c2} & \text{d3} \\ \hline \text{a1} & \text{b2} & \text{c1} & \text{d2} \\ \hline \text{a5} & \text{b1} & \text{c2} & \text{d3} \\ \hline \text{a5} & \text{b1} & \text{c1} & \text{d2} \\ \hline \text{a2} & \text{b4} & \text{c2} & \text{d3} \\ \hline \text{a2} & \text{b4} & \text{c1} & \text{d2} \\ \hline \end{array}$$

PROJEÇÃO

$$\pi_{(\text{A}, \text{C})} \begin{array}{|c|c|c|} \hline \text{A} & \text{B} & \text{C} \\ \hline \text{a2} & \text{b3} & \text{c4} \\ \hline \text{a1} & \text{b2} & \text{c1} \\ \hline \text{a2} & \text{b4} & \text{c4} \\ \hline \end{array} = \begin{array}{|c|c|} \hline \text{A} & \text{C} \\ \hline \text{a2} & \text{c4} \\ \hline \text{a1} & \text{c1} \\ \hline \end{array}$$

SELECCÃO

$$\sigma_{(\text{A} = \text{a2})} \begin{array}{|c|c|c|} \hline \text{A} & \text{B} & \text{C} \\ \hline \text{a2} & \text{b3} & \text{c2} \\ \hline \text{a1} & \text{b2} & \text{c1} \\ \hline \text{a2} & \text{b4} & \text{c4} \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline \text{A} & \text{B} & \text{C} \\ \hline \text{a2} & \text{b3} & \text{c2} \\ \hline \text{a2} & \text{b4} & \text{c4} \\ \hline \end{array}$$

JUNÇÃO

$$\begin{array}{|c|c|c|} \hline \text{A} & \text{B} & \text{C} \\ \hline \text{a1} & \text{b2} & \text{c1} \\ \hline \text{a5} & \text{b1} & \text{c2} \\ \hline \text{a2} & \text{b4} & \text{c4} \\ \hline \end{array} \bowtie \begin{array}{|c|c|} \hline \text{C} & \text{D} \\ \hline \text{c2} & \text{d3} \\ \hline \text{c1} & \text{d2} \\ \hline \text{c2} & \text{d1} \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline \text{A} & \text{B} & \text{C} & \text{D} \\ \hline \text{a1} & \text{b2} & \text{c1} & \text{d2} \\ \hline \text{a5} & \text{b1} & \text{c2} & \text{d3} \\ \hline \text{a5} & \text{b1} & \text{c2} & \text{d1} \\ \hline \end{array}$$


Álgebra Relacional

DIVISÃO

$$\begin{array}{|c|c|c|c|} \hline
 \textbf{A} & \textbf{B} & \textbf{C} & \textbf{D} \\ \hline
 \text{a1} & \text{b2} & \text{c2} & \text{d3} \\ \hline
 \text{a5} & \text{b1} & \text{c1} & \text{d2} \\ \hline
 \text{a2} & \text{b4} & \text{c1} & \text{d2} \\ \hline
 \text{a3} & \text{b5} & \text{c4} & \text{d4} \\ \hline
 \text{a2} & \text{b4} & \text{c2} & \text{d3} \\ \hline
 \text{a1} & \text{b2} & \text{c1} & \text{d2} \\ \hline
 \end{array} \div \begin{array}{|c|c|} \hline
 \textbf{C} & \textbf{D} \\ \hline
 \text{c2} & \text{d3} \\ \hline
 \text{c1} & \text{d2} \\ \hline
 \end{array} = \begin{array}{|c|c|} \hline
 \textbf{A} & \textbf{B} \\ \hline
 \text{a1} & \text{b2} \\ \hline
 \text{a2} & \text{b4} \\ \hline
 \end{array}$$


Álgebra Relacional

Operações básicas :

$R \cup S$

$R - S$

$R \times S$

$\Pi_{A_i, \dots, A_j}(R) = \Pi_{i, \dots, j}(R)$

$\sigma_{\text{Cond}}(R)$

Operações adicionais:

$R \cap S$

$R \div S$

$R \bowtie_{i \Theta j} S$

(Θ -junção e.g., $<$ -junção)

$R \bowtie S$

$R \bowtie S$



Equivalências

a) $R \cap S = R - (R - S)$

b) $R(A_1, \dots, A_r) S(A_{r-s+1}, \dots, A_s) r > s \text{ e } S \neq \emptyset$

$$R \div S = \Pi_{1, \dots, r-s}(R) - \Pi_{1, \dots, r-s}((\Pi_{1, \dots, r-s}(R) \times S) - R)$$

c) $R \bowtie_{i \Theta j} S = \sigma_{\$i \Theta \$r+j}(R \times S)$

d) $R \bowtie S = \Pi_{i=1, \dots, im}(\sigma_{R.A_i=S.A_i \text{ e } \dots \text{ e } R.A_k=S.A_k}(R \times S))$

e) $R \ltimes S = \Pi_R(R \bowtie S)$



Bases de Dados Dedutivas

$R(A_1, \dots, A_r) \quad S(A_1, \dots, A_r) \quad T(A_1, \dots, A_r)$

$R \cup S = T$

$t(A_1, \dots, A_r) \leftarrow r(A_1, \dots, A_r).$ ou $t(A_1, \dots, A_r) \leftarrow r(A_1, \dots, A_r) \vee s(A_1, \dots, A_r).$

$t(A_1, \dots, A_r) \leftarrow s(A_1, \dots, A_r).$

$R - S = T$

$t(A_1, \dots, A_r) \leftarrow r(A_1, \dots, A_r) \wedge \neg s(A_1, \dots, A_r).$

$R(A_1, \dots, A_r) \quad S(A_{r+1}, \dots, A_{r+s}) \quad T(A_1, \dots, A_{r+s}) \quad T = R \times S$

$t(A_1, \dots, A_{r+s}) \leftarrow r(A_1, \dots, A_r) \wedge s(A_{r+1}, \dots, A_{r+s}).$

$R(A_1, \dots, A_r) \quad T(A_i, \dots, A_j) \quad i \leq r, j \leq r \quad \Pi_{A_i, \dots, A_j}(R) = T$

$t(A_i, \dots, A_j) \leftarrow r(A_1, \dots, A_r).$

$R(A_1, \dots, A_r) \quad \sigma_{\text{Cond}}(R) = T$

$t(A_1, \dots, A_r) \leftarrow r(A_1, \dots, A_r) \wedge \text{Cond}.$



Bases de Dados Dedutivas

$R(A_1, \dots, A_r) S(A_1, \dots, A_r) T(A_1, \dots, A_r)$

$R \cap S = T \quad t(A_1, \dots, A_r) \leftarrow r(A_1, \dots, A_r) \wedge s(A_1, \dots, A_r).$

$R - (R - S) = T \quad t(A_1, \dots, A_r) \leftarrow r(A_1, \dots, A_r) \wedge \neg(r(A_1, \dots, A_r) \wedge \neg s(A_1, \dots, A_r)).$

$R - (R - S) = R \cap S \quad A \wedge \neg(A \wedge \neg B) = A \wedge (\neg A \vee B) = A \wedge B$

$R \div S = T \quad R(A_1, \dots, A_r) S(A_{r-s+1}, \dots, A_r) T(A_1, \dots, A_{r-s}) \quad R \div S = \Pi_{1, \dots, r-s}(R) - \Pi_{1, \dots, r-s}((\Pi_{1, \dots, r-s}(R) \times S) - R)$

$r1(A_1, \dots, A_{r-s}) \leftarrow r(A_1, \dots, A_r). \quad R1 = \Pi_{1, \dots, r-s}(R)$

$r2(A_1, \dots, A_r) \leftarrow r1(A_1, \dots, A_{r-s}) \wedge s(A_{r-s+1}, \dots, A_r). \quad R2 = R1 \times S$

$r3(A_1, \dots, A_r) \leftarrow r2(A_1, \dots, A_r) \wedge \neg r(A_1, \dots, A_r). \quad R3 = R2 - R$

$r4(A_1, \dots, A_{r-s}) \leftarrow r3(A_1, \dots, A_r). \quad R4 = \Pi_{1, \dots, r-s}(R3)$

$t(A_1, \dots, A_{r-s}) \leftarrow r1(A_1, \dots, A_{r-s}) \wedge \neg r4(A_1, \dots, A_{r-s}). \quad T = R1 - R4$

$R \bowtie_{i \Theta j} S = T \quad t(A_1, \dots, A_{r+s}) \leftarrow r(A_1, \dots, A_r) \wedge s(A_{r+1}, \dots, A_{r+s}) \wedge A_i \Theta A_{r+j}.$

$R \bowtie S = T$ e k é o número de atributos comuns

$t(A_1, \dots, A_{r+s-k}) \leftarrow r(A_1, \dots, A_r) \wedge s(A_{r+1}, \dots, A_{r+s}) \wedge A_{i1} = A_{j1} \wedge \dots \wedge A_{ik} = A_{jk}.$



SQL

União : (select * from R) union (select * from S)

Diferença : (select * from R) minus(select * from S)

select * from R where R.* not in (select * from S)

Produto : select R.*,S.* from R,S

Projecção : select Ai,..,Aj from R

Selecção : select * from R where Cond

Intersecção : (select * from R) intersect (select * from S)

select * from R where R.* in (select * from S)

Θ -junção : select R.*,S.* from R,S where Ai Θ Ar+j

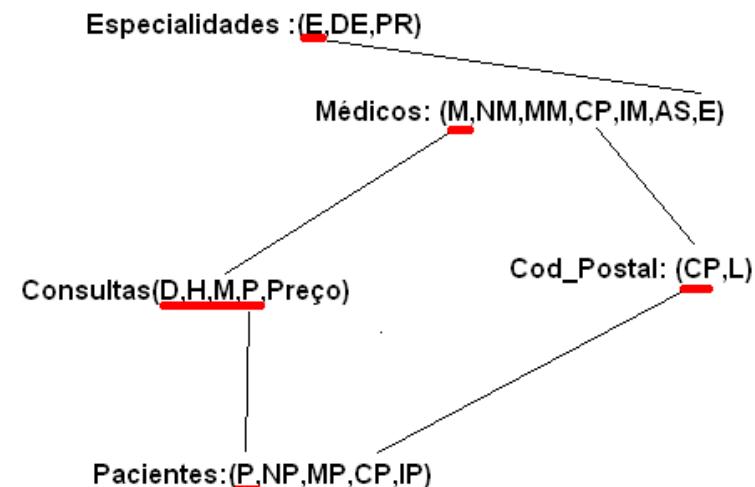
Junção : select R.* ,S.Ar+1,S.Ar+s-k from R,S where R.Ai1 = S.Ai1 and ...
and R.Aik = S.Aik

Divisão : (ver exemplo e aplicar $\forall x P(x) = \neg \exists x \neg P(x)$)



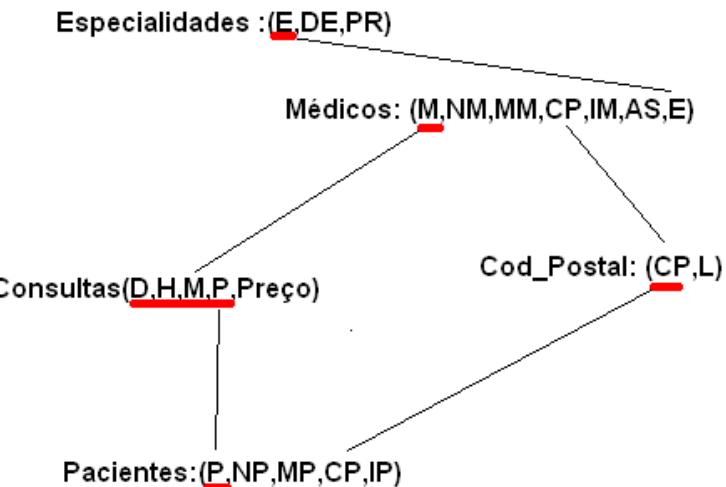
Exemplo – Clínica - Exercícios

1. Qual é o nome dos médicos com mais de 10 anos de serviço.
2. Indique o nome dos médicos e respectiva especialidade
3. Qual é o par nome e morada dos pacientes residentes em Braga.
4. Qual é o nome dos médicos da especialidade de oftalmologia.
5. Quais são os médicos com mais de 40 anos com a especialidade de Clínica Geral.
6. Quais são os médicos de Oftalmologia que consultaram pacientes de Braga.
7. Quais são os médicos com mais de 50 anos que deram consultas de tarde a pacientes com menos de 20 anos.
8. Quais são os pacientes com mais de 10 anos que nunca foram consultados a Oftalmologia.



Exemplo – Clínica - Exercícios

9. Quais são as especialidades consultadas no mês de Janeiro de 2002.
10. Qual é o nome dos médicos com mais de 30 anos ou menos de 5 anos de serviço.
11. Quais são os médicos de Clínica Geral que não consultaram em Janeiro de 2002.
12. Quais são os pacientes que já foram consultados por todos os médicos.
13. Quais são as especialidades que não foram consultadas durante os meses de Janeiro e Março de 2002.
14. Quais são os médicos que nunca consultaram pacientes de Braga.
15. Quais são os pacientes que só foram consultados a Clínica Geral.



Soluções

Qual é o nome dos médicos com mais de 10 anos de serviço?

Select NM from medicos where AS > 10

Indique o nome dos médicos e respectiva especialidade.

*Select NM,DE from especialidades,medicos
where especialidades.E = medicos.E*

Qual é o par (nome e morada) dos pacientes residentes em Braga?

*Select NP,MP from pacientes,cod_postal
where pacientes.cp = cod_postal.cp and cod_postal.L = "Braga"*

Qual é o nome dos médicos da especialidade de oftalmologia?

*Select NM from medicos,especialidades
where medicos.E = especialidades.E and especialidades.DE = "Oftalmologia"*



Soluções

Quais são os médicos com mais de 40 anos com a especialidade de Clínica Geral?

Select me. from medicos me,especialidades es
where me.E = es.E and me.IM > 40 and es.DE = "Clínica Geral"*

Quais são os médicos de Oftalmologia que consultaram pacientes de Braga?

Select me. from especialidades es,medicos me,consultas co, pacientes pa,
cod_postal cl where es.e = me.e and co.m = me.m and co.p = pa.p
and cl.cp = pa.cp and es.de = "Oftalmologia" and L = "Braga"*

Quais são os médicos com mais de 50 anos que deram consultas de tarde a pacientes com menos de 20 anos?

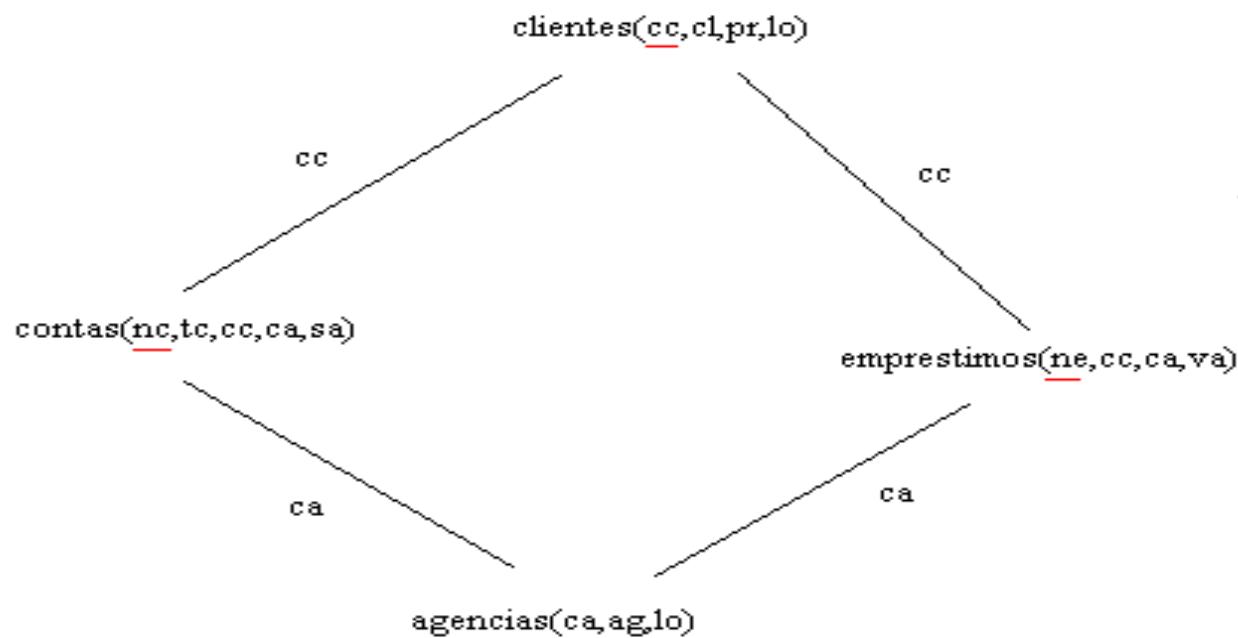
Select me. from medicos me,consultas co, pacientes pa
where me.m = co.m and co.p = pa.p and me.im > 50
and format(co.h,"hh:mi") > "12:00" and pa.ip < 20*

Quais os pacientes com mais de 10 anos que nunca foram consultados a Oftalmologia?

Select pa. from pacientes pa where pa.ip > 10 and not exists
(select * from consultas co,medicos me, especialidades es
where me.m = co.m and co.p = pa.p and es.de = "Oftalmologia")*



Exemplo – Um Banco



1. Quais os clientes deste banco?

*SELECT cc, cl FROM clientes SELECT * FROM clientes*

2. Quais os clientes que residem em Braga?

*SELECT * FROM clientes*

WHERE lo = "BRAGA"

*SELECT * FROM clientes*

WHERE lo like "%BRAGA%"

3. Quais os clientes com conta(s) na agência ca = 123?

SELECT DISTINCT cc FROM contas

WHERE ca = "123"

*SELECT DISTINCT c.**

FROM contas co,clientes c

WHERE c.cc = co.cc and co.ca = "123"

SELECT cc,cl FROM clientes c

*WHERE EXISTS (SELECT * FROM contas co*

WHERE co.cc = c.cc and co.ca = "123")



4. Quais os clientes que residem na mesma localidade das agências?

```
SELECT DISTINCT clientes.* FROM
clientes,agencias WHERE clientes.lo =
agencias.lo
```

5. Quais os clientes com empréstimos de valor superior a 500000 €?

```
SELECT DISTINCT c.* FROM clientes
c,emprestimos e WHERE c.cc = e.cc
AND e.va > 500000
```

```
SELECT DISTINCT c.* FROM clientes
c,emprestimos e WHERE c.cc = e.cc
AND e.va > 500000
```

6. Quais os nomes dos clientes com a mesma profissão que o cliente cc=1234?

```
SELECT DISTINCT c1.cl FROM clientes
c1,clientes c2 WHERE c1.pr = c2.pr AND
c2.cc = "1234"
```

7. Listar as contas da agência (ca=123) por ordem decrescente do valor do seu saldo?

8. Quantas contas existem em todas as agências do banco?

```
SELECT DISTINCT co.* FROM contas co WHERE
co.ca = "123" ORDER BY sa DESC
```

```
SELECT count(*) FROM contas
```



9. Quantos clientes possuem conta(s) na agência ca=123?

SELECT count(distinct cc) FROM contas WHERE ca = "123"

10. Listar o número de contas existentes em cada agência?

SELECT ca, count() FROM contas GROUP BY ca*

11. Para cada agência com menos de 1000 contas, listar os valores máximos e mínimos dos saldos dessas contas, assim como o saldo médio.

SELECT ca, max(sa), min(sa), avg(sa) FROM contas GROUP BY ca HAVING count() < 1000*



12. Quais os clientes cuja profissão é desconhecida?

```
SELECT * FROM clientes WHERE pr  
is null
```

13. Quais os clientes da agência ca = 123?

```
SELECT DISTINCT c.*  
FROM clientes c, contas co  
WHERE co.ca = "123" AND co.cc = c.cc  
UNION  
SELECT DISTINCT c.*  
FROM clientes c, emprestimos e  
WHERE e.ca = "123" AND e.cc = c.cc
```

```
SELECT DISTINCT c.*  
FROM clientes c, contas co, emprestimos e  
WHERE (co.ca = "123" AND co.cc = c.cc)  
OR (e.ca = "123" AND e.cc = c.cc)
```



14. Quais os clientes que são simultaneamente depositantes e devedores na agência ca = 123?

```
SELECT DISTINCT c.*  
FROM clientes c, contas co  
WHERE co.ca = "123" AND co.cc = c.cc  
INTERSECT  
SELECT DISTINCT c.*  
FROM clientes c, emprestimos e  
WHERE e.ca = "123" AND e.cc = c.cc
```

```
SELECT DISTINCT c.*  
FROM clientes c, contas co, emprestimos e  
WHERE (co.ca = "123" AND co.cc = c.cc)  
AND (e.ca = "123" AND e.cc = c.cc)
```

15. Quais os clientes que são apenas depositantes na agência ca = 123?

```
SELECT DISTINCT c.*  
FROM clientes c, contas co  
WHERE co.ca = "123" AND co.cc = c.cc  
MINUS  
SELECT DISTINCT c.*  
FROM clientes c, emprestimos e  
WHERE e.ca = "123" AND e.cc = c.cc
```

```
SELECT DISTINCT c.*  
FROM clientes c, contas co, emprestimos e  
WHERE (co.ca = "123" AND co.cc = c.cc)  
AND NOT (e.ca = "123" AND e.cc = c.cc)
```



16. Quais os clientes com, pelo menos, um empréstimo no banco?

```
SELECT c.* FROM clientes c
WHERE EXISTS (SELECT * FROM emprestimos e where e.cc = c.cc)
```

17. Quais as agências com depositantes residentes em Lisboa?

```
SELECT ag.* FROM agencias ag
WHERE ag.ca IN (SELECT co.ca FROM contas co where co.cc IN
(SELECT cc FROM clientes where lo = "Lisboa"))
```

18. Quais os clientes cujo saldo total das suas contas é superior a qualquer empréstimo contraído neste banco?

```
SELECT c.* FROM clientes c WHERE
(SELECT sum(sa) FROM contas co WHERE c.cc = co.cc)
>
(SELECT max(va) FROM emprestimos)
```



19. Quais os clientes que possuem contas em todas as agências do Porto?

```
SELECT c.* FROM clientes c WHERE
(SELECT count(DISTINCT co.ca) FROM contas co, agencias ag
WHERE co.cc = c.cc AND co.ca = ag.ca AND ag.lo = "PORTO")
=
(SELECT count(distinct ca) FROM agencias WHERE lo = "PORTO")
```

```
SELECT c.* FROM clientes c WHERE NOT EXISTS
(SELECT * FROM agencias ag WHERE ag.lo = "PORTO" AND NOT EXISTS
(SELECT * FROM contas co WHERE ag.cc = co.cc AND c.cc = co.cc))
```

20. Para cada cliente apresentar o seu saldo total.

```
SELECT cc,sum(sa) FROM contas GROUP BY cc
```

```
SELECT c.cc,c.cl,(SELECT sum(sa) FROM contas co where co.cc = c.cc)
FROM clientes c
```



Oracle

Informação sobre uma tabela

O comando DESCRIBE (ou DESC) dá a estrutura de uma tabela
 DESCRIBE *tabela*

Alteração da estrutura de uma tabela

Para alterar a estrutura de uma tabela deve-se usar o comando ALTER TABLE
 alter table *tabela* add(*nomecoluna tipo*, ...);
 alter table *tabela* modify (*coluna novotipo*);
 alter table drop *coluna*;

Eliminação de tabelas

Para eliminar uma tabela usa-se o comando DROP TABLE
 drop table *tabela*;

A seguinte script elimina todas as tabelas:

```
SET NEWPAGE 0
SET SPACE 0
SET LINESIZE 80
SET PAGESIZE 0
SET ECHO OFF
SET FEEDBACK OFF
SET HEADING OFF
SET MARKUP HTML OFF
SET ESCAPE \
SPOOL DELETME.SQL
select 'drop table ', table_name, 'cascade constraints \;' from user_tables;
SPOOL OFF
@DELETME
```



Oracle

Um exemplo de Sessão em SQLPlus

```
SQL> create table sign_sales(Color varchar2(30),date_sold date,  
2 price_each number);
```

Table created.

```
SQL> alter table students add( sign_shape number);
```

Table altered.

```
SQL> alter table students modify (sign_shape varchar2(10));
```

Table altered.

```
SQL> alter table students drop column sign_shape ;
```

Table altered.

```
SQL> DESCRIBE sign_shape
```

Name	Null?	Type
------	-------	------

COLOR		VARCHAR2(30)
-------	--	--------------

DATE SOLD		DATE
-----------	--	------

PRICE EACH		NUMBER
------------	--	--------

```
SQL> drop table sign_sale;
```

Table dropped.



Oracle

Dados

Inserção de dados

Insert into tabela values (umvalor, outrovalor, ...);
sendo *umvalor*, *outrovalor*, ... valores a inserir e *tabela* o nome da tabela.

Os valores são inseridos pela ordem das colunas na estrutura da tabela. O primeiro valor é inserido na primeira coluna e assim sucessivamente.

Interrogação

select nomecoluna, nomecoluna ... from tablename;
sendo *nomecoluna* o nome de uma coluna ou * todas as colunas da tabela.

Modificação de dados

update tabela set coluna=expressão where clausula;

Remoção de dados

delete from tabela where clausula;



Oracle

```
SQL> insert into signs values(19.95,'White','Rectangle','Park Somewhere Else');  
1 row created.
```

```
SQL>
```

```
SQL> select * from signs;
```

PRICE_EACH	COLOR	SHAPE	DESCRIPTION
19.95	White	Rectangle	Park Somewhere Else

```
SQL>
```

```
SQL> update signs set price_each = 4.00 where price_each < 20;
```

```
1 row updated
```

```
.SQL>
```

```
SQL>
```

```
delete from signs ;
```

```
1 row deleted.
```

```
SQL>
```



Oracle

Restrições

Adição de restrições

create table tabela (coluna tipo, coluna tipo ..., primary key(colunachave,colunachave,...);

sendo *colunachave* o nome de uma coluna que é parte da chave.

As chaves estrangeiras devem referir-se a tuplos únicos.

create table tabela (coluna tipo, coluna tipo...,
primary key(colunachave,colunachave,...),
foreign key(colunachaveestrangeira, colunachaveestrangeira,...) references tabelaestrangeira,
foreign key(colunachaveestrangeira, colunachaveestrangeira,...) references
tabelaestrangeira,...);

Alter table tablename add tableconstraint;



Oracle

Observação de restrições

Para ver as restrições associadas a uma tabela deve-se usar a "view" USER_CONSTRAINTS.

Execute DESCRIBE USER_CONSTRAINTS para mais informação.

```
select column_name, position, constraint_name from User_cons_columns;
```

Utilização de restrições

```
create table tabela ( coluna tipo, coluna tipo ...,
                     foreign key(colunachaveestrangeira,colunachaveestrangeira,...) references
                     tabelaestrangeira deferrable);
set constraints all deferred;
set constraints all immediate;
```

Eliminação de restrições

```
alter table tabela drop constraint umarestrição;
```



Oracle

Exemplo

```
SQL> create table bids( bid_id varchar2(10), bidder_id varchar2(10), item_id varchar2(10),
2   bid_amount number, primary key(bid_id), foreign key ( item_id ) references
3   auction_items, foreign key (bidder_id) references members(member_id) deferrable);
```

Table created.

SQL>

```
SQL> select constraint_name, constraint_type from user_constraints where
2   table_name='BIDS';
```

CONSTRAINT_NAME	C
SYS_C001400	P
SYS_C001401	R
SYS_C001401	R

```
SQL>
----- -
SYS_C001400      P
SYS_C001401      R
SYS_C001401      R
```

SQL>

```
SQL> alter table students drop constraint SYS_C001400;
```

Table altered.

```
SQL> alter table students add primary key( bid_id );
```

Table altered.

```
SQL> set constraints all deferred;
```

Constraint set.



Oracle

Utilização do SQLPlus

Fim de sessão

quit

Alteração de password

O comando passw altera a password do utilizador.

Não use os caracteres @ ou / .

passw

Terminação de comandos

Os comandos são guardados num buffer até aparecer o símbolo ;

Podem ocupar várias linhas. Em caso de erro faça:

Edit

O editor de texto associado ao SQLPlus pode ser alterado:

define _editor = nomeeditor

Confirmação de dados

commit;

Em caso de erro para não confirmar a operação:

rollback;



Oracle

“Logging”

```
spool nomeficheiro
```

Para terminar o “logging”:

```
spool off
```

Caracteres especiais

```
set escape \
select 'It"s mine\'; I like it' from dual;
```

Para terminar:

```
set escape off
```

Carregamento de comandos a partir de um ficheiro

```
@nomeficheiro.sql
```

Execução de comandos no “login”

Os comandos devem constar no ficheiro login.sql



Oracle

Formato da data

O Oracle oferece muitas opções para a data.

O comando `ALTER SESSION` permite alterar o formato da data.

Use `SET NLS_DATE_FORMAT` seguido da string de formato da data.

O formato por omissão é `DD-MON-YY`.

`ALTER SESSION SET NLS_DATE_FORMAT='YYYY/MM/DD HH24:MI'`

String	Significado
YYYY	Ano
MM	Mês
DD	Dia
HH	Hora
HH24	Hora (24 horas)
MI	Minuto
SS	Segundo



Oracle

Formato de Saída

```
SET NEWPAGE 0
SET SPACE 0
SET LINESIZE 80
SET PAGESIZE 0
SET ECHO OFF
SET FEEDBACK OFF
SET HEADING OFF
```

Ajuda adicional

help comando

Por exemplo:

help index

Exemplo

```
SQL> spool answer.txt
SQL> DEFINE _EDITOR= pico
SQL> spool off
SQL> commit;
Commit complete.

SQL> select username from junk;select username from junk
*ERROR at line 1:ORA-00942: table or view does not exist
SQL> c /junk/user_users 1* select username from user_users
SQL> /
USERNAME
-----
STUDENTNAME
SQL> quit
Disconnected from Personal Oracle9i Release 9.0.1.1.1 –
Production
With the Partitioning option
JServer Release 9.0.1.1.1 – Production
```



Constraints e Triggers

- “Constraints” são restrições sobre dados que devem manter-se verdadeiros em situação de mudança de estado, i.e., em transacções. Podem ser de diferentes tipos: baseadas no atributo, baseadas no tuplo, definição de chaves ou restrições referenciais. O sistema verifica se uma acção viola as restrições e em caso positivo aborta a execução da mesma. A implementação de “constraints” no Oracle é um pouco diferente do SQL standard.
- “Triggers” no Oracle são construções em PL/SQL semelhantes a “procedures”. No entanto, enquanto uma “procedure” é executada explicitamente a partir de um bloco e através de uma invocação (CALL), um “trigger” é executado implicitamente sempre que u determinado evento ocorre (INSERT, DELETE, ou UPDATE). O “trigger” pode ser executado antes ou depois do evento (BEFORE ou AFTER).



Adiamento da verificação de “Constraint”

É às vezes necessário adiar a verificação de alguma “constraint”, por exemplo:

```
CREATE TABLE chicken  
(cID INT PRIMARY KEY,  
eID INT REFERENCES egg(eID));
```

```
CREATE TABLE egg(eID INT PRIMARY KEY,  
cID INT REFERENCES chicken(cID));
```



Adiamento da verificação de “Constraint”

Para resolver o problema deve-se criar primeiro as tabelas sem restrições:

```
CREATE TABLE chicken(cID INT PRIMARY KEY, eID INT);
CREATE TABLE egg(eID INT PRIMARY KEY, cID INT);
```

E depois adicionar as restrições (chaves estrangeiras):

```
ALTER TABLE chicken ADD CONSTRAINT chickenREFegg
FOREIGN KEY (eID) REFERENCES egg(eID)
INITIALLY DEFERRED DEFERRABLE;
```

```
ALTER TABLE egg ADD CONSTRAINT eggREFchicken
FOREIGN KEY (cID) REFERENCES chicken(cID)
INITIALLY DEFERRED DEFERRABLE;
```

INITIALLY DEFERRED DEFERRABLE permite adiar a verificação de restrições até ao comando COMMIT.

Por exemplo:

```
INSERT INTO chicken VALUES(1, 2);
INSERT INTO egg VALUES(2, 1);
COMMIT;
```



Adiamento da verificação de “Constraint”

Para eliminar as tabelas deve-se eliminar primeiro as “constraints”:

```
ALTER TABLE egg  
DROP CONSTRAINT eggREFchicken;
```

```
ALTER TABLE chicken  
DROP CONSTRAINT chickenREFegg;
```

```
DROP TABLE egg;  
DROP TABLE chicken;
```



Violação de “Constraint”

O Oracle retorna uma mensagem de erro em caso de violação de “constraint”. Dependendo do tipo de acesso à base de dados (ODBC, JDBC, PL/SQL) o controlo de erros pode ser implementado. O Oracle disponibiliza mensagens de erro tais como:

ORA-02290: check constraint (YFUNG.GR_GR) violated

Ou

ORA-02291: integrity constraint (HONDROUL.SYS_C0067174)
violated - parent key not found



Síntaxe básica de um “Trigger”

```
CREATE [OR REPLACE] TRIGGER <trigger_name>
{BEFORE|AFTER}
{INSERT|DELETE|UPDATE} ON <table_name>
[REFERENCING
[NEW AS <new_row_name>]
[OLD AS <old_row_name>]]
[FOR EACH ROW [WHEN (<trigger_condition>)]]]
<trigger_body>
```



Síntaxe básica de um “Trigger”

- Triggers do tipo BEFORE e AFTER são apenas usados para tabelas (podem ser usados outro tipo de “triggers” para views, para implementar updates de views).
- Podem ser especificados até 3 eventos usando a palavra OR. UPDATE pode ser seguido, opcionalmente, pela palavra OF e uma lista de atributos em <table_name>. Desse modo, a cláusula OF define um evento que afecta apenas os atributos especificados. Por exemplo:
 - ... INSERT ON R ...
 - ... INSERT OR DELETE OR UPDATE ON R ...
 - ... UPDATE OF A, B OR INSERT ON R ...
- Com a opção FOR EACH ROW, o trigger do tipo *row-level*; senão é do tipo *statement-level*.



Síntaxe básica de um “Trigger”

- Para triggers do tipo *row-level*:
 - As variáveis do tipo NEW e OLD estão disponíveis para referenciar o valor do tuplo respectivamente depois ou antes da transacção. **Note-se:** No corpo do trigger, NEW e OLD devem ser precedidos por (":"), mas na cláusula WHEN, tal não se verifica.
 - A cláusula REFERENCING é usada para atribuir sinónimos à variáveis NEW e OLD.
 - Uma restrição pode ser especificada na cláusula WHEN. Esta condição não pode conter *subqueries*.
 - <trigger_body> é um bloco em PL/SQL. Existem algumas restrições a ter em conta de forma a não entrar em ciclos infinitos.



Exemplo

```
CREATE TABLE T4 (a INTEGER, b CHAR(10));
```

```
CREATE TABLE T5 (c CHAR(10), d INTEGER);
```

```
CREATE TRIGGER trig1
AFTER INSERT ON T4
REFERENCING NEW AS newRow
FOR EACH ROW
WHEN (newRow.a <= 10)
BEGIN
INSERT INTO T5 VALUES(:newRow.b, :newRow.a);
END trig1;
.
run;
```

Sem o comando run o trigger nunca seria executado.



Erros na definição de um Trigger

Após a mensagem:

Warning: Trigger created with compilation errors.

Os erros podem ser consultados com:

show errors trigger <trigger_name>;

ou

SHO ERR; (abreviatura de SHOW ERRORS)

para aceder aos erros mais recentes.



Consulta de Triggers

select trigger_name from user_triggers;

Para mais detalhe sobre um trigger, em particular:

*select trigger_type, triggering_event,
table_name, referencing_names,
trigger_body from user_triggers
where trigger_name = '<trigger_name>';*



Outros

Eliminação de Triggers

drop trigger <trigger_name>;

Desactivação de Triggers

alter trigger <trigger_name> {disable|enable};



Suspensão de Triggers em situação de Erro

```
create table Person (age int);
CREATE TRIGGER PersonCheckAge
AFTER INSERT OR UPDATE OF age ON Person
FOR EACH ROW
BEGIN
    IF (:new.age < 0) THEN
        RAISE_APPLICATION_ERROR(-20000, 'no negative age allowed');
    END IF;
END;
RUN;
```

Ao executar:

```
insert into Person values (-3);
```

obtem-se a mensagem de erro:

ERROR at line 1:

ORA-20000: no negative age allowed

ORA-06512: at "MYNAME.PERSONCHECKAGE", line 3

ORA-04088: error during execution of trigger 'MYNAME.PERSONCHECKAGE'



Exemplo – Uma Função Oracle

```

CREATE OR REPLACE FUNCTION
"SONHO"."PESQUISA" (n1 number,
n2 number, n3 number, n4 number,
n5 char, n6 varchar2, n7 varchar2, n8
char, n9 integer, n10 integer, apl
char, n11 integer) return integer as r
integer; x integer;
begin
r := 0;
select max(ped) into r from
pesquisa_temp;
if (r = 0)
then
    r := 1;
else
    r := r + 1;
end if;

```

```

insert into pesquisa_temp values
(r,n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,apl);
commit;
loop
x := 0;
select ped into x from pesquisa_temp
where ped = r;
if (x != r)
then
    exit;
end if;
end loop;
return r;
end;

```



Execução da Função em VB

```
SQL = "Declare n1 Number(8) := " + trus(1) + "; n2  
Number(8) := " + trus(2) + "; n3 Number(9):=" +  
trus(3) + "; n4 Number(8):=" + trus(4) + "; n5 char  
(3):=" + trus(5) + "; n6 varchar2(10):=" + trus(6) +  
"; n7 varchar2(100) := " + Trim(trus(7)) + "; n8  
Char(1) := " + Trim(trus(8)) + "; n9 Integer:=" +  
trus(9) + "; n10 Integer:=" + trus(10) + "; n11  
Integer:=" + trus(11) + "; reg Integer; Begin reg :=  
sonho.pesquisa (n1, n2,n3,n4,n5,n6,n7,n8,n9,n10,"  
+ apl + ",n11); End;"
```

db.Execute Trim(SQL)



Triggers

```
CREATE OR REPLACE TRIGGER "PCE"."TCCORRENTE" AFTER INSERT ON
"PCE"."CONSULTAS" REFERENCING OLD AS OLD NEW AS NEWROW
FOR EACH ROW
declare r number;
begin
select P into r from ccorrente
where P = :newrow.P and A = to_char(:newrow.d,'yyyy')
      and M = to_char(:newrow.d,'mm');
update ccorrente set v = v + :newrow.precio
where P = :newrow.P and A = to_char(:newrow.d,'yyyy') and M = to_char
(:newrow.d,'mm');
exception
when NO_DATA_FOUND then
insert into ccorrente(p,A,m,v)
values (:newrow.p, to_char(:newrow.d,'yyyy'), to_char
(:newrow.d,'mm'),:newrow.precio);
end tccorrente;
```



Exemplos

```
create table encomenda (encomendaid int primary key, fornecedor name varchar2(50) not null, valortotal number(10,2) not null , constraint qmzero check (valortotal > 0) );
```

```
create table encomendalinha (
    encomendaid int not null,
    Numlinha int not null check(numlinha > 0),
    dsc varchar2(50) not null,
    quantidade int not null, precounitario number(6,2) check ( precounitario > 0.0),
    constraint ol_pk primary key (encomendaid, numlinha),
    constraint olofk foreign key(encomendaid) references encomenda(encomendaid) );
```



Exemplos

```

create or replace trigger settotal
after insert or update or delete on encomendalinha
for each row
declare val number(10 ,2) := 0; eid int;
Begin
  if inserting then
    val := :new.quantidade * :new.precounitario;
    eid := :new. encomendaid ;
  elsif updating then
    val := :new.quantidade * :new. Precunitario - :old.quantidade * :old.precounitario;
    eid := :new.encomendaid;
  elsif deleting then val := 0 - :old.quantidade * :old.precounitario;
    eid := :old.encomendaid;end if; execute updateencomenda(eid,val); end;

```

```

create or replace procedure updateencomenda(eid int, val number(10,2)) is
begin ... End,
Insert into encomenda ..
Insert into encomendalinha ...

```



Exemplos

```

drop table user1.tab1;
create table user1.tab1 as
  select f.d1 as desp,
         f.c1 as cesp,f.d2 as dunid, nvl(f.c2,0) as cunid,
         to_char(f.dataq,'yyyymm') as mesq,
         x1(nvl(f.c2,0),f.c1,
              to_char(f.dataq,'yyyymm')) as tot, x2(nvl(f.c2,0), f.c1,to_char
(f.dataq,'yyyymm')) as totr, x3(nvl(f.c2,0), f.c1,to_char(f.dataq,'yyyymm'),1) as q1,
         x3(nvl(f.c2,0), f.c1,to_char(f.dataq,'yyyymm'),2) as q2, x3(nvl(f.c2,0),
f.c1,to_char(f.dataq,'yyyymm'),3) as q3, x3(nvl(f.c2,0), f.c1,to_char
(f.dataq,'yyyymm'),4) as q4, x3(nvl(f.c2,0), f.c1,to_char(f.dataq,'yyyymm'),5)
as q5, x4(nvl(f.c2,0), f.c1,to_char(f.dataq,'yyyymm'),1) as tq1, x4(nvl(f.c2,0),
f.c1,to_char(f.dataq,'yyyymm'),2) as tq2, x4(nvl(f.c2,0), f.c1,to_char
(f.dataq,'yyyymm'),3) as tq3, x4(nvl(f.c2,0), f.c1,to_char(f.dataq,'yyyymm'),4)
as tq4, x4(nvl(f.c2,0), f.c1,to_char(f.dataq,'yyyymm'),5) as tq5 from user2.tab2
f group by f.d1,f.c1,f.d2, nvl(f.c2,0), to_char(f.dataq,'yyyymm');

```



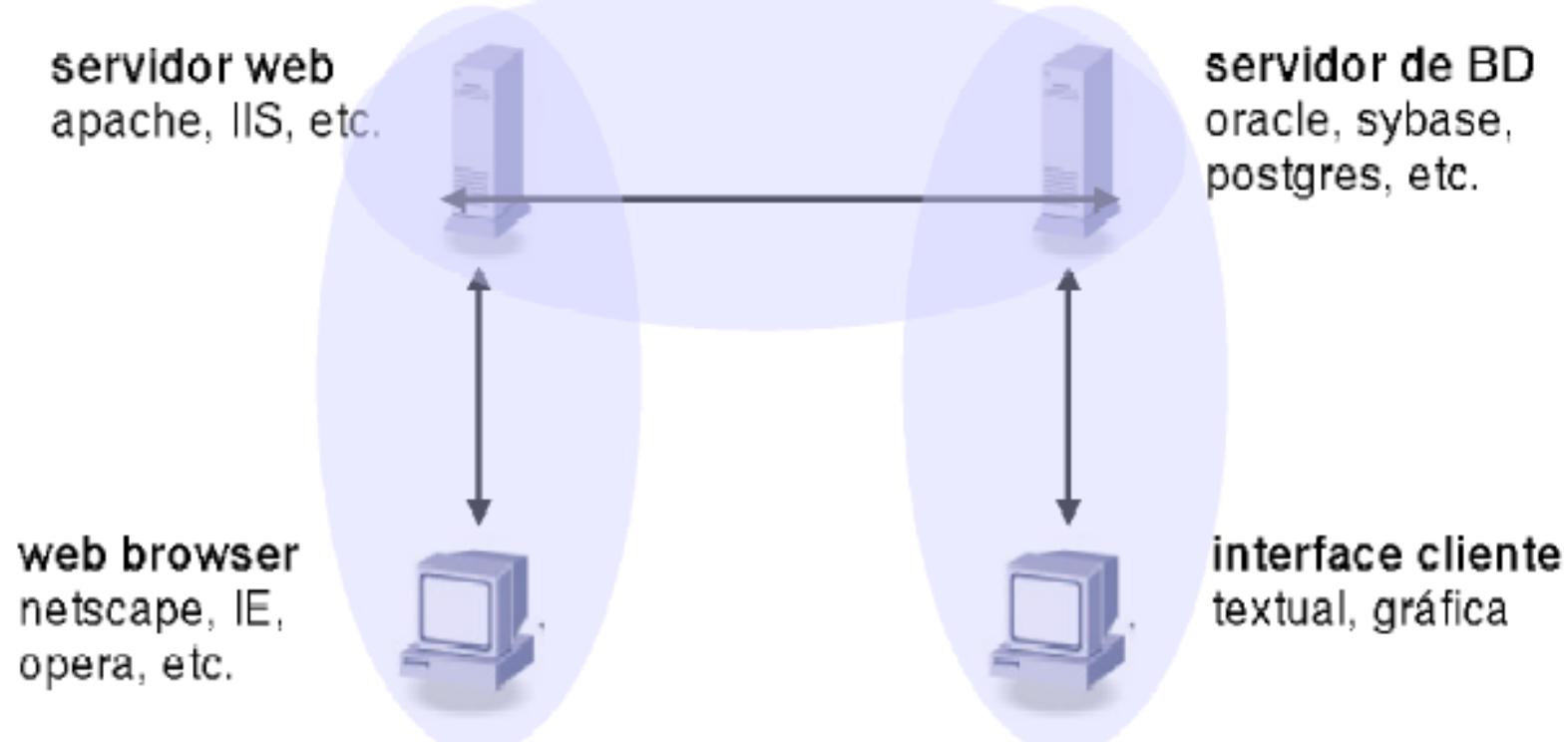
Bases de Dados Distribuídas

Uma base de dados distribuída é um sistema de bases de dados cujos dados se encontram fisicamente dispersos por várias máquinas, ligadas por meios de comunicação, mas integrados logicamente.

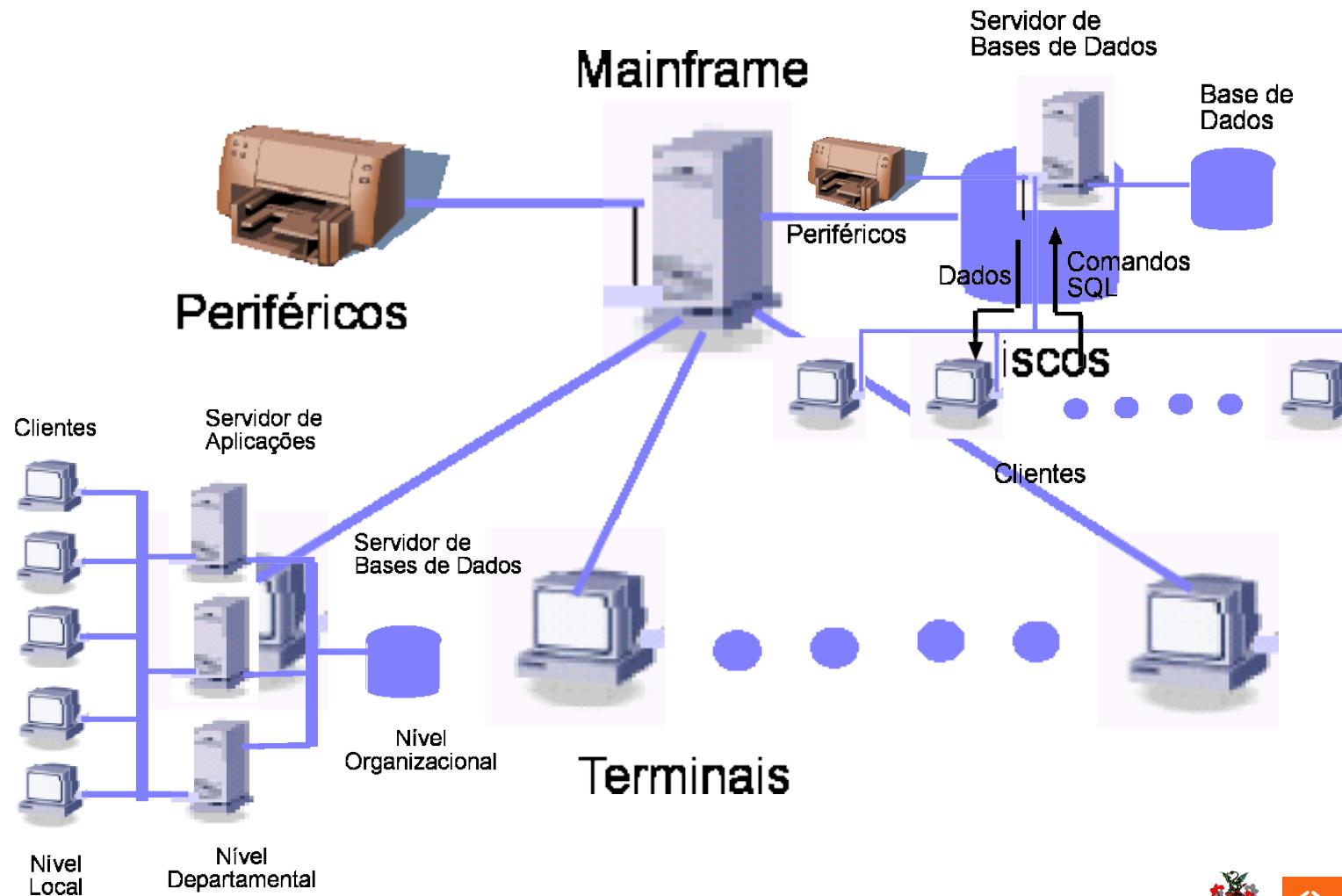
Define-se uma base de dados distribuída como sendo um conjunto de centros de computação, ligados por redes de computadores em que, por um lado, cada centro constitui um sistema de bases de dados por si; por outro lado, os vários centros concordaram em cooperar, de tal forma que um utilizador de um dos centros pode utilizar dados armazenados nos outros centros como se esses dados lhe fossem locais.



Arauitecturas



Arquitecturas



Conceitos Básicos

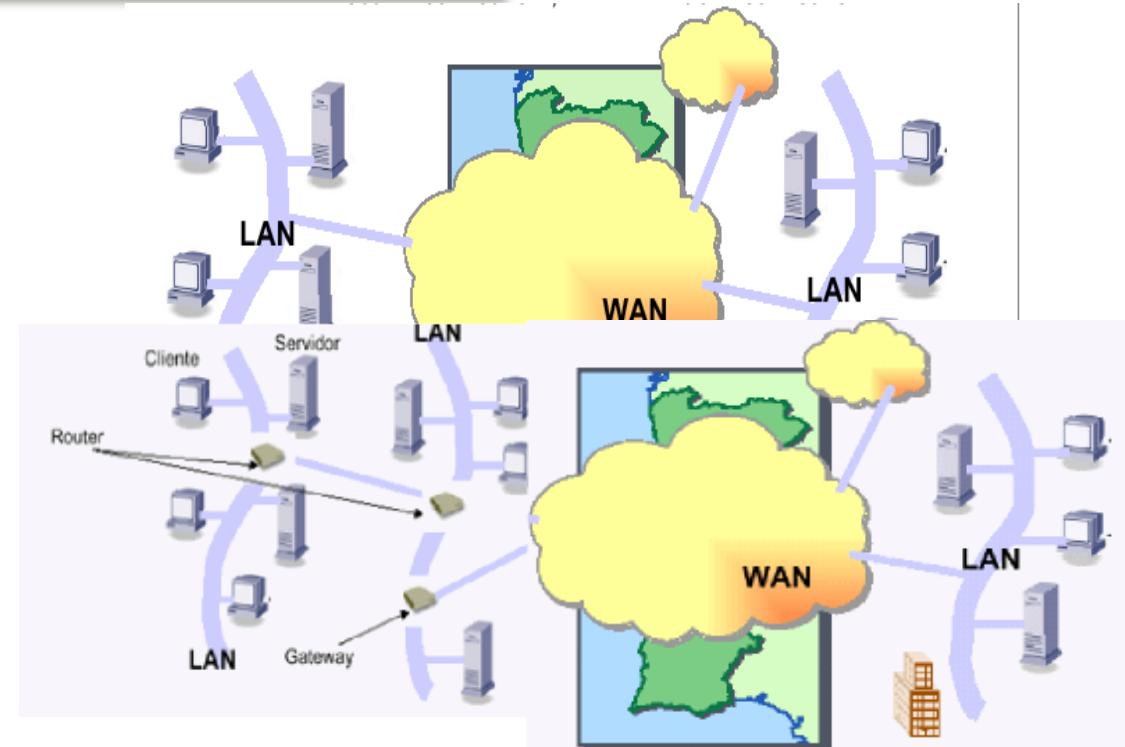
- **Homogéneos.** Neste caso, todos os nodos usam o mesmo SGBD, fazendo lembrar um único sistema de bases de dados mas em que, em vez de todos os dados estarem armazenados num único repositório, os dados estão armazenados por vários repositórios ligados por meios de comunicação.
- **Heterogéneos.** Este tipo de sistema de bases de dados distribuídas caracteriza-se pela existência de SGBDs diferentes nos vários nodos. As diferenças entre os SGBDs presentes podem colocar-se a vários níveis, desde SGBDs diferentes baseados no mesmo modelo de dados (por exemplo, Oracle, DB2, Informix, Sybase, etc., consistindo em implementações distintas do modelo relacional!), até SGBDs baseados em modelos de dados diferentes (relacionais, rede, OO, etc.).



Infra-estruturas de Comunicação

LAN (Local Area Network). Os computadores localizam-se relativamente próximo uns dos outros (menos de 1km), interligados por cablagem directa. Actualmente, as taxas de transmissão mais comuns situam-se entre os 10 Mbps (Ethernet) e os 100 Mbps (Fast Ethernet), sendo a transmissão do tipo broadcasting.

WAN (Wide Area Network). Neste caso, os computadores encontram-se geograficamente distantes, interligados por meios de comunicação próprios ou alugados a terceiros. As taxas de transmissão podem rondar os 50 Kbps, sendo a transmissão do tipo "ponto-a-ponto".



Dados vs Processamento

Em termos de tratamento informático, qualquer sistema, quer este seja suportado por um ambiente de bases de dados ou de gestão de ficheiros, envolve, fundamentalmente, dois elementos:

- Dados.
- Processamento.

No contexto específico das bases de dados, e em termos de distribuição destes dois elementos, é possível definir algumas configurações distintas, desde as configurações em que a sua distribuição é nula, ou seja, os dois elementos encontram-se centralizados numa só máquina, até às configurações em que estes dois elementos se encontram distribuídos por várias máquinas, ligadas por meios de comunicação.



Replicação de Dados

Se os dados estão distribuídos por pontos geograficamente distintos, sempre que houver necessidade de lhes aceder, estes terão que ser transferidos desde os seus locais de origem, até ao local que os solicitou. Estas “viagens” significam custos, quer em termos de exploração dos meios de comunicação, quer em termos de tempo consumido.

Uma alternativa, que pode reduzir significativamente estes custos, consiste em replicar os dados mais frequentemente solicitados pelos nodos que mais os solicitam. Desta forma, aumenta-se o processamento local a cada nodo pois há maior probabilidade de os dados necessários se encontrarem armazenados localmente.



Sincronização

Naturalmente, a necessidade de manter as réplicas permanentemente sincronizadas traz algumas dificuldades. De facto, para satisfazer este requisito é necessário garantir que cada actualização só entra em vigor quando todas as suas réplicas forem também actualizadas com sucesso. Numa situação dessas, a finalização de uma transacção distribuída envolvendo actualização de dados tem que utilizar um protocolo específico que garanta que todas as réplicas ficam sincronizadas, caso isso não seja possível a transacção deve falhar em todas elas, desfazendo os seus efeitos.

Infelizmente, num sistema distribuído existem vários factores que podem colidir com a necessidade de manter todas as réplicas sincronizadas, desde falhas nas comunicações, até falhas dos próprios nodos individuais.



Tipos de Replicação

Existem dois tipos básicos de replicação. Um, mais simples, designado “primary site replication”, em que um dos nodos (“primary site”) detém a posse dos dados, sendo o único que os pode actualizar. Este, de forma sincronizada ou não, propaga essas actualizações para os nodos onde estão as suas réplicas - denominadas “snapshots”.

Numa versão mais sofisticada, designada “dynamic ownership”, a autorização para actualizar dados replicados vai-se movimentando entre os nodos. Em todo o caso, em cada momento apenas um dos nodos possui a autorização.

Um outro tipo de replicação mais complexo, designado replicação simétrica, permite que qualquer réplica seja actualizada, a qualquer momento, eventualmente em simultâneo, propagando-se o efeito dessas actualizações, de forma sincronizada ou não, para as restantes réplicas.

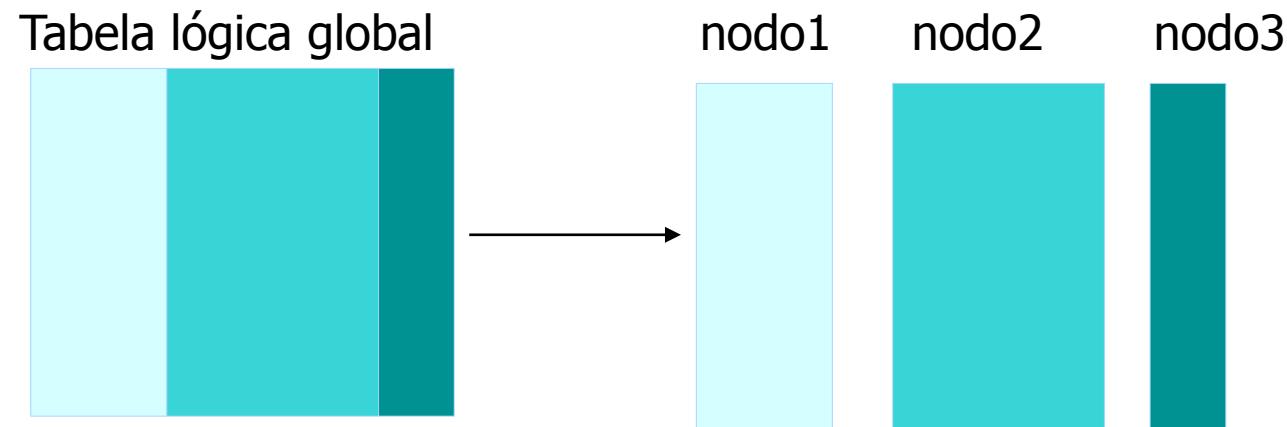


Fragmentação

- **Fragmentação vertical.** A tabela lógica é dividida em tabelas cujos esquemas são subconjuntos do esquema da tabela original (na sua versão mais simples, a chave da tabela lógica propaga-se a todos os fragmentos verticais). Por analogia com as operações da álgebra relacional, a fragmentação vertical pode ser vista como a execução de projecções sobre a tabela lógica, armazenando as tabelas resultantes (tabelas físicas) em nodos diferentes.
- **Fragmentação horizontal.** Neste caso, a tabela lógica é dividida em várias partes, cada uma delas com o mesmo esquema da tabela original. Continuando a analogia com as operações da álgebra relacional, a tabela lógica é o resultado da união de todos os fragmentos horizontais armazenados em nodos diferentes.
- **Fragmentação mista.** Sobre uma mesma tabela lógica são definidos fragmentos verticais e horizontais.

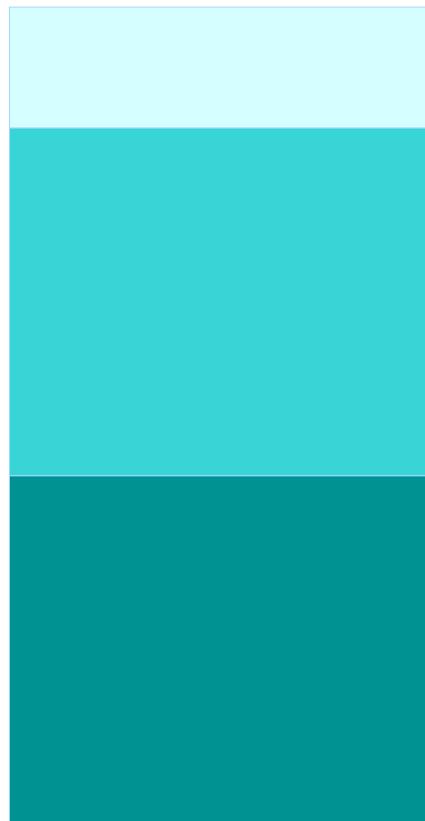


Fragmentação Vertical



Fragmentação Horizontal

Tabela lógica global



nodo1

nodo2

nodo3



Características

Num sistema de bases de dados distribuídas, no nível aplicacional, deveria-se aceder a dados localizados algures na rede com a mesma facilidade com que se acederia a esses mesmos dados caso estes residissem numa mesma máquina.

Ao suportar esta característica, a tecnologia de bases de dados distribuídas combina dois conceitos divergentes:

- Distribuição física dos dados.
- Integração lógica dos mesmos.

As bases de dados distribuídas reduzem:

- As dificuldades de desenvolvimento; e
- As dificuldades de manutenção.

As bases de dados distribuídas asseguram:

- A disponibilidade permanente dos nodos
- Uma fragmentação transparente
- Uma replicação transparente.
- Um processamento de questões distribuído.
- A Independência do hardware, sistema operativo e tecnologias de rede.



Abordagens

- **Top-down.** Correspondente à divisão de uma base de dados pré-existente ou, mais genericamente, de um esquema de base de dados pré-definido em várias partes a armazenar em diferentes nodos. Normalmente, o resultado deste processo será um ambiente distribuído homogéneo de bases de dados.
- **Bottom-up.** Correspondente, não à desagregação, mas sim à integração de várias bases de dados pré-existentes numa base de dados global, distribuída por várias máquinas. Dada a maior provável heterogeneidade das várias bases de dados em presença, esta será a abordagem que mais dificuldades oferece.



Bases de Dados Distribuídas Heterogéneas

- **Utilização de “database gateways/middleware”.** Trata-se da abordagem normalmente utilizada quando se encontram presentes SGBDs diferentes mas baseados no mesmo modelo de dados.
- **Integração num modelo global.** É a abordagem adequada quando os SGBDs presentes se baseiam em modelos de dados diferentes. Neste caso é necessário utilizar um modelo global onde os modelos locais são mapeados.
- **Objectos distribuídos.** Dadas as características intrínsecas ao próprio modelo 00 (objectos comunicando entre si através de mensagens), este ajusta-se de forma natural aos ambientes distribuídos, em que os objectos se encontram geograficamente dispersos, comunicando entre si através de mensagens que percorrem a rede.



XML e Bases de Dados

Um documento XML é uma base de dados no sentido básico do termo (i.e., é um conjunto de dados e contém dados com tipos diferentes).

Vantagem: apresenta um formato aceite universalmente que facilita a troca de informação.

Desvantagem: o acesso aos dados é lento devido a questões de “parsing” e de conversão de texto.



XML e Bases de Dados

O XML oferece muitas das características encontradas em sistemas de bases de dados:

- armazenamento (ficheiros XML);
- esquemas (DTDs, XSL, etc...);
- linguagens de interrogação (XQuery, XPath, XQL, XML-QL, QUILT, etc...);
- interfaces de programação (SAX, DOM, JDOM).

Falhas :

- armazenamento eficiente;
- índices;
- segurança;
- transacções;
- integridade;
- multi-utilização;
- triggers;
- multi-interrogação / junção;
- etc...



XML e Bases de Dados

```

<encomenda numero="12345">
  <cliente nclie="123">
    <nomecli> Empresa Ida </nomecli>
    <localidade> Braga </localidade>
    ...
  </cliente>
  <dataencom>20-12-2003</dataencom>
  <artigo codarti = "XYZ">
    <designacao> Monitor X </designacao>
    <quantidade> 1 </quantidade>
    <preco> 351 </preco>
  </artigo>
  <artigo codarti = "XYZ2">
    ...
  </artigo>
</encomenda>

```

encomenda

numero	nclie	dataencom
12345	123	20-12-2003
...

clientes

nclie	ncomecli	localidade
123	Empresa Ida	Braga
...

artiencom

codarti	designacao	quantidade	preco
XYZ	Monitor X	1	351
XYZ2
...



XML e Bases de Dados

```
<database databasename=...>
  <table tablename = ...>
    <row>
      <column1> ... </column1>
      <column2>...</column2>
      ...
    </row>
    <row>
      ...
    </row>
  </table>
  <table tablename = ...>
    ...
  </table>
  ...
</database>
```

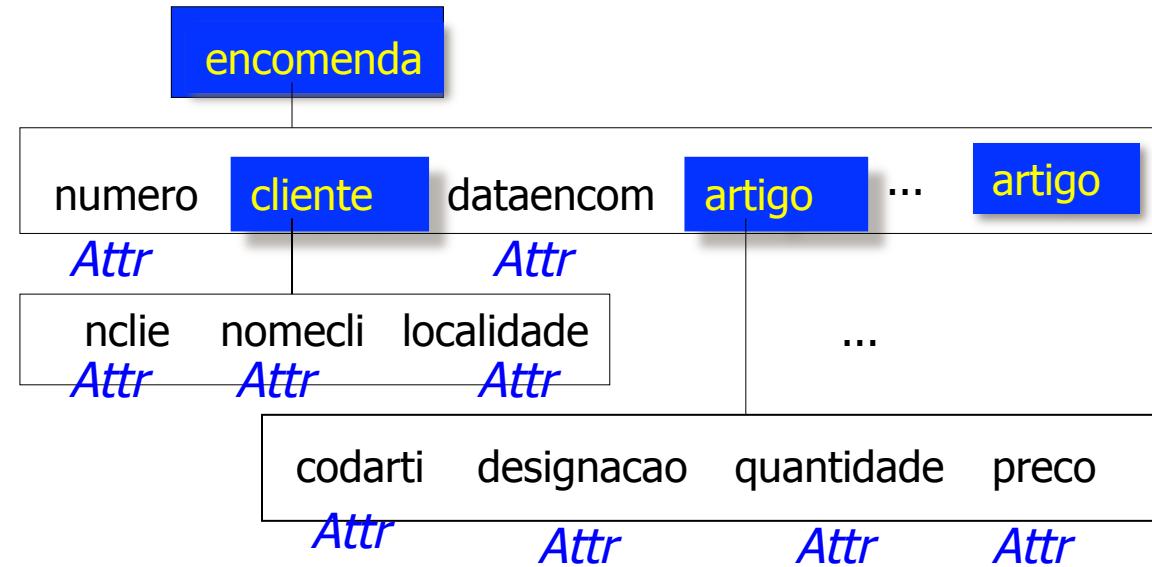


Document Object Model

```

<encomenda numero="12345">
  <cliente nclie="123">
    <nomecli> Empresa
    Ida</nomecli>
    <localidade> Braga </
    localidade>
    ...
  </cliente>
  <dataencom>20-12-2003</
  dataencom>
  <artigo codarti = "XYZ">
    <designacao>Monitor
    X</designacao>
    <quantidade>1</
    quantidade>
    <preco>351</preco>
  </artigo>
  <artigo codarti = "XYZ2">
  ...
  </artigo> </encomenda>

```



Oracle e XML

O Oracle disponibiliza o tipo de dados sys.XMLTYPE para gravar conteúdos em XML em tabelas. Por exemplo:

```
create table tabxml (
        idser varchar2(10),
        grupo varchar2(10),
        sintaxe varchar2(20),
        documento sys.XMLTYPE);
```

ou

```
create table PURCHASEORDER (PODOCUMENT sys.XMLTYPE);
```



Oracle e XML

```

insert into PURCHASEORDER (PODOCUMENT) values ( sys.XMLTYPE.createXML(
    <PurchaseOrder> <Reference>BLAKE-2001062514034298PDT</Reference>
    <Actions><Action> <User>KING</User> <Date/></Action> </Actions>
    <Reject/>
    <Requester>David E. Blake</Requester><User>BLAKE</User>
    <CostCenter>S30</CostCenter>
    <ShippingInstructions>
        <name>David E. Blake</name>
        <address>400 Oracle Parkway Redwood Shores, CA, 94065 USA</
address>
        <telephone>650 999 9999</telephone>
    </ShippingInstructions>
    <SpecialInstructions>Air Mail</SpecialInstructions>
    <LineItems><LineItem ItemNumber="1">
        <Description>The Birth of a Nation</Description>
        <Part Id="EE888" UnitPrice="65.39" Quantity="31"/>
    </LineItem></LineItems> </PurchaseOrder>
));

```



Oracle e XML

```
insert into tabxml values('RXU','UN','amostra',
sys.XMLTYPE.createXML(
'<amostra>
    <numero>123</numero> <episodio>445</
episodio><modulo>CON</modulo>
<facturacao>
    <resultado>    <codigo>abc</codigo><preco>100.25</preco>
    </resultado>
<resultado> <codigo>abd</codigo> <preco>200.5</preco>
</resultado>
</facturacao>
</amostra>'
));
```



Oracle e XML

```
select p.podocument.getClobVal() from purchaseorder p;
```

```
P.PODOCUMENT.GETCLOBVAL()
```

```
<PurchaseOrder> <Reference>BLAKE-2001062514034298PDT</Reference>
  <Actions> <Action> <User>KING</User> <Date/> </Action>
  </Actions> <Reject/>
  <Requester>David E. Blake</Requester> <User>BLAKE</User>
  <CostCenter>S30</CostCenter> <ShippingInstructions>
    <name>David E. Blake</name><address>400 Oracle Parkway Redwood Shores, CA,
  94065 USA</address>
    <telephone>650 999 9999</telephone></ShippingInstructions>
  <SpecialInstructions>Air Mail</SpecialInstructions> <LineItems>
    <LineItem ItemNumber="1"> <Description>The Birth of a Nation</Description>
      <Part Id="EE888" UnitPrice="65.39" Quantity="31"/> </LineItem>
  </LineItems>
```



Oracle e XML

```
select P.PODOCUMENT.extract('/PurchaseOrder/ShippingInstructions').getClobVal()  
      from PURCHASEORDER P
```

where

```
P.PODOCUMENT.extract('/PurchaseOrder/Reference/text()').getStringVal() =  
'BLAKE-2001062514034298PDT')
```

```
P.PODOCUMENT.EXTRACT('/PURCHASEORDER/SHIPPINGINSTRUCTIONS').GETCLOBVAL()
```

```
<ShippingInstructions>  
  <name>David E. Blake</name>  
  <address>400 Oracle Parkway Redwood Shores, CA, 94065 USA</address>  
  <telephone>650 999 9999</telephone>  
</ShippingInstructions>
```

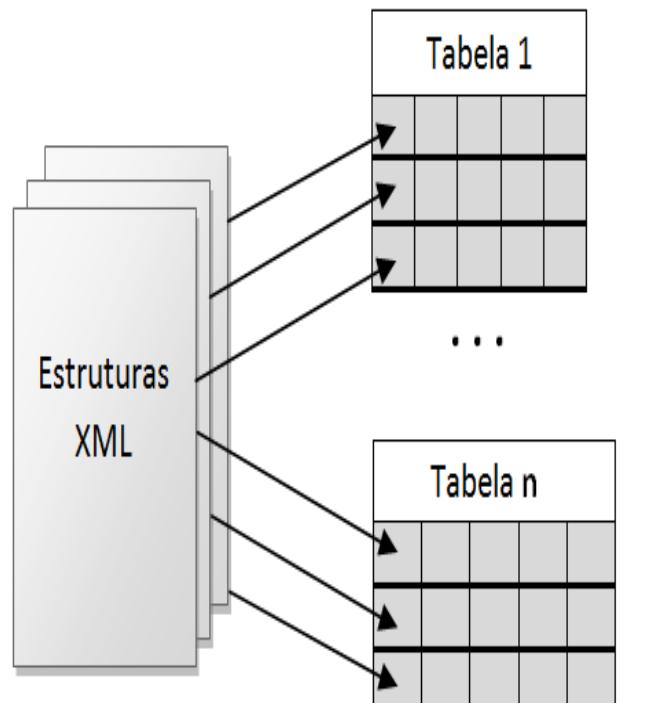


Oracle e XML

```
select P.DOCUMENTO.extract('/amostra/facturacao').getClobVal()  
  from tabxml P  
 where P.DOCUMENTO.extract('/amostra/numero/text()').getStringVal() = '123'
```

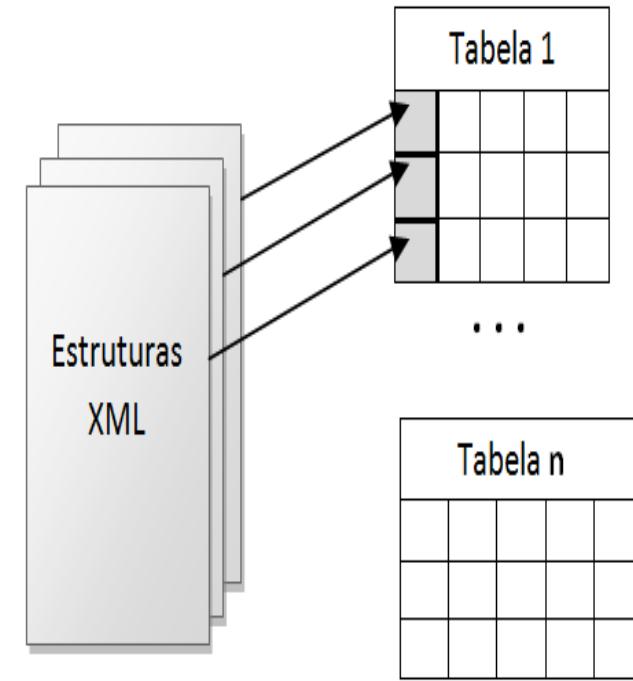


Oracle e XML



Base de Dados Relacional

a)



Base de Dados Relacional

b)



INSERT INTO ... SELECT

```
create table klinhas
(ki number(3),
ano varchar2(4),
mes varchar2(2),
cod_servico_unidade varchar2(1),
cod_servico_unidade_prev varchar2(1),
cod_unidade_inter number(3),
cod_unidade_inter_prev number(3),
cod_especialidade number(5),
cod_especialidade_prev number(5),
tint number(8),
tnom number(8),
tdenom number(8)
);
```

```
insert into klinhas(ki,ano,mes,cod_servico_unidade,
cod_servico_unidade_prev,cod_unidade_inter,
cod_unidade_inter_prev,
cod_especialidade,cod_especialidade_prev,tint)
select 1,to_char(dta_saida,'yyyy'),to_char
(dta_saida,'mm'),cod_servico_unidade,
cod_servico_unidade_prev,cod_unidade_inter,
cod_unidade_inter_prev,cod_especialidade,
cod_especialidade_prev, count(distinct int_episodio)
from transferencias
where to_char(dta_saida,'yyyy') = '2009'
group by 1,to_char(dta_saida,'yyyy'),
to_char(dta_saida,'mm'),
cod_servico_unidade,cod_servico_unidade_prev,
cod_unidade_inter,cod_unidade_inter_prev,
cod_especialidade,cod_especialidade_prev;
```



Select k,a,b,c from
(Select rownum as k,a,b,c from x)
Where k between 1 and 20;

Select k,a,b,c from
(Select rownum as k,a,b,c from x)
Where k between 21 and 40;

....



CREATE TABLE ... SELECT

```
CREATE TABLE T1  
SELECT A,B,C,D FROM tabela  
WHERE cond
```

```
CREATE TABLE T2  
SELECT A,B,C,D,op FROM tabela  
WHERE cond  
GROUP BY A,B,C,D
```

```
CREATE OR REPLACE VIEW V1 AS SELECT ....
```

```
CREATE MATERIALIZED VIEW MV1 AS SELECT ....
```



CREATE PROCEDURE

```
CREATE OR REPLACE PROCEDURE CRIA_KI (iniki_ number, finiki_ number) is
Begin
declare ki_ number(3); ano_ varchar2(4);
mes_ varchar2(2); cod_servico_unidade_ varchar2(1);
cod_servico_unidade_prev_ varchar2(1); cod_unidade_inter_ number(5);
cod_unidade_inter_prev_ number(5); cod_especialidade_ number(5);
cod_especialidade_prev_ number(5);
tnom_ number(8); tdenom_ number(8);
exp1_ varchar2(200); exp2_ varchar2(200); exp3_ varchar2(200);
tipo_ number(2); interid_ number(8); interid1_ number(8); interid2_ number(8);
Begin
....
End;
End;
```



Begin ... End

Begin

```
ki_ := iniki_ - 1;  
while (ki_ < finiki_) loop  
    ki_ := ki_ + 1;  
    if (kpi_ = 1) then  
        exp1_ := '.....';  tipo_ := 1;  
    end if;  
  
    ....  
    if (kpi_ = 12) then  
        exp1_ := '...';  
        tipo_ := 5;  
    end if;  
    begin ... end;  
end;
```



Cursos

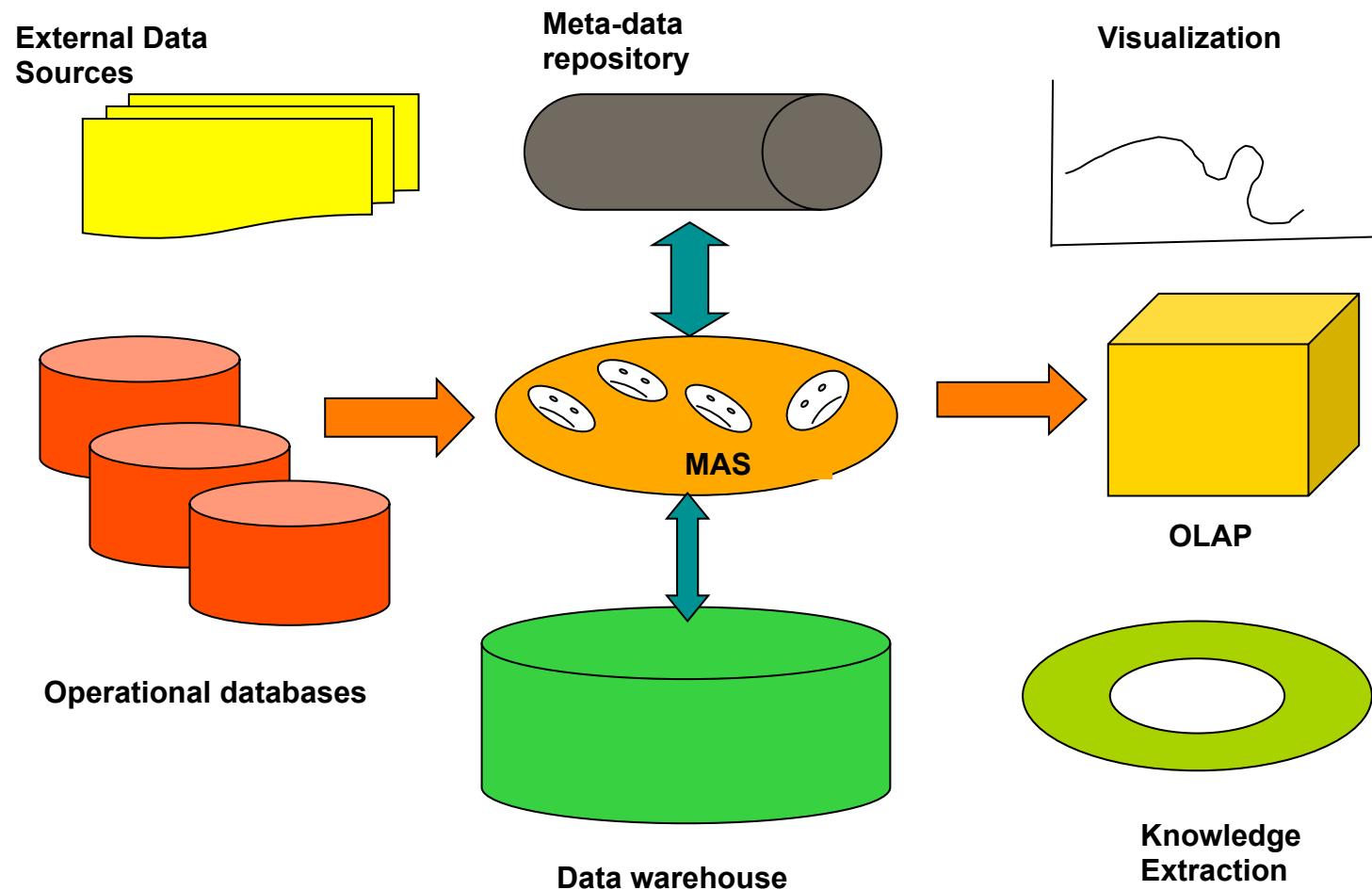
```

declare cursor c1 is select ano, mes, cod_servico_unidade, cod_servico_unidade_prev,
nvl(cod_unidade_inter,0), nvl(cod_unidade_inter_prev,0), cod_especialidade,
cod_especialidade_prev from kilinhas where ki = ki_;
begin
open c1;
fetch c1 into ano_,mes_,cod_servico_unidade_,cod_servico_unidade_prev_,
cod_unidade_inter_, cod_unidade_inter_prev_, cod_especialidade_, cod_especialidade_prev_;
while c1%FOUND loop
    SELECT ...
    UPDATE ...
    COMMIT
    fetch c1 into ano_, mes_, cod_servico_unidade_,
cod_servico_unidade_prev_, cod_unidade_inter_, cod_unidade_inter_prev_,
cod_especialidade_, cod_especialidade_prev_;
end loop;
close c1;
end;

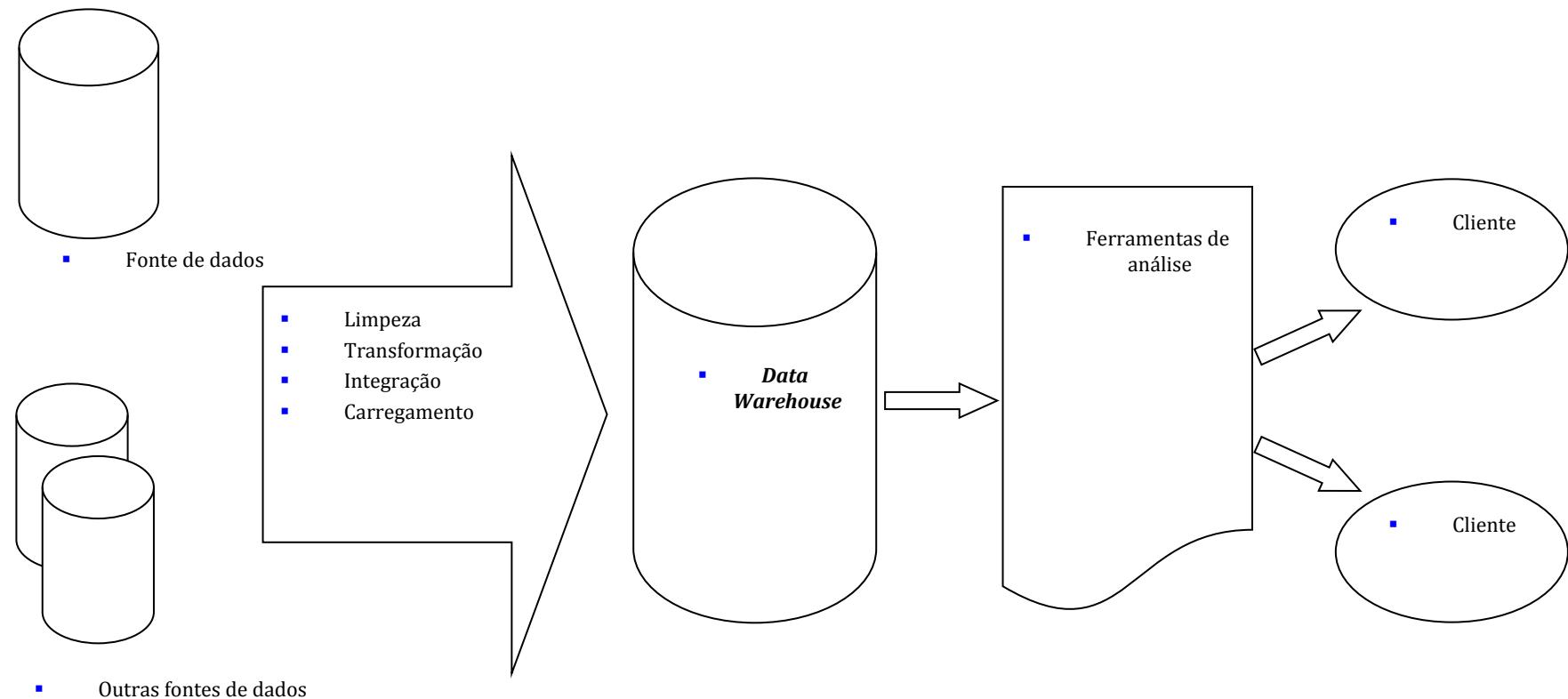
```



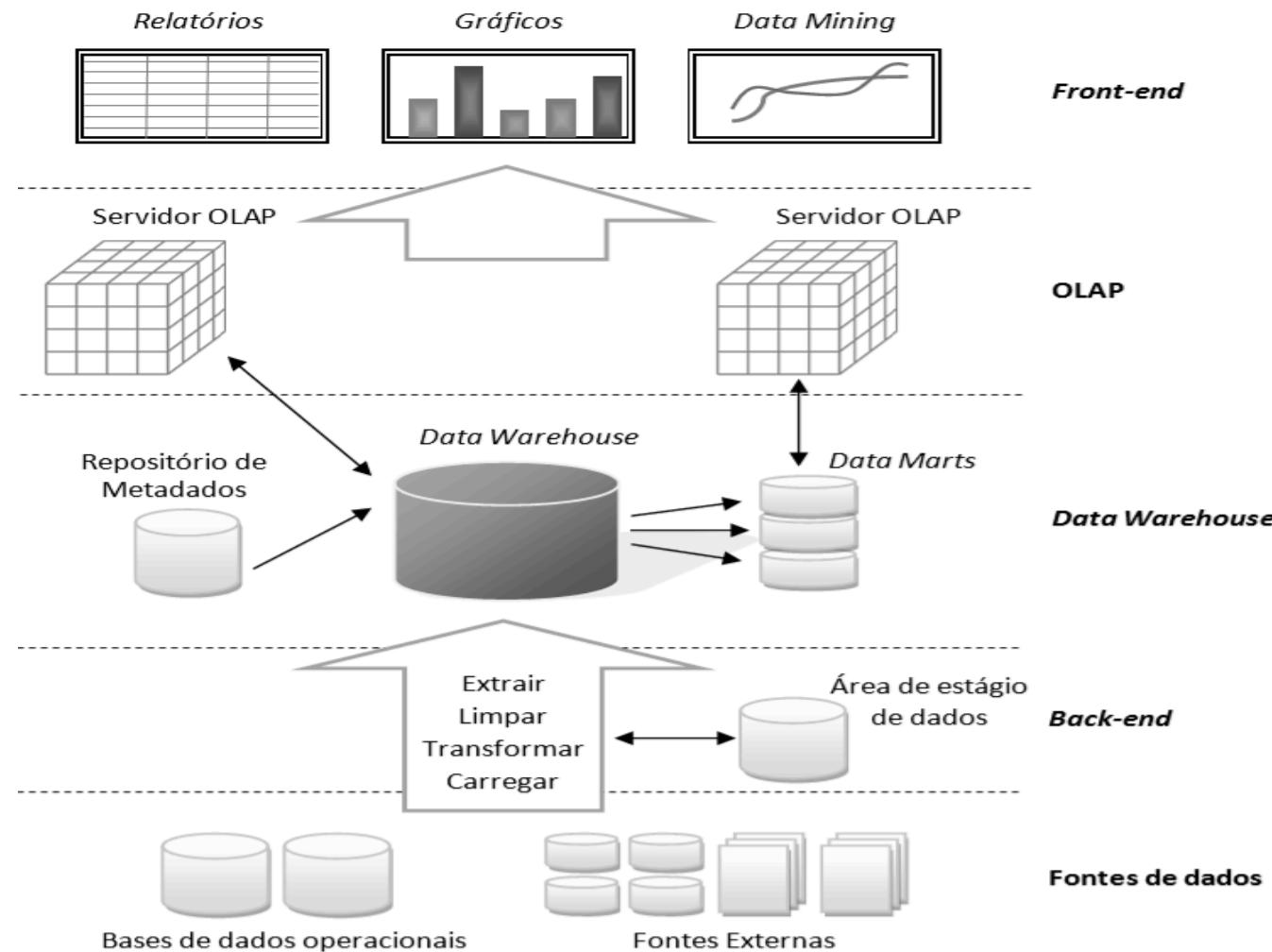
Datawarehousing



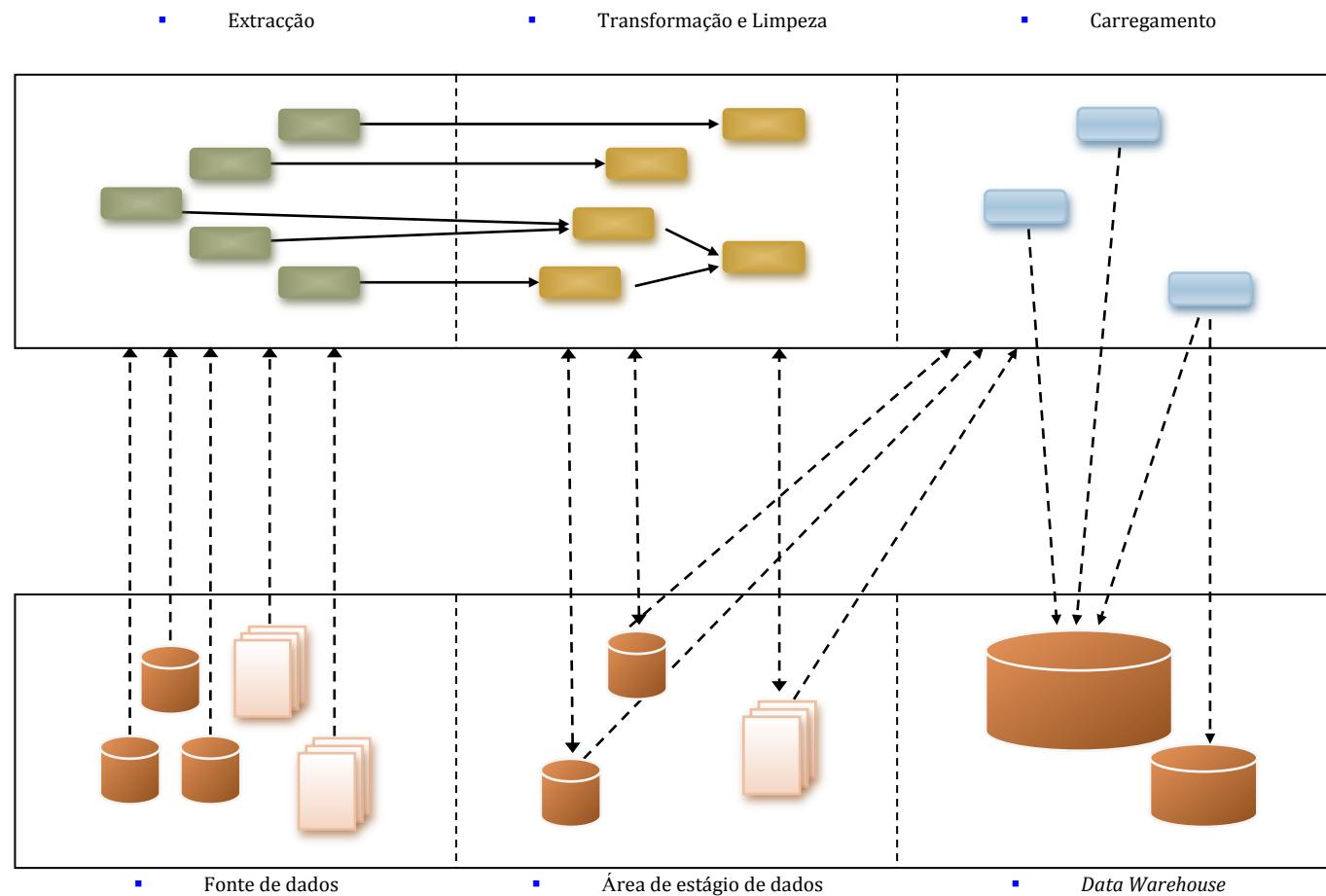
Dataware house vs BD



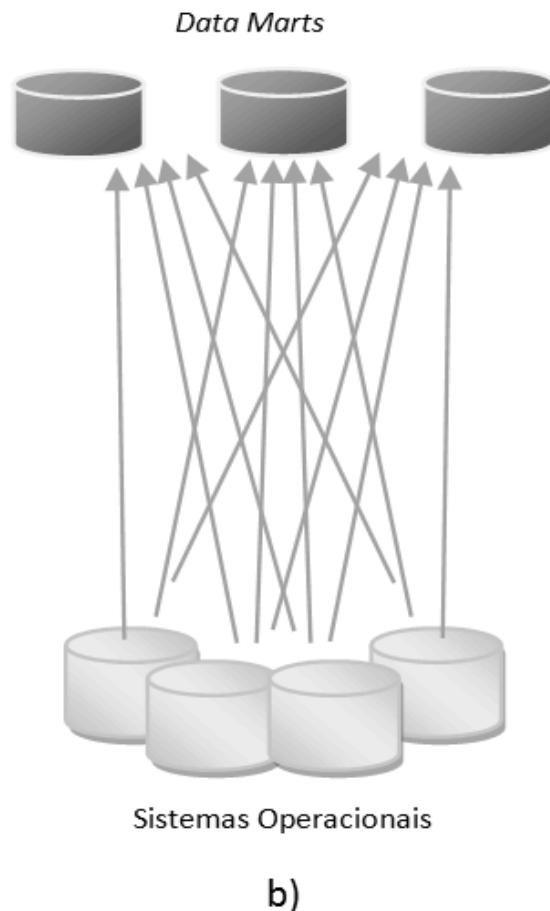
Arquitectura típica do DW



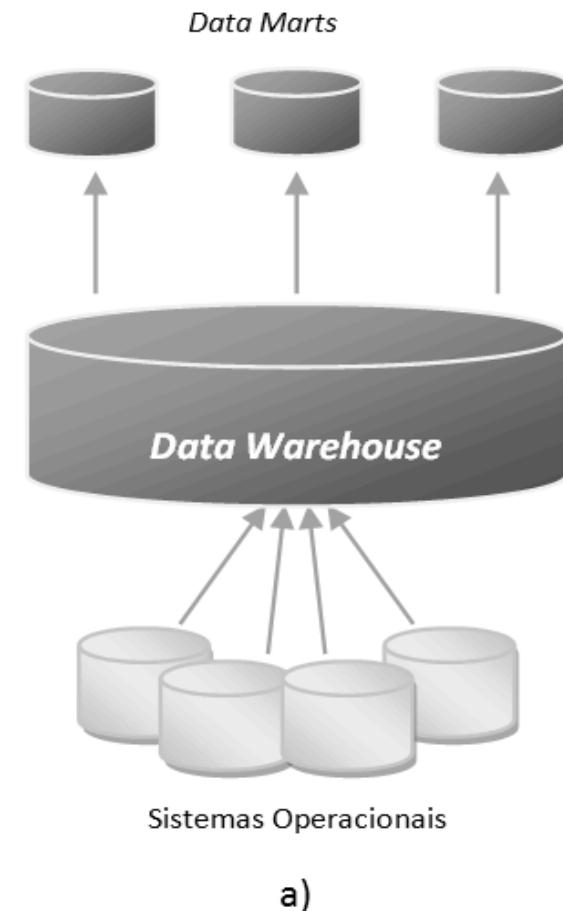
Processo ETL



Data Marts



b)



a)



Sistemas Multiagente

Definem um novo conceito de computação e inteligência.

- SMA é um novo paradigma na área da computação, baseado em novas formas de demonstração de teoremas; i.e., a computação baseada em agentes é uma revolução na análise e no desenvolvimento de software, simplificando a complexidade, a distribuição e a interactividade.
- Os Agentes são um foco de intenso interesse em muitas disciplinas das *Ciências de Computação*, sendo usados em várias aplicações, em sistemas pequenos até sistemas grandes, abertos, complexos e críticos.



Propriedades dos SMA

- *autonomia*;
- *reactividade*;
- *pro-actividade*;
- comportamento *social*;
- *etc...*

Estas propriedades têm facilitado a monitorização do comportamento dos agentes com impacto significativo no processo de aquisição de conhecimento e validação.

SMA são ferramentas poderosas para a resolução de problemas, em particular porque estes articulam com bases de dados, bases de conhecimento, sistemas de “datawarehouse” e extracção de conhecimento, ajudando o processo de integrar, difundir e arquivar informação (e.g., registos médicos).



Datawarehousing

- As organizações analisam data correntes e históricos para identificar padrões e suportar estratégias de negócio.
- A análise é complexa e interactiva e considera volumes elevados de dados integrados a partir de fontes variadas.
- A diferença entre OLAP (On-Line Analytic Processing) e OLTP (On-line Transaction Processing) é que a actualização da informação é substituída por interrogações frequentemente longas.



3 Áreas Complementares

- Data Warehousing: consolida dados a partir de muitas fontes num repositório mais vasto, considera o carregamento de dados, a sincronização periódica entre réplicas e a integração semântica.
- OLAP: Interrogações complexas em SQL e “views”; interrogações baseadas em operações do tipo folha de cálculo e visualização multidimensional dos dados; Interrogações interactivas e em tempo real.
- Data Mining: procura exploratória de tendências interessantes e de anomalias.



Datawarehouse

- Dados integrados que medem períodos de tempo longos;
- Adição frequente de informação sumária;
- Volume medido em gigabytes ou até terabytes;
- Melhoria dos tempos de resposta interactiva para perguntas complexas;
- Não é habitual fazer “updates” ad hoc.

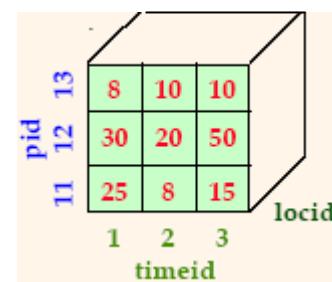


Datawarehouse

- Integração semântica: quando são processados dados de fontes múltiplas devem-se eliminar erros, e.g., moedas diferentes, esquemas,
- Fontes heterogéneas: os dados devem ser acedidos a partir de formatos de dados e repositórios variados (as potencialidades da replicação podem ser exploradas aqui).
- Integração, Actualização e Eliminação: os dados integrados devem ser periodicamente actualizados e os dados obsoletos devem ser eliminados.
- Gestão de meta-dados: devem ser guardados os parâmetros sobre a fonte da informação e os tempos de carregamento, assim como outros parâmetros relevantes sobre os dados.



Vista tri-dimensional



Locid = 1

pid	timeid	locid	vendas
11	1	1	25
11	2	1	8
11	3	1	15
12	1	1	30
12	2	1	20
12	3	1	50
13	1	1	8
13	2	1	10
13	3	1	10
11	1	2	35



Comparação com SQL

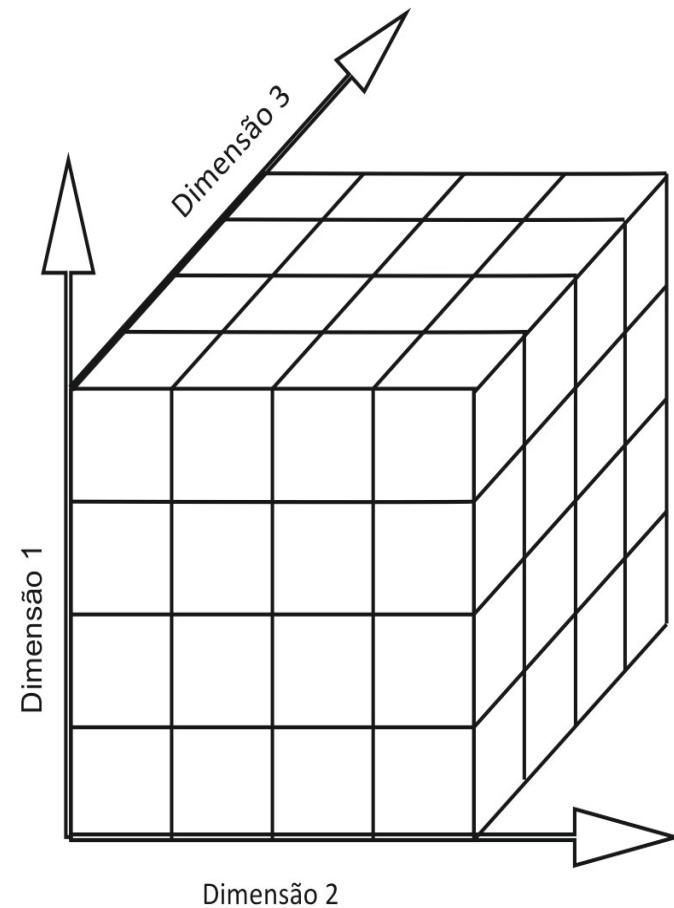
```
SELECT SUM(S.sales) FROM Sales S, Times T,  
Locations L WHERE S.timeid=T.timeid AND  
S.timeid=L.timeid GROUP BY T.year, L.state
```

```
SELECT SUM(S.sales) FROM Sales S, Times T  
WHERE S.timeid=T.timeid GROUP BY T.year
```

```
SELECT SUM(S.sales) FROM Sales S, Location L  
WHERE S.timeid=L.timeid GROUP BY L.state
```

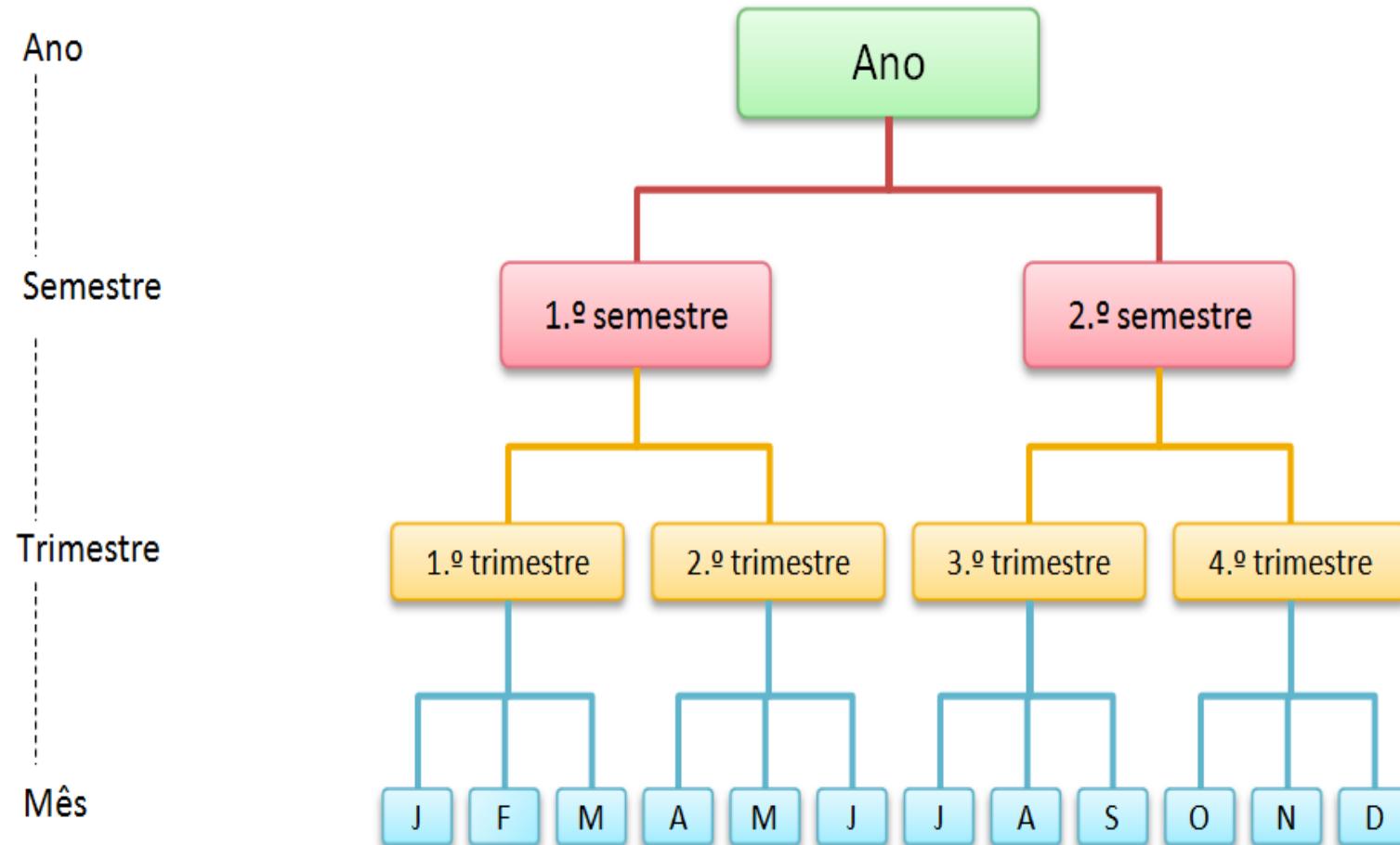


Cubos

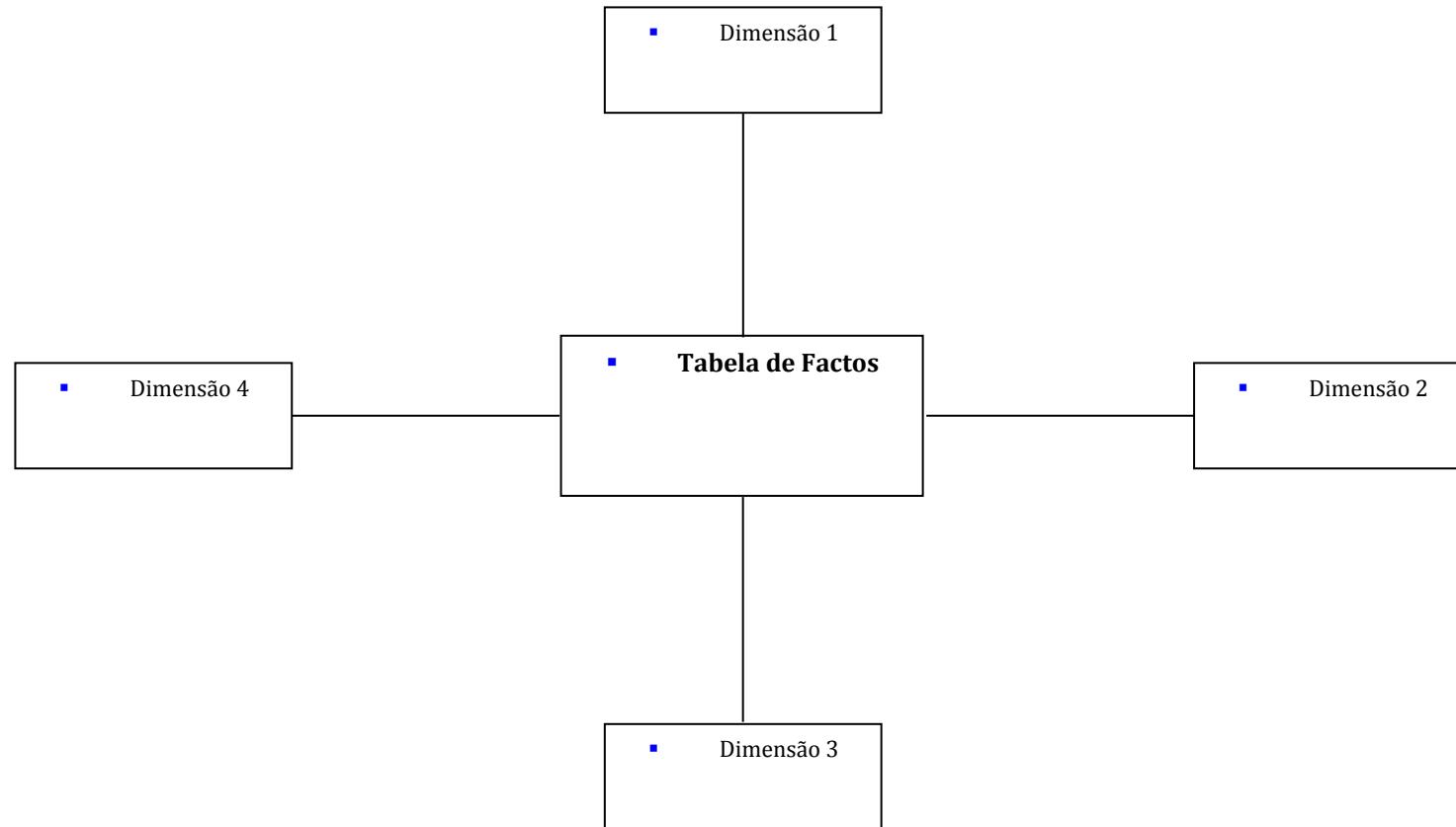


		Localização				
		Braga	90	235	110	
		Trofa	135	190	140	85
		Famalicão	150	20	80	40
		Guimarães	200	150	170	90
1.º trim		805	450	625	325	
2.º trim		790	530	700	400	
3.º trim		850	540	710	410	
4.º trim		900	520	600	390	
		CARD	URO	DERMA	ORTOP	Serviço

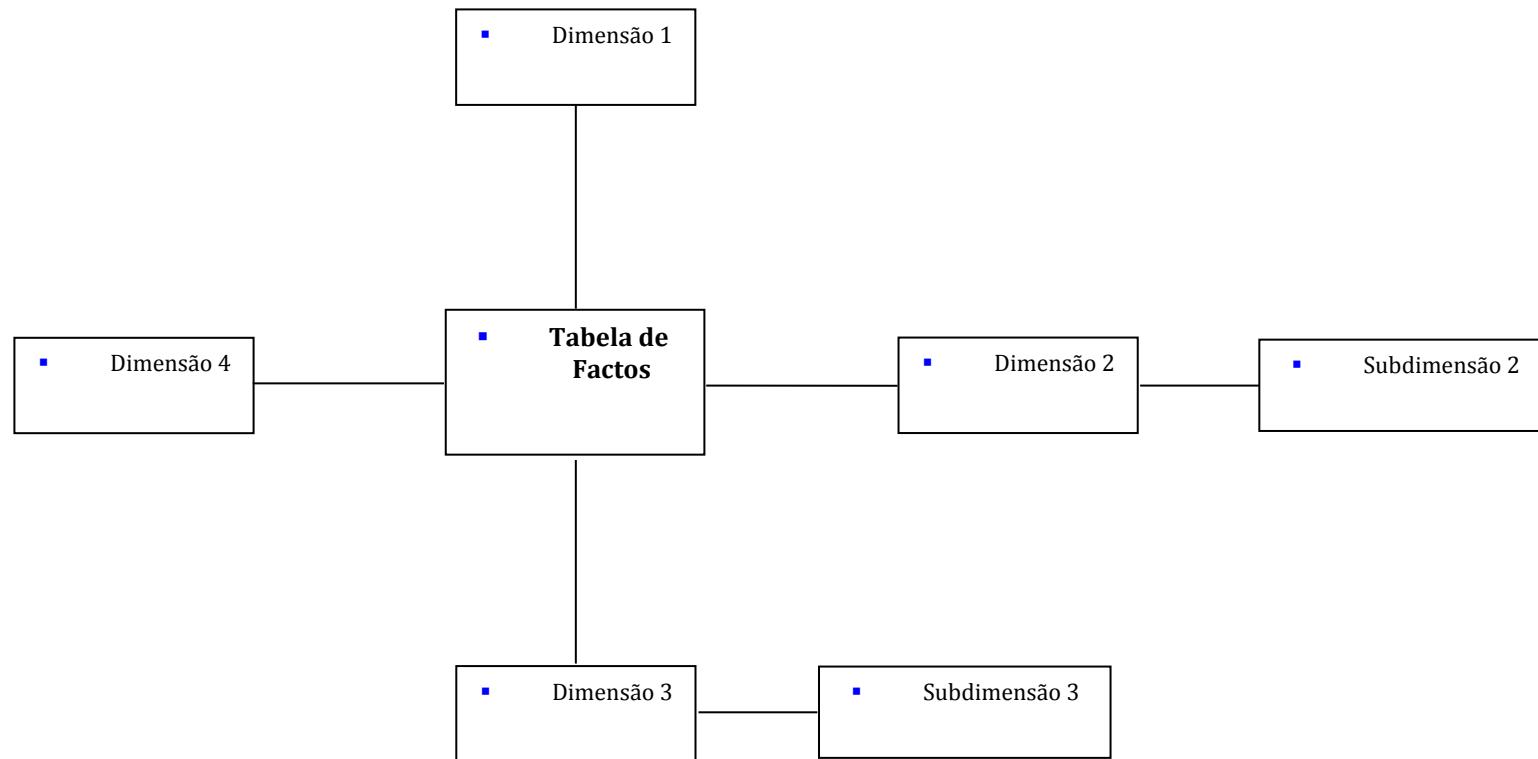
Hierarquia da dimensão tempo



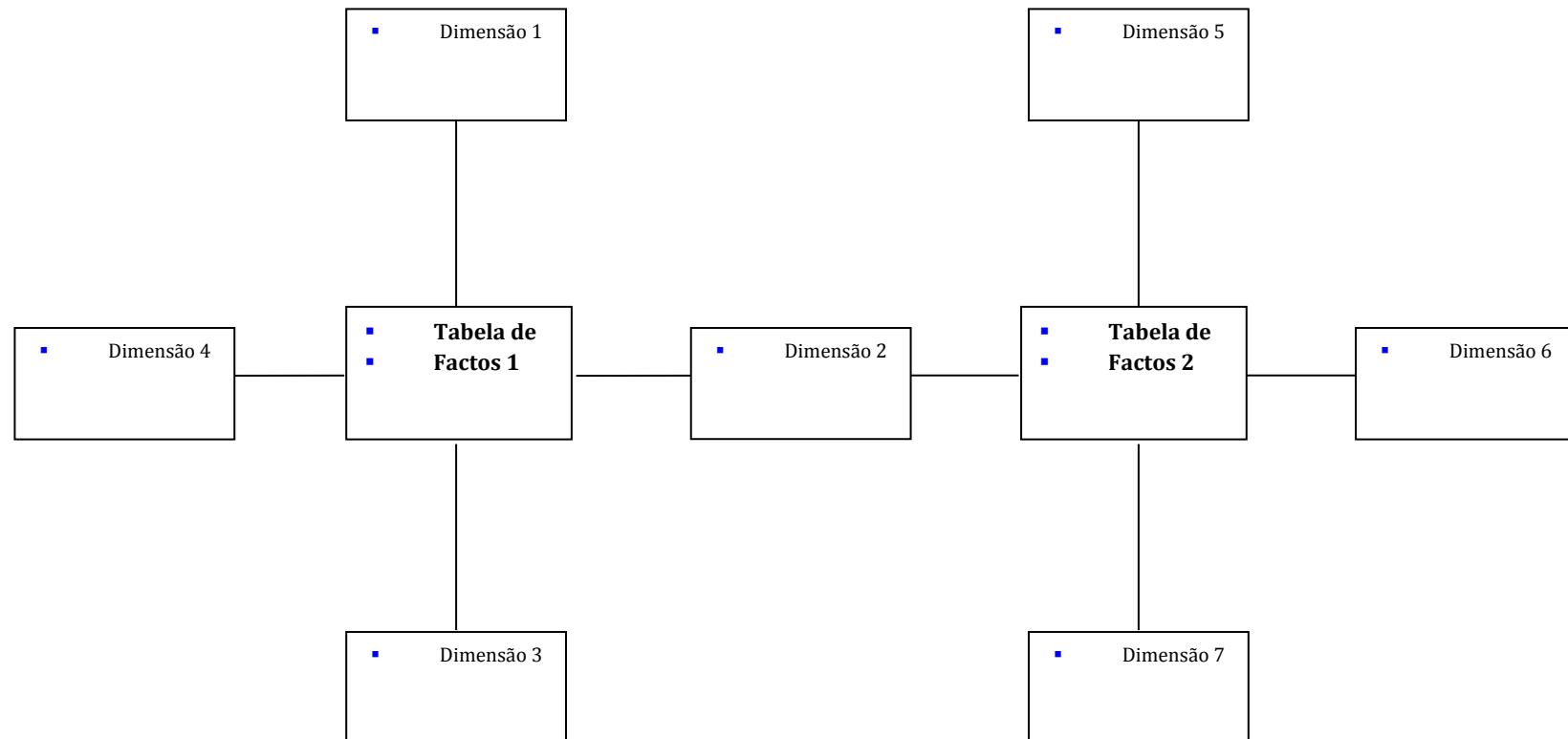
Esquema em estrela



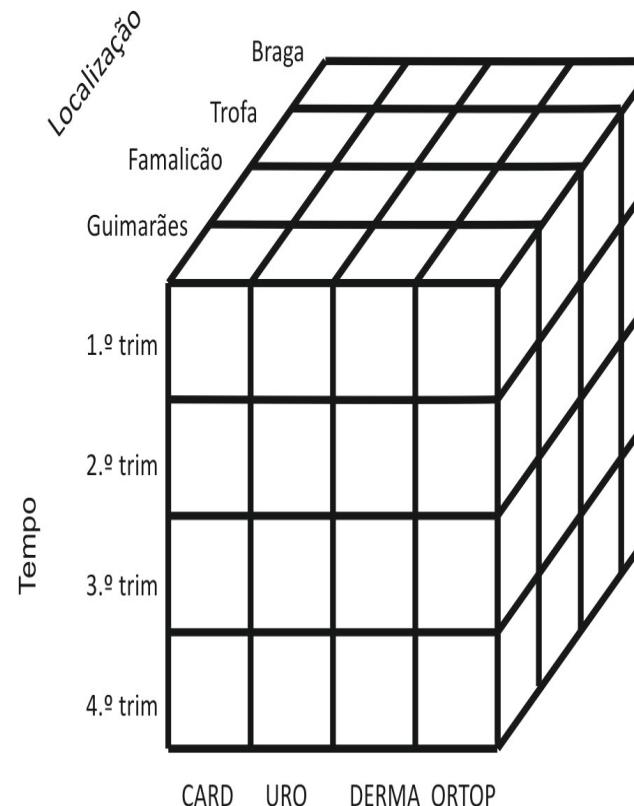
Esquema em floco de neve



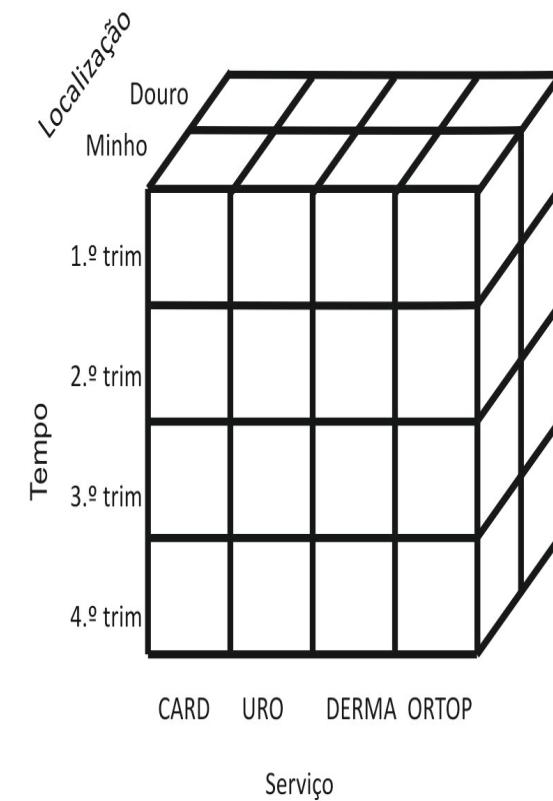
Esquema em constelação



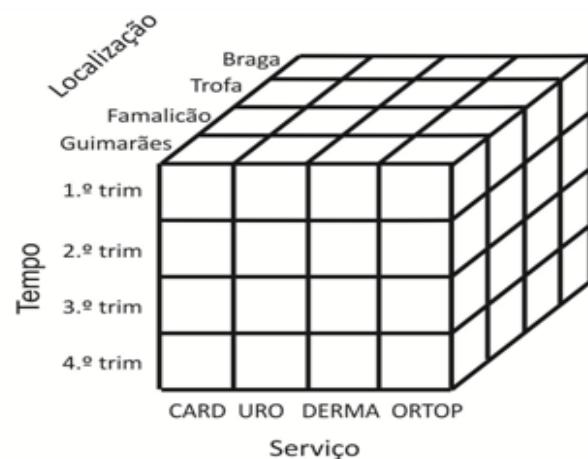
Operações OLAP – Roll-Up



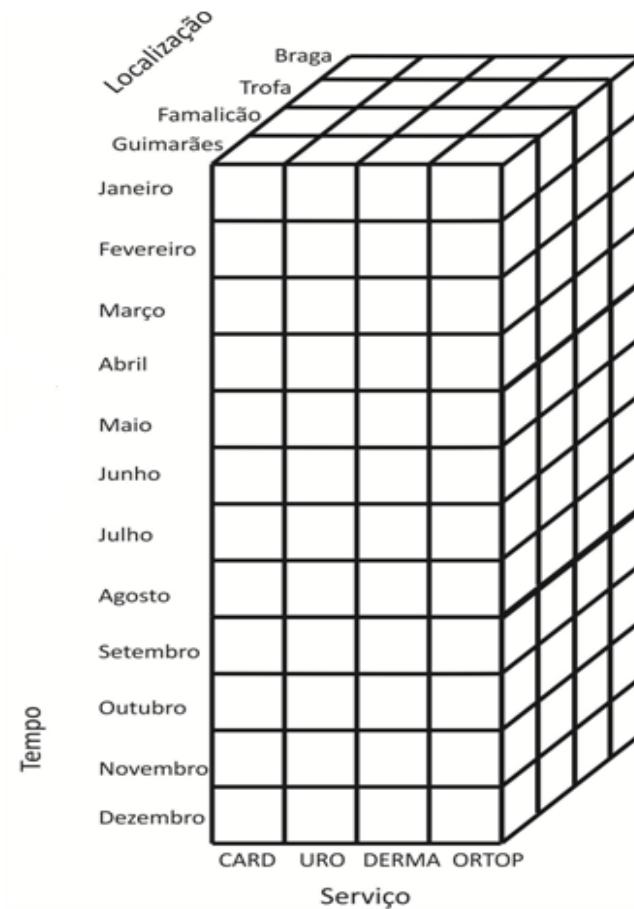
roll-up
na localização
(de cidades para províncias)



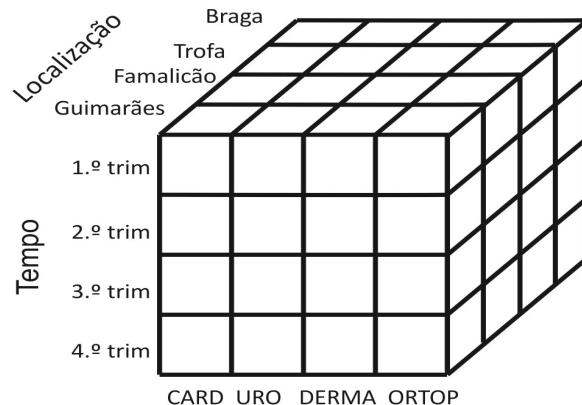
Drill-down



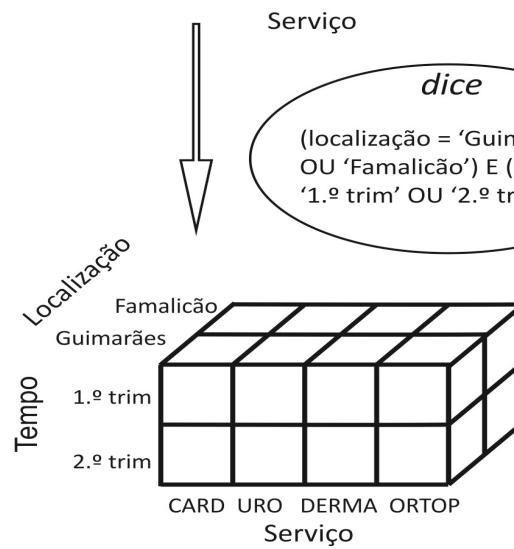
drill-down
no tempo
(de trimestres para meses)



Slide, Dice and Pivot



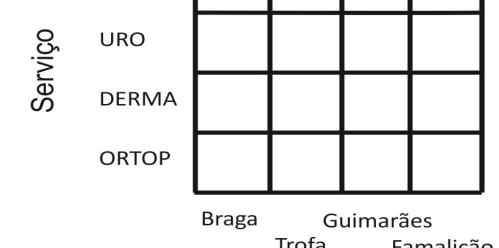
slice
(tempo = '1.º trim')



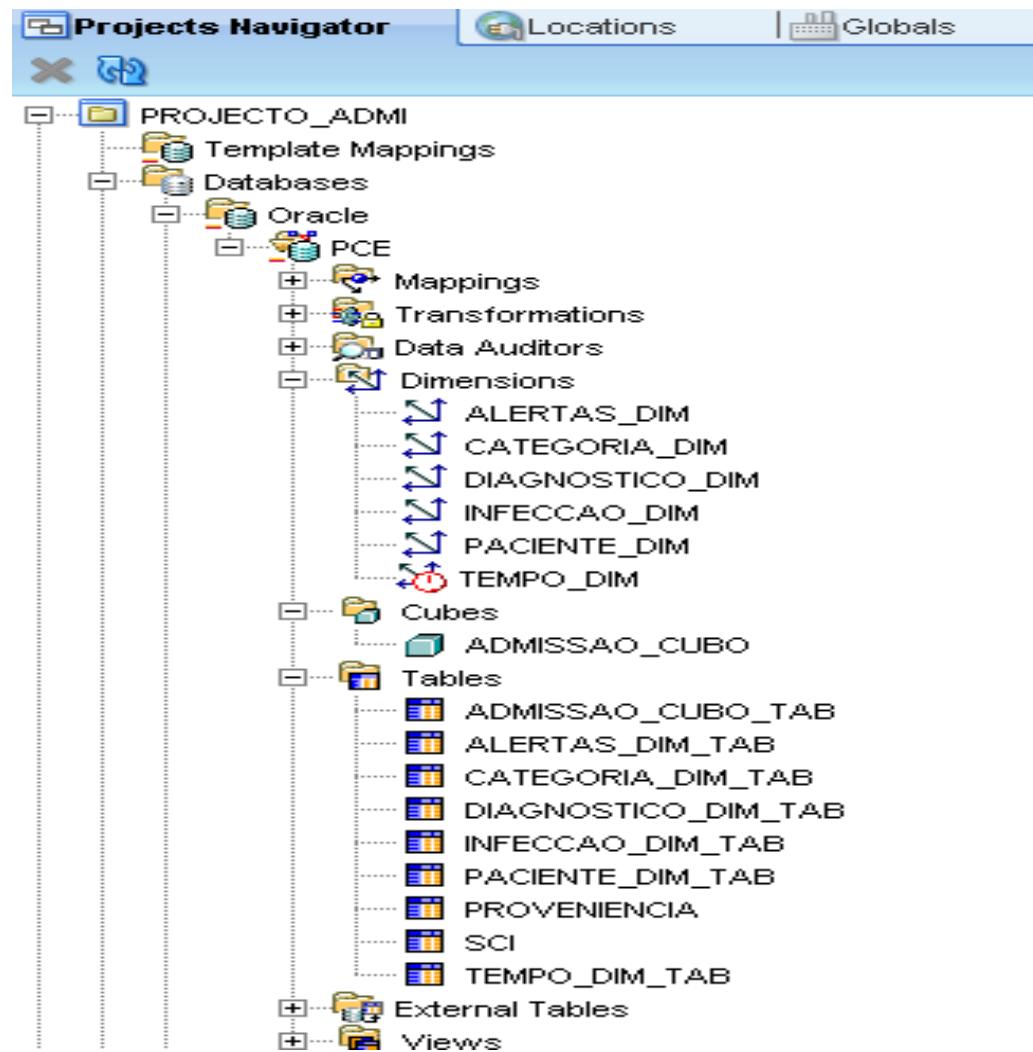
dice
(localização = 'Guimarães'
OU 'Famalicão') E (tempo=
'1.º trim' OU '2.º trim')



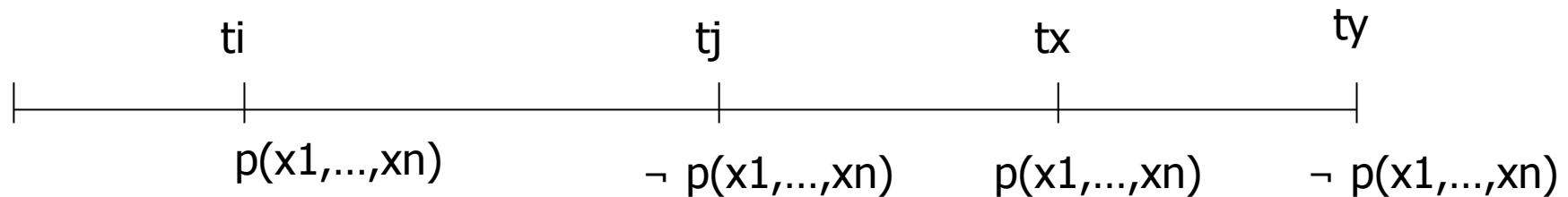
pivot



Oracle Warehouse Builder



A dimensão temporal



Abordagem baseada em informação negativa

Tabela p+

T	A ₁	...	A _n
t _i	x ₁	...	x _n
t _x	x ₁	...	x _n

Tabela p-

T	A ₁	...	A _n
t _j	x ₁	...	x _n
t _y	x ₁	...	x _n

A₁,...,A_n ∈ p no ponto t_k sse

$$\exists t_1 \mid t_1, A_1, \dots, A_n \in p+ \wedge t_1 \leq t_k$$

$$\neg \exists t_2 \mid t_2, A_1, \dots, A_n \in p- \wedge t_2 \leq t_k \wedge t_2 \geq t_1$$

A₁,...,A_n ∉ p no ponto t_k sse

$$\exists t_1 \mid t_1, A_1, \dots, A_n \in p- \wedge t_1 \leq t_k$$

$$\neg \exists t_2 \mid t_2, A_1, \dots, A_n \in p+ \wedge t_2 \leq t_k \wedge t_2 \geq t_1$$

$$\text{ou } \neg \exists t_2 \mid t_2, A_1, \dots, A_n \in p+ \wedge t_2 \leq t_k$$



Abordagem baseada em intervalos

Tabela p*

From	To	A_1	...	A_n
t_i	t_j	x_1	...	x_n
t_x	t_y	x_1	...	x_n

$A_1, \dots, A_n \in p$ no ponto t_k sse

$$\exists t_1, t_2 \mid t_1, t_2, A_1, \dots, A_n \in p^* \wedge t_1 \leq t_k \leq t_2$$

$A_1, \dots, A_n \notin p$ no ponto t_k sse

$$\neg \exists t_1, t_2 \mid t_1, t_2, A_1, \dots, A_n \in p^* \wedge t_1 \leq t_k \leq t_2$$



Optimização de queries

Avaliação do desempenho de uma interrogação (query)

Os procedimentos comuns para avaliar o desempenho de uma interrogação são:

- reescrever a interrogação em álgebra relacional;
- encontrar algoritmos para calcular resultados intermédios da forma mais simples;
- executar os algoritmos e obter os resultados.



Avaliação do desempenho de uma query

181

Estes procedimentos podem esbarrar em dificuldades na medida em que na álgebra relacional podem-se obter expressões equivalentes, há problemas em lidar com sub-interrogações e os tamanhos intermédios obtidos são muito importantes. Os resultados da avaliação dependem da extensão das relações, o que nem sempre é possível saber dada a dinâmica de crescimento das tabelas.



Avaliação do desempenho de uma query

182

Medicos(M,NM,CE,Anos) 50 bytes por tuplo, 100 tuplos

Consultas(M,P,D,Preço) 20 bytes por tuplo, 5000 tuplos

paciente P = 200 só tem 1 consulta!

Interrogação:

```
SELECT m.NM FROM Medicos m,Consultas c where  
c.M = m.M and P = 200 and Anos > 5
```

Tradução para a Álgebra relacional:

$$\Pi_{NM}(\sigma_{P=200 \wedge Anos>5}(Medicos \bowtie Consultas))$$


Avaliação do desempenho de uma query

183

A selecção e a projecção são realizadas após a junção!

Custo da solução: a volta de $100 * 5000 * 2 = 1\ 000\ 000$ tuplos processados!

Nova expressão em Álgebra Relacional:

$$\Pi_{NM}(\sigma_{P=200} \text{Consultas} \bowtie \sigma_{Anos>5} \text{Medicos})$$

É feito um *full table scan* às 2 tabelas, obtendo duas tabelas mais pequenas (cardinalidade respectivamente C1 e 1) e depois é feita a junção entre as 2 tabelas resultantes.

Custo da solução: a volta de $100 + 5000 + C1$ tuplos processados!

Por exemplo admitindo uma redução de cerca de 50% na selecção sobre Medicos ($C1 = 50$), o resultado é $100 + 5000 + 50 = 5150$



Optimização de queries

As expressões seguintes são expressões equivalentes mas com custos muito diferentes:

$$\sigma_C(E_1 \cup E_2) = \sigma_C(E_1) \cup \sigma_C(E_2)$$

$$\sigma_C(E_1 \times E_2) = E_1 \bowtie_C E_2$$

$$\sigma_C(E_1 - E_2) = \sigma_C(E_1) - \sigma_C(E_2)$$

$\sigma_C(E_1 \times E_2) = \sigma_C(E_1) \times E_2$ se o conjunto de atributos de E_1 pertence a C.

$$\sigma_C(E_1 \cap E_2) = \sigma_C(E_1) \cap \sigma_C(E_2)$$

$$\Pi_L(E_1 \bowtie E_2) = \Pi_L(\Pi_{L \cup AE2} \cap_{AE1}(E_1) \bowtie \Pi_{L \cup AE1} \cap_{AE2}(E_2))$$

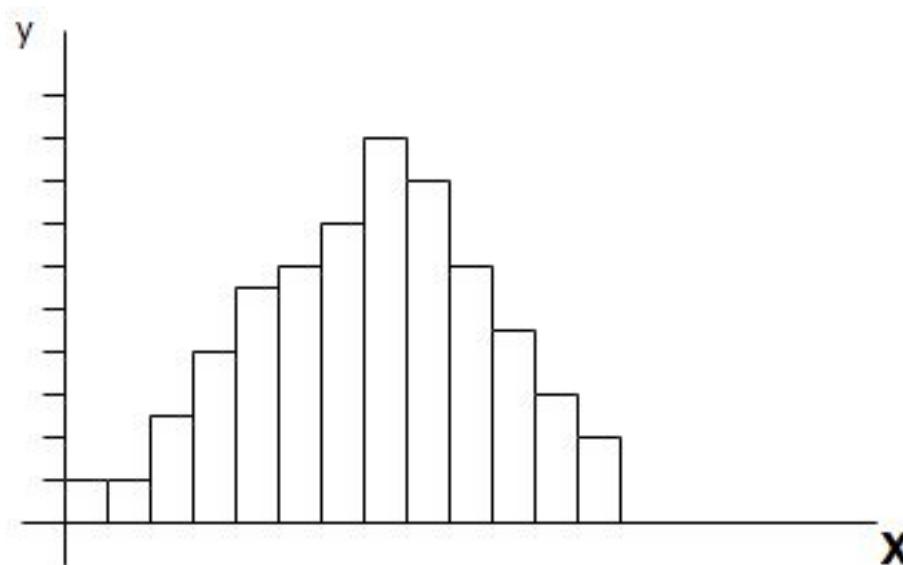
$\Pi_L(\sigma_C(E_1)) = \Pi_L(\sigma_C(\Pi_A(E_1)))$ onde A é o conjunto de atributos mencionados em C.



Histogramas

Os histogramas podem ser usados para avaliar o custo da selecção.

Permitem analisar cardinalidades por intervalos.



Tuning

O Tuning é um processo de acompanhamento da performance das interrogações de forma a escolher bons caminhos.

O tuning pode sugerir a reescrita das interrogações / código, a indexação de algumas tabelas e a eliminação de *full table scans*.



Administração de bases de dados Oracle

187

- Tipos de utilizadores
- Tarefas de um administrador
- Identificação da versão de software
- Segurança e privilégios
- Autenticação
- Criação e manutenção de “Password Files”
- Utilitários de administração



Tipos de utilizadores

- Administradores de bases de dados
- Responsáveis pela segurança
- Administradores de rede
- Programadores de aplicações
- Administradores de aplicações
- Utilizadores de bases de dados



Administradores de bases de dados

Cada base de dados tem pelo menos um administrador (DBA). Considerando que uma base de dados Oracle pode ser um sistema de elevada complexidade e ter muitos utilizadores o administrador pode não ser uma única pessoa. Nestes casos existe um grupo de DBAs que partilham as seguintes responsabilidades:

- Instalar e actualizar o servidor Oracle e as ferramentas aplicacionais
- Alocar espaço de armazenamento e gerir as necessidades futuras em termos de capacidade de armazenamento
- Criar estruturas para o armazenamento (tablespaces) para serem usadas pelas aplicações desenvolvidas pelos programadores
- Criar objectos ou entidades principais (tabelas, views, índices) para executar eficientemente as aplicações
- Modificar a estrutura da base de dados a partir da informação enviada pelos programadores
- Gerir utilizadores e segurança do sistema
- Verificar licenças de software Oracle
- Controlar e monitorizar o acesso dos utilizadores à base de dados
- Monitorizar e optimizar a performance da base de dados
- Gerir cópias de segurança e recuperação de informação perdida
- Manter os dados arquivados em suportes auxiliares
- Contactar a Oracle Corporation para pedir suporte técnico.



Responsáveis pela segurança

Em alguns casos, são atribuídos um ou mais responsáveis pela segurança a uma base de dados. O responsável pela segurança gere utilizadores, controla e monitoriza o acesso à base de dados e mantém a segurança do sistema, libertando o administrador da base de dados dessas responsabilidades.



Administradores de rede

Pode ser contratados um ou mais administradores de rede para administrar os produtos de rede da Oracle, tal como Oracle Net Services. Em ambientes distribuídos e na gestão de bases de dados distribuídas estas responsabilidades são mais visíveis.



Programadores de aplicações

Os programadores desenvolvem, algumas destas tarefas em colaboração com os DBAs:

- Desenhar e desenvolver aplicações
- Desenhar a estrutura da base de dados para correr uma aplicação
- Estimar as necessidades em termos de capacidade de armazenamento
- Especificar alterações à estrutura da base de dados para correr uma aplicação
- Relatar necessidades estruturais aos administradores
- Ajustar as aplicações à base de dados na fase de desenvolvimento
- Estabelecer medidas de segurança aplicacionais durante o desenvolvimento



Administradores de aplicações

Podem ser contratados um ou mais administradores de aplicações para uma aplicação em particular, libertando os DBAs desta responsabilidade. Cada aplicação pode ser o seu próprio administrador.



Utilizadores de bases de dados

Os utilizadores de bases de dados podem ter as seguintes permissões:

- Aceder a, modificar ou apagar dados quando permitido;
- Gerar relatórios a partir dos dados.



Tarefas do administrador de bases de dados

- Tarefa 1: Avaliar o hardware do servidor da base de dados
- Tarefa 2: Instalar software
- Tarefa 3: Projectar a base de dados
- Tarefa 4: Criar e abrir a base de dados
- Tarefa 5: Copiar os dados da base de dados
- Tarefa 6: Gerir os utilizadores do sistema
- Tarefa 7: Implementar o desenho da base de dados
- Tarefa 8: Copiar as funcionalidades da base de dados
- Tarefa 9: Ajustar a performance da base de dados



Tarefa 1: Avaliar o hardware do servidor da base de dados

- Avaliar o número de discos disponíveis para o Oracle e suas bases de dados;
- Avaliar o número de unidades auxiliares para copiar dados das bases de dados (cópias de segurança);
- Avaliar a quantidade de memória disponível para correr as instâncias das bases de dados.



Tarefa 2: Instalar software

O administrador da base de dados deve instalar o servidor de bases de dados e as ferramentas de administração disponíveis, assim como as aplicações desenvolvidas pelos programadores. Em instalações de processamento distribuído, a base de dados é controlada por um processo central (servidor de bases de dados) e as aplicações são instaladas e executadas em computadores de acesso remoto (clientes). Neste caso, nos clientes devem ser instalados os componentes Oracle Net necessários para ligar as máquinas clientes ao servidor que executa o Oracle.

Para requisitos específicos e instruções de instalação deve ser consultada a documentação referente ao sistema operativo do servidor de bases de dados e a documentação referente a cada ferramenta de acesso remoto e driver Oracle Net (e.g. ODBC).



Tarefa 3: Projectar a base de dados

A base de dados deve ser concebida tendo em conta:

- A estrutura lógica de armazenamento de dados
- O modelo conceptual da base de dados
- A estratégia de cópias de segurança definida

Deve-se também considerar de que forma a estrutura lógica de amarzenamento pode afectar:

- A performance do computador que corre Oracle
- A performance da base de dados durante as operações de acesso a dados
- A eficiência dos procedimentos de cópia e recuperação de dados



Tarefa 4: Criar e abrir a base de dados

Depois de conceber o desenho da base de dados deve-se criar a estrutura e disponibilizá-la para uso corrente. A base de dados pode ser criada usando o Database Configuration Assistant (dbca) ou podem ser desenvolvidas “scripts” para esse propósito.



Tarefa 5: Copiar os dados da base de dados

Depois de criar a estrutura da base de dados deve-se implementar a estratégia definida para as cópias de segurança. Deve-se ter em conta:

- a possibilidade de usar “redo log files”;
- fazer o primeiro “backup” completo da base de dados (a guardar);
- implementar mecanismos de “backup” *online* ou *offline*;
- escalonar as cópias para ser efectuadas em intervalos regulares.



Tarefa 6: Gerir os utilizadores do sistema

Depois de copiar a estrutura da base de dados, deve-se fazer uma gestão dos utilizadores de acordo com as licenças de utilização da Oracle. Devem ser criados papéis (“roles”) e permissões (“grants”).



Tarefa 7: Implementar o desenho da base de dados

Depois de criar e iniciar a base de dados, assim como depois de definir e criar os utilizadores, pode ser implementado o plano para a estrutura lógica de dados criando primeiro os “tablespaces” necessários. Depois de completar este passo serão criados os objectos da base de dados.



Tarefa 8: Copiar as funcionalidades da base de dados

203

Depois de completar o processo de definição dos objectos da base de dados, deve-se fazer uma nova cópia de segurança. Em adição ao “backup” de dados correctamente escalonados, deve-se implementar um mecanismo de cópia sempre que a estrutura da base de dados seja alterada.



Tarefa 9: Ajustar a performance da base de dados

Ajustar e optimizar a performance da base de dados é uma tarefa importante para o DBA. Adicionalmente o Oracle fornece uma funcionalidade de gestão de recursos que permite alocar recursos a vários grupos de utilizadores.



Administrador da base de dados – Segurança e privilégios

A conta do sistema operativo para o DBA

O administrador deve ter acesso a comandos do sistema operativo para executar operações fundamentais para a administração da base de dados. A conta do sistema operativo do Administrador deve ter maiores privilégios de que a conta de um simples utilizador Oracle. Os ficheiros Oracle não devem ser armazenados em sub-directórias da conta do Administrador no sistema operativo mas este deve ter acesso a esses ficheiros.

Nomes de utilizador do Administrador da base de dados

Duas contas são criadas automaticamente na base de dados:

- SYS (password inicial: CHANGE_ON_INSTALL)
- SYSTEM (password inicial: MANAGER)



Comentários adicionais sobre segurança

- Durante a instalação do software de base de dados usando o *Database Configuration Assistant (DCBA)*, todos os utilizadores estão bloqueados (locked) e desactivados (expired), excepto SYS, SYSTEM, SCOTT, DBSNMP, OUTLN, AURORA\$JIS\$UTILITY\$, AURORA \$ORB\$UNAUTHENTICATED e OSE\$HTTP\$ADMIN. Para reactivar contas bloqueadas, o DBA deve desbloquear a conta e reatribuir-lhe uma nova palavra de passe.
- O DBCA pede uma palavra de passe para os utilizadores SYS e SYSTEM durante a fase de instalação da base de dados. O comando CREATE DATABASE submetido manualmente também permite atribuir estas duas palavras de passé fundamentais.



Nomes de utilizadores para administrar a base de dados

SYS

- O utilizador **SYS** é criado automaticamente sempre que a base de dados é criada com o papel de DBA.
- Todas as tabelas e views de base para o dicionário de dados da base de dados são armazenados no esquema **SYS**. Estas tabelas e views são fundamentais para o bom funcionamento do sistema e são apenas manipuladas pelo motor Oracle para manter a integridade dos dados. Não é aconselhado criar novas tabelas no esquema **SYS**, podendo apenas alterar-se os parâmetros de armazenamento do tablespace associado. Os utilizadores são devem usar a conta **SYS** para realizar exclusivamente operações de administração.

SYSTEM

- O utilizador **SYSTEM** é criado automaticamente sempre que a base de dados é criada com o papel de DBA.
- O utilizador **SYSTEM** serve para criar tabelas e views adicionais para mostrar informação de carácter administrativo e tabelas ou views internas usadas em ferramentas ou opções do Oracle. Não se deve usar esta valência para o interesse individual de utilizadores.

O papel de DBA (DBA role)

- Um papel pré-definido, **DBA**, é automaticamente criado para cada base de dados. Este papel contém muitos privilégios do sistema, e deve ser atribuído apenas a utilizadores que necessitam de realizar todas as operações de administração.

O papel de DBA não inclui os privilégios SYSDBA e SYSOPER. Estes privilégios são especiais e além de permitir fazer tarefas básicas de administração também permitem criar a base de dados e executar a instâncias de *startup* e *shutdown* de base de dados.



Autenticação do Administrador

O DBA executa operações especiais (e.g. ligar (*start*) e desligar (*shutdown*) a base de dados). Por isso, os utilizadores com privilégios de administração obdecem a um esquema de autenticação com preocupações especiais em matéria de segurança.



Privilégios de administração

Os dois privilégios de administração para operações básicas de administração são o SYSDBA e o SYSOPER. Dependendo do nível de autorização exigido, um desses dois privilégios podem ser atribuídos a um administrador.



SYSDBA e SYSOPER

Privilégio	Operações
SYSDBA Este privilégio oferece ao utilizador as funcionalidades do utilizador SYS	STARTUP e SHUTDOWN ALTER DATABASE: open, mount, backup, e change character set CREATE DATABASE CREATE SPFILE ARCHIVELOG e RECOVERY Inclui o privilégio RESTRICTED SESSION
SYSOPER Este privilégio permite ao utilizador desenvolver tarefas operacionais básicas mas sem a possibilidade de aceder aos dados individuais dos outros utilizadores.	STARTUP e SHUTDOWN CREATE SPFILE ALTER DATABASE OPEN/MOUNT/BACKUP ARCHIVELOG e RECOVERY Inclui o privilégio RESTRICTED SESSION



Ligaçāo com privilégio de Administração

Assuma que o utilizador *scott* executa:

```
CONNECT scott/password  
CREATE TABLE admin_test(name VARCHAR2(20));
```

Mais tarde executa o comando seguinte:

```
CONNECT scott/password AS SYSDBA  
SELECT * FROM admin_test;
```

scott recebe a seguinte mensagem de erro

ORA-00942: table or view does not exist

A tabela foi criada no esquema *scott* e não no esquema *SYS*. O comando correcto seria:

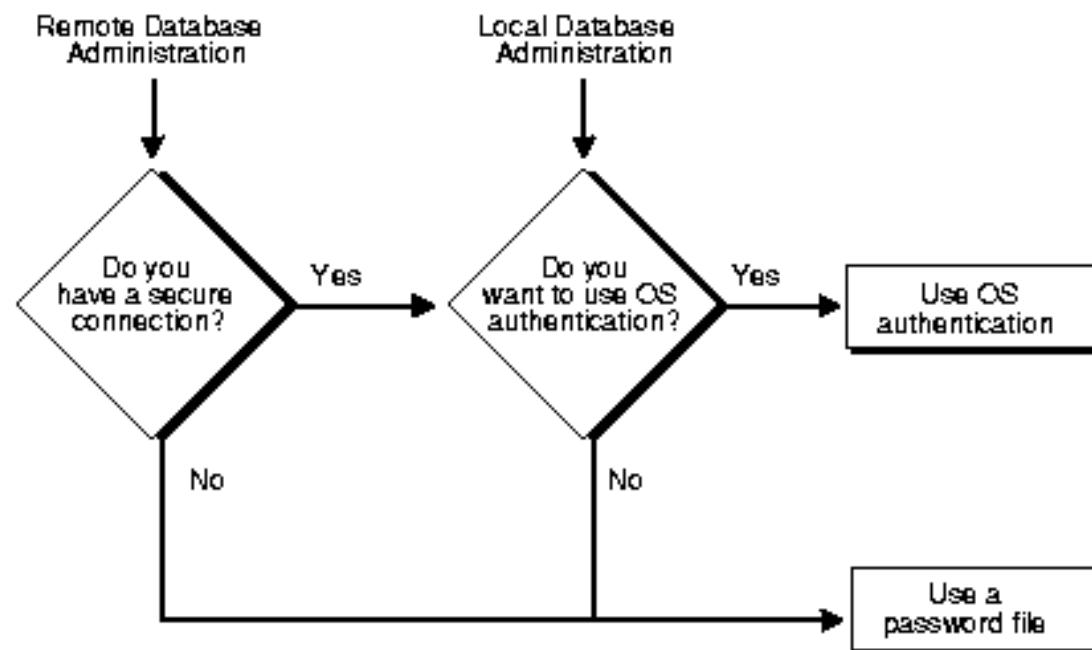
```
SELECT * FROM scott.admin_test;
```



Métodos de autenticação

Existem dois métodos de autenticação:

- Autenticação via Sistema Operativo (OS authentication)
- Password files



Autenticação OS

Para permitir a autenticação de um administrador usando o sistema operativo deve-se:

- Criar uma conta no sistema operativo
- Adicione o utilizador aos grupos OSDBA ou OSOPER do sistema operativo

Verifique se o parâmetro `REMOTE_LOGIN_PASSWORDFILE` é `NONE` (valor por omissão).

Ligaçāo usando autenticação OS

Por exemplo, no caso de ligação local:

CONNECT / AS SYSDBA

CONNECT / AS SYSOPER

No caso de uma ligação remota:

CONNECT /@net_service_name AS SYSDBA

CONNECT /@net_service_name AS SYSOPER



OSDBA e OSOPER

Dois grupos especiais do sistema operativo controlam as ligações dos administradores da base de dados em situação de autenticação via sistema operativo. Estes grupos são designados por OSDBA e OSOPER e dependem do sistema operativo:

Grupo no Sistema Operativo	UNIX	Windows
OSDBA	dba	ORA_DBA
OSOPER	oper	ORA_OPER



Autenticação por Password File

Para autenticação por password file, um administrador deve:

- Criar uma conta de sistema operativo;
- Criar a password file (caso não exista) usando o utilitário ORAPWD:
ORAPWD FILE=filename PASSWORD=password ENTRIES=max_users
- Inicializar o parâmetro REMOTE_LOGIN_PASSWORDFILE com o valor EXCLUSIVE.
- Ligar-se à base de dados com o utilizador SYS (ou outro com privilégio de administrador).
- Criar o utilizador se não existir. Atribuir o privilégio SYSDBA ou SYSOPER ao utilizador:
GRANT SYSDBA to scott;

Este comando adiciona o utilizador à password file, permitindo a ligação como SYSDBA.

Ligação usando autenticação por Password File
CONNECT scott/tiger AS SYSDBA



Utilitários de Administração

SQL*Loader

O SQL*Loader é usado por administradores ou utilizadores normais. Permite carregar dados a partir de ficheiros do sistema (e.g. ficheiros em formato texto) para tabelas de bases de dados Oracle.

Export e Import

Os utilitários *export* e *import* permitem mover data em formato proprietário entre bases de dados Oracle. Por exemplo, ficheiros criados através do *export* podem ser depois carregados através do *import* noutras bases de dados que corram em máquinas com o mesmo sistema operativo ou não.



Exemplos

Utilização de uma máscara e da função TO_DATE :

```
INSERT INTO empregado (n_emp, nome, categoria, entrada, salario)
VALUES (1234, JOSE', 'PROFESSOR',
TO_DATE ('22-02-2002', 'DD-MONTH-YYYY'), 3000);
```

Utilização de restrições ou *constraints*:

```
CREATE TABLE socio
```

(nsocio	NUMBER(9)	PRIMARY KEY,
nome	VARCHAR2(30)	NOT NULL,
num_fiscal	NUMBER(9)	UNIQUE,
a_quota	NUMBER(9)	CHECK (a_quota <5000)
);		



Exemplos

Criação de tabela com partições usando o campo `data_da_venda`:

```
CREATE TABLE venda
  (cod_venda      NUMBER(9)      NOT NULL,
   vendedor       VARCHAR2(30)    NOT NULL,
   produto        VARCHAR2(30)    NOT NULL,
   distrito       VARCHAR2(30)    NOT NULL,
   data_da_venda  DATE           NOT NULL,
   valor          NUMBER(12)      NOT NULL)
```

PARTITION BY RANGE (data_da_venda)

**(PARTITION vendas_jan VALUES LESS THAN
(TO_DATE ('1-Fev-2002','DD-MON-YYYY'))**

**PARTITION vendas_fev VALUES LESS THAN
(TO_DATE ('1-Mar-2002','DD-MON-YYYY'))**

**PARTITION vendas_mar VALUES LESS THAN
(TO_DATE ('1-Abr-2002','DD-MON-YYYY'))**

**PARTITION vendas_q2 VALUES LESS THAN
(TO_DATE ('1-Jul-2002','DD-MON-YYYY'))**

**PARTITION vendas_q3 VALUES LESS THAN
(TO_DATE ('1-Out-2002','DD-MON-YYYY'))**

**PARTITION vendas_q4 VALUES LESS THAN
(TO_DATE ('1-Jan-2003','DD-MON-YYYY'))**

NOT NULL,
NOT NULL)

TABLESPACE users,

TABLESPACE users,

TABLESPACE users,

TABLESPACE users,

TABLESPACE users,

TABLESPACE users);



Exemplos

Criação de sequências:

```
CREATE SEQUENCE cod_seq
    MINVALUE 1
    MAXVALUE 9999999
    START WITH 1
    INCREMENT BY 1
    CACHE 20;
```

Utilização de sequências:

```
INSERT INTO venda
    (cod_venda, vendedor, produto, distrito, data_da_venda,
     valor )
VALUES
    (cod_seq.nextval, 'José Ferreira', 'Artigo B', 'Braga',
     TO_DATE('5-4-2005','DD-MM-YYYY'), 2000);
```



Exemplos

Criação de procedimentos:

```
CREATE OR REPLACE PROCEDURE total_vendas_distrito
(
    aux IN      varchar2,
    total OUT    number
)
AS BEGIN
    SELECT SUM(valor) INTO total FROM venda2 WHERE distrito =
    aux;
END;
```



Exemplos

Invocação de *packages*:

```
SQL> var numero_aleatorio number
```

```
SQL> execute dbms_random.seed(12345678)
```

```
SQL> execute :numero_aleatorio :=dbms_random.random
```

```
SQL> print numero_aleatorio
```

```
NUMERO_ALEATORIO
```

```
-----
```

```
722700502
```



Exemplos

Utilização de *triggers*:

```
CREATE OR REPLACE TRIGGER regista_del
  AFTER DELETE ON vendas FOR EACH ROW
  BEGIN
    INSERT INTO registro_del (operacao, data, utilizador)
      VALUES ('DELETE', SYSDATE, USER);
  END;
```

```
CREATE OR REPLACE TRIGGER reg_logon
  AFTER LOGON ON DATABASE
  BEGIN
    INSERT INTO registro_del(operacao,data,utilizador)
      VALUES('LOGON',sysdate,user);
  END;
```



Exemplos

Criação de sinónimos públicos:

```
CREATE PUBLIC SYNONYM empregados FOR scott.emp;
```

Criação de Ligações (*database links*):

```
CREATE DATABASE LINK vendas.com USING 'sales_us';
CREATE PUBLIC DATABASE LINK vendas
    CONNECT TO scott IDENTIFIED BY tiger
    USING 'sales_us';
```

Utilizar *database links*:

```
SELECT * FROM employee@vendas;
```

Atribuição de privilégios sobre a forma de *roles*:

```
CREATE ROLE estagiario NOT IDENTIFIED;
GRANT INSERT ON tomanix.vendas TO "estagiario";
GRANT SELECT ON tomanix.vendas TO "estagiario";
GRANT UPDATE ON tomanix.vendas TO "estagiario";
GRANT estagiario TO scott;
```



Exemplos

Consultar a informação sumária sobre a SGA (System Global Area):

```
SELECT * FROM v$sga;
```

NAME	VALUE
-----	-----
Fixed Size	282576
Variable Size	83886080
Database Buffers	33554432
Redo Buffers	532480



Exemplos

Consultar a informação sobre os parâmetros dinâmicos da SGA:

```
SELECT * FROM v$sga_dynamic_components;
COMPONENT          CURRENT_SIZE  MIN_SIZE  MAX_SIZE
-----
shared pool          50331648   50331648   50331648
large pool           8388608    8388608    8388608
buffer cache         25165824   25165824   25165824
```

Alterar o parâmetro SHARED_POOL_SIZE:

```
ALTER SYSTEM SET shared_pool_size = 50331648;
```

Forçar um *checkpoint*:

```
ALTER SYSTEM CHECKPOINT;
```

Consultar a tabela de *jobs*:

```
SELECT * FROM all_jobs;
```

Consultar os *jobs* em execução:

```
SELECT * FROM all_jobs;
```

Consultar a lista de *tablespaces* existentes na base de dados:

```
SELECT * FROM dba_tablespaces;
```

Consultar a lista dos *free extents*, ou seja, espaço livre nos vários *tablespaces*:

```
SELECT * FROM dba_free_space;
```



Exemplos

Atribuir um determinado *tablespace* a um utilizador:

```
SELECT * FROM dba_free_space; CREATE USER manuela IDENTIFIED BY
    password_da_manuela
    DEFAULT TABLESPACE USERS;
ALTER USER manuela QUOTA 0 ON SYSTEM;
```

Alterar o *tablespace* por omissão para um utilizador já existente:

```
ALTER USER fernando DEFAULT TABLESPACE USERS;
```

Criação de um novo *undo tablespace*:

```
CREATE UNDO TABLESPACE UNDOTBS2
DATAFILE 'D:\ORACLE\ORADATA\ORA920\UNDOTBS2.DBF' SIZE 50M;
```

Criação de um *temporary tablespace*:

```
CREATE TEMPORARY TABLESPACE TEMP2
TEMPFILE 'D:\ORACLE\ORADATA\ORA920\TEMP2.ora' SIZE 20M;
```

Criação de utilizadores, indicando qual o seu *temporary tablespace* por omissão:

```
ALTER TABLESPACE "USERS_LOCAL_MANAGED" READ ONLY;
```



Exemplos

```
CREATE USER manuela IDENTIFIED BY password_da_manuela
  DEFAULT TABLESPACE USERS
  TEMPORARY TABLESPACE TEMP2;
```

**Atribuir um *temporary* TABLESPACE TEMP2;
tablespace a um utilizador já existente:**
ALTER USER fernando TEMPORARY

Criação de um *tablespace locally managed*:
CREATE TABLESPACE "USERS_LOCAL_MANAGED"
 DATAFILE 'D:\ORACLE\ORADATA\ORA920\USERS.DBF'
 SIZE 5M EXTENT MANAGEMENT LOCAL;

Criação de um *tablespace dictionary managed*
CREATE TABLESPACE "USERS_DICT_MANAGED"
 DATAFILE 'D:\ORACLE\ORADATA\ORA920\USERS.DBF'
 SIZE 5M EXTENT MANAGEMENT DICTIONARY;

Alterar um *tablespace* para *read-write*:
ALTER TABLESPACE "USERS_LOCAL_MANAGED" READ WRITE ;



Exemplos

Recriar índices noutro *tablespace*:

```
ALTER INDEX livro_idx REBUILD TABLESPACE INDICES;
```

Mover uma tabela para outro *tablespace*:

```
ALTER TABLE livro MOVE TABLESPACE DATA1;
```

Definir uma quota num *tablespace*:

```
ALTER USER fernando QUOTA 5M ON USERS;
```

Colocar um *tablespace offline*:

```
ALTER TABLESPACE DATA1 OFFLINE NORMAL;
```



Exemplos

Consultar a lista dos nomes, tamanhos e *tablespaces* associados aos *datafiles*:

```
SELECT file_name, blocks, tablespace_name  
      FROM dba_data_files;
```

Consultar a lista dos *control files* e a sua localização:

```
SELECT * FROM V$CONTROLFILE;
```

Fazer um *backup* do *control file*:

```
ALTER DATABASE BACKUP CONTROLFILE TO 'd:\backups\control.bkp';
```

Criação de um *script SQL* que permita recriar o *control file*:

```
ALTER DATABASE BACKUP CONTROLFILE TRACE;
```



Exemplos

Consultar a informação sobre os *redo log groups* e os seus membros:

```
SELECT * FROM V$LOGFILE;
```

Criação de um novo grupo de *redo logs*:

```
ALTER DATABASE ADD LOGFILE GROUP 5  
  ('c:\oradata\log5a.rdo',d:\Oracle\oradata\ora920\log5b.rdo')  
  SIZE 200K;
```

Adicionar um membro (*redo logs*) a um grupo:

```
ALTER DATABASE ADD LOGFILE MEMBER  
  'c:\oradata\log1b.rdo' TO GROUP 1;
```

Remover membros (*redo logs*) do grupo:

```
ALTER DATABASE DROP LOGFILE MEMBER 'c:\oradata\log1b.rdo';
```



Exemplos

Forçar uma mudança de *redo log group*:

```
ALTER SYSTEM SWITCH LOGFILE;
```

Criação de um *tablespace* usando *Oracle Managed Files* (OMF):

```
ALTER SYSTEM SET DB_CREATE_FILE_DEST = 'C:\ORADATA';
CREATE TABLESPACE dados3 DATAFILE AUTOEXTEND ON
  MAXSIZE 500M;
```

Criação de um *tablespace* usando OMFs com dois *datafiles*:

```
ALTER SYSTEM SET DB_CREATE_FILE_DEST = 'C:\ORADATA';
CREATE TABLESPACE indices2 DATAFILE SIZE 5M, SIZE 5M;
```



Exemplos

Adicionar um *datafile* a um *tablespace* usando OMFs:

```
ALTER SYSTEM SET DB_CREATE_FILE_DEST = 'C:\ORADATA';
ALTER TABLESPACE dados3 ADD DATAFILE AUTOEXTEND ON
MAXSIZE 100M;
```

Consultar a vista *USER_OBJECTS* que mostra todos os objectos pertencentes ao utilizador que está ligado à base de dados:

```
SELECT * FROM USER_OBJECTS;
```

Consultar a vista *DBA_OBJECTS* que dá uma vista geral da base de dados mostrando os objectos pertencentes ao utilizador SYS:

```
SELECT * FROM DBA_OBJECTS;
```

Consultar a vista *ALL_OBJECTS* que dá uma perspectiva de todos os objectos da base de dados que o utilizador corrente tem acesso:

```
SELECT * FROM ALL_OBJECTS;
```

Adicionar um *datafile* a um *tablespace* usando OMFs:

```
ALTER SYSTEM SET DB_CREATE_FILE_DEST = 'C:\ORADATA';
ALTER TABLESPACE dados3 ADD DATAFILE AUTOEXTEND ON
MAXSIZE 100M;
```



Exemplos

Consultar a vista USER_OBJECTS que mostra todos os objectos pertencentes ao utilizador que está ligado à base de dados:

```
SELECT * FROM USER_OBJECTS;
```

Consultar a vista DBA_OBJECTS que dá uma vista geral da base de dados mostrando os objectos pertencentes ao utilizador SYS:

```
SELECT * FROM DBA_OBJECTS;
```

Consultar a vista ALL_OBJECTS que dá uma perspectiva de todos os objectos da base de dados que o utilizador corrente tem acesso:

```
SELECT * FROM ALL_OBJECTS;
```

Consultar a vista V\$SGA que mostra informação sobre a *System Global Area (SGA)*:

```
SELECT * FROM V$SGA;
```

NAME	VALUE
Fixed Size	453512
Variable Size	113246208
Database Buffers	25165824
Redo Buffers	667648



Exemplos

Consultar a vista V\$TABLESPACE que mostra os vários *tablespaces* na base de dados:

```
SELECT * FROM V$TABLESPACE;
```

TS#	NAME	INC
-----	-----	---
0	SYSTEM	YES
1	UNDOTBS1	YES
9	USERS	YES
2	TEMP	YES



Exemplos

Consultar o espaço livre em cada *tablespace* usando a view DBA_FREE_SPACE:

```
SELECT tablespace_name, sum(bytes) FROM dba_free_space  
    GROUP BY tablespace_name;
```

TABLESPACE_NAME	SUM(BYTES)
CWMLITE	11141120
DRSYS	10813440
EXAMPLE	131072
INDX	26148864
ODM	11206656
SYSTEM	4325376
TOOLS	4128768
UNDOTBS1	204865536
USERS	26148864
XDB	196608

10 linhas seleccionadas.



Exemplos

Criação um novo *datafile* no *tablespace* USERS com a dimensão inicial de 5 M que cresce dinamicamente em pedaços de 1280 K e sem limite de tamanho:

```
ALTER TABLESPACE "USERS" ADD  
  DATAFILE 'E:\ORACLE\ORADATA\ORA9I\USERS02.DBF'  
  SIZE 5M REUSE AUTOEXTEND  
  ON NEXT 1280K MAXSIZE UNLIMITED;
```

Parar o crescimento automático do *datafile*:

```
ALTER DATABASE  
  DATAFILE 'D:\ORACLE\ORADATA\ORA920\USERS1.DBF'  
  AUTOEXTEND OFF;
```

Diminuir o tamanho de um *datafile*:

```
ALTER DATABASE  
  DATAFILE 'E:\ORACLE\ORADATA\ORA9I\USERS05.DBF'  
  RESIZE 40M;
```



Exemplos

Mover *datafiles* entre *tablespaces*:

Antes de mais, fazer backup da base de dados.

Partido do princípio que se pretende mover um dos *datafiles* do *tablespace* DATA1, ver a lista dos *datafiles* que fazem parte desse *tablespace*.

```
SELECT file_name, bytes FROM dba_data_files  
WHERE tablespace_name = 'DATA1';
```

FILE_NAME	BYTES
E:\ORACLE\ORADATA\ORA9I\DATA1.DBF	5242880
E:\ORACLE\ORADATA\ORA9I\DATA2.DBF	5242880



Exemplos

Pôr o *tablespace DATA1 offline*.

```
ALTER TABLESPACE "DATA1" OFFLINE NORMAL;
```

Copiar o *datafile* pretendido para o novo local, usando os comandos do sistema operativo.

Indicar ao SGBD que o *datafile* está noutro local.

```
ALTER TABLESPACE data1  
  RENAME DATAFILE 'E:\ORACLE\ORADATA\ORA9I\DATA2.DBF'  
    TO 'C:\ORADATA\DATA2.DBF';
```

Trazer o *tablespace DATA1* novamente para *online*.

```
ALTER TABLESPACE "DATA1" ONLINE;
```

Fazer novo backup da base de dados.

Remover *datafiles*:



Exemplos

Criar um novo *tablespace*.

```
CREATE TABLESPACE "LIXO"  
  DATAFILE 'E:\ORACLE\ORADATA\ORA9I\LIXO.DBF'  
  SIZE 5M
```

Mover o *datafile* para o novo *tablespace*, ver exemplo anterior.

Remover o *tablespace*.

Embora não obrigatório, é preferível pôr o *tablespace* *offline* antes de o remover.



Exemplos

ALTER TABLESPACE "LIXO" OFFLINE NORMAL;

É possível indicar ao *Oracle* que para além de apagar o *tablespace* deve remover fisicamente os *datafiles* que lhe pertencem com a cláusula INCLUDING CONTENTS AND DATAFILES.

```
DROP TABLESPACE "LIXO" INCLUDING CONTENTS AND DATAFILES;
```

Mover tabelas entre *tablespaces*:

```
ALTER TABLE MEUS_CDS MOVE TABLESPACE DADOS1;
```

Mover índices entre *tablespaces*:

```
ALTER INDEX SCOTT.NOME REBUILD TABLESPACE DADOS1;
```

Consultar a *view DBA_TABLES* de forma a saber que objectos lhe pertencem:

```
SELECT table_name, tablespace_name FROM dba_tables  
WHERE table_name = 'MEUS_CDS';
```

TABLE_NAME

TABLESPACE_NAME

MEUS_CDS

DADOS1



Exemplos

Consultar a view DBA_INDEXES de forma a saber que objectos lhe pertencem:

```
SELECT index_name,tablespace_name FROM dba_indexes  
WHERE index_name='NOME';
```

INDEX_NAME	TABLESPACE_NAME
NOME	DADOS1

Renomear um *tablespace*:

```
ALTER TABLESPACE "USERS" RENAME TO "USERS_OLD";
```



Exemplos

Verificar se diferentes sistemas operativos têm o mesmo *byte order* podemos usar uma *view* da base de dados, V\$TRANSPORTABLE_PLATFORM:

```
SELECT * FROM V$TRANSPORTABLE_PLATFORM;
```

PLATFORM_ID	PLATFORM_NAME	ENDIAN_FORMAT
1	Solaris[tm] OE (32-bit)	Big
2	Solaris[tm] OE (64-bit)	Big
7	Microsoft Windows IA (32-bit)	Little
10	Linux IA (32-bit)	Little
6	AIX-Based Systems (64-bit)	Big
3	HP-UX (64-bit)	Big
5	HP Tru64 UNIX	Little
4	HP-UX IA (64-bit)	Big
11	Linux IA (64-bit)	Little
15	HP Open VMS	Little
8	Microsoft Windows IA (64-bit)	Little
9	IBM zSeries Based Linux	Big
13	Linux 64-bit for AMD	Little
16	Apple Mac OS	Big
12	Microsoft Windows 64-bit for AMD	Little



Exemplos

Transportar um *tablespace* entre duas bases de dados *Oracle10g* instaladas respectivamente em *MS Windows* e em *Linux*:

Colocar o *tablespace* a transportar em modo só leitura, *read only*.

SQL > ALTER TABLESPACE DADOS5 READ ONLY;

Exportar a metainformação sobre o *tablespace*. A partir de uma janela MS-DOS, executar o comando exp, de notar que o utilizador é o sys com o privilégio de sysdba com a respectiva *password*, fazer:

```
D:\> exp tablespaces=dados5 transport_tablespace=y file=tbs_dados5.dmp
Export: Release 10.1.0.2.0 - Production on Qua Jul 14 11:05:15 2004
Copyright (c) 1982, 2004, Oracle. All rights reserved.
Username: sys as sysdba
Password:
Connected to: Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production
With the Partitioning, OLAP and Data Mining options
Export done in WE8MSWIN1252 character set and AL16UTF16 NCHAR character set
Note: table data (rows) will not be exported
About to export transportable tablespace metadata...
For tablespace DADOS5 ...
. exporting cluster definitions
. exporting table definitions
.. exporting table TAB1
.. exporting table TAB2
.. exporting table TAB3
.. exporting table TAB4
. exporting referential integrity constraints
. exporting triggers
. end transportable tablespace metadata export
Export terminated successfully without warnings.
```



Exemplos

Copiar o *tablespace* e o ficheiro de *export* com a metainformação para o novo sistema, neste caso copiar os ficheiros "DADOS5.DBF" e o "tbs_dados5.dmp".

Esta operação pode ser feita usando, por exemplo, o **FTP**; neste caso não esquecer de copiar em modo binário.

Importar o *tablespace* com a metainformação, fazer:

```
$ imp tablespaces=dados5 transport_tablespace=y file=tbs_dados5.dmp
datafiles='DADOS5.DBF'
```

Import: Release 10.1.0.2.0 - Production on Qua Jul 14 11:58:07 2004

Copyright (c) 1982, 2004, Oracle. All rights reserved.

Username: sys as sysdba

Password:

Connected to: Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production

With the Partitioning, OLAP and Data Mining options

Export file created by EXPORT:V10.01.00 via conventional path

About to import transportable tablespace(s) metadata...

import done in WE8MSWIN1252 character set and AL16UTF16 NCHAR character set

- . importing SYS's objects into SYS

- . importing SCOTT's objects into SCOTT

- .. importing table "TAB1"

- .. importing table "TAB2"

- .. importing table "TAB3"

- .. importing table "TAB4"

- . importing SYS's objects into SYS

Import terminated successfully without warnings.



Exemplos

Finalmente não esquecer de voltar a colocar o *tablespace on line*, para permitir operações de escrita.

```
SQL> ALTER TABLESPACE DADOSS5 READ WRITE;
```

Para visualizar as operações de *sort* numa determinada instância, recorrendo à vista **V\$SORT_USAGE** em conjunto com a **V\$SESSION**, é possível recolher informação sobre que utilizadores estão a executar operações de *sort*:

```
SELECT 'vs.username', vs.sid, vs.serial#, vsu.contents  
  FROM v$session vs, v$sort_usage vsu  
 WHERE vs.saddr = vsu.session_addr;
```

USERNAME	SID	SERIAL#	CONTENTS
SCOTT	23	75	PERMANENT



Exemplos

Identificar utilizadores a executar operações de sort:

```

SELECT substr(vs.username,1,20) "Utilizador BD",
       substr(vs.osuser,1,20)   "Utilizador SO",
       substr(vsn.name,1,20)    "Tipo de Sort",
       vss.value
  FROM v$session vs,
       v$sesstat vss,
       v$statname vsn
 WHERE (vss.statistic#=vsn.statistic#)
   AND (vs.sid = vss.sid)
   AND (vsn.name like '%sort%')
 ORDER BY 2,3;

```

Utilizador BD	Utilizador so	Tipo de Sort	VALUE
SYSTEM	tomanix	sorts (disk)	0
SYS	tomanix	sorts (disk)	0
SYSTEM	tomanix	sorts (disk)	0
SYSTEM	tomanix	sorts (memory)	17
SYSTEM	tomanix	sorts (memory)	67
SYS	tomanix	sorts (memory)	18
SYSTEM	tomanix	sorts (rows)	107
SYSTEM	tomanix	sorts (rows)	1241
SYS	tomanix	sorts (rows)	64



Exemplos

Adicionar um *datafile* temporário a um *tablespace* temporário:

```
ALTER TABLESPACE "TEMP"  
ADD TEMPFILE 'D:\ORACLE\ORADATA\ORA92\TEMP02.DBFb'  
SIZE 5M;
```

Utilizar vários *temporary tablespaces*:

```
CREATE TEMPORARY TABLESPACE TBSTEMP1  
TEMPFILE 'e:\oracle\oradata\ora10g\tbstemp1.dbf'  
SIZE 5M  
TABLESPACE GROUP TBSTEMPGRP1;
```

```
CREATE TEMPORARY TABLESPACE TBSTEMP2  
TEMPFILE 'e:\oracle\oradata\ora10g\tbstemp2.dbf'  
SIZE 15M  
TABLESPACE GROUP TBSTEMPGRP1;
```



Exemplos

Alterar o *tablespace* temporário por omissão na base de dados:

```
ALTER DATABASE ora10g DEFAULT TEMPORARY TABLESPACE TBSTEMPGRP1;
```

Consultar a lista de grupos de *tablespaces*, temporários ou não:

SQL> SELECT * FROM DBA_TABLESPACE_GROUPS;
GROUP_NAME TABLESPACE_NAME
TBSTEMPGRP1 TBSTEMP1
TBSTEMPGRP1 TBSTEMP2

Criar um *undo tablespace*:

```
CREATE UNDO TABLESPACE "NOVOUNDOTBS" DATAFILE '/home/ora10g/oradata/
orcl/novoundotbs.dbf' SIZE 5M;
```

Controlar o tempo que a instância retém os dados no *undo tablespace* através do parâmetro de arranque *UNDO_RETENTION*:

```
ALTER SYSTEM SET UNDO_RETENTION = 3600;
```



Exemplos

Monitorar e recolher estatísticas do *undo tablespace*, podemos usar a vista **V\$UNDOSTAT**:

```
SQL> SELECT TO_CHAR(BEGIN_TIME, 'DD/MM/YYYY HH24:MI:SS')
  BEGIN_TIME,
  TO_CHAR (END_TIME, 'DD/MM/YYYY HH24:MI:SS') END_TIME,
  UNDOTSN,UNDOBLKS, TXNCOUNT, MAXCONCURRENCY AS "MAXCON"
  FROM V$UNDOSTAT;
```

BEGIN_TIM	END_TIME	UNDOTSN	UNDO	BLKS	TXN	COUNT	MAXCON
-----	-----	-----	-----	-----	-----	-----	-----
15/07/2004	11:19:08	15/07/2004 11:28:36	1	3		46	1
15/07/2004	11:09:08	15/07/2004 11:19:08	1	4		71	1
15/07/2004	10:59:08	15/07/2004 11:09:08	1	38		109	1
15/07/2004	10:49:08	15/07/2004 10:59:08	1	6		52	2
15/07/2004	10:39:08	15/07/2004 10:49:08	1	253		386	3
15/07/2004	10:29:08	15/07/2004 10:39:08	1	566		1095	3

6 rows selected.



Exemplos

Criar um novo segmento de *rollback*:

```
CREATE ROLLBACK SEGMENT "RBS01"  
    TABLESPACE "RBS01";
```

```
ALTER ROLLBACK SEGMENT "RBS01" ONLINE;
```

**Consulta de informação sobre os vários
segmentos de *rollback*:**

```
SELECT SEGMENT_NAME, TABLESPACE_NAME, BYTES  
FROM DBA_SEGMENTS  
WHERE SEGMENT_TYPE = 'ROLLBACK';
```



Exemplos

Verificar a lista de utilizadores na base de dados e o seu estado:

```
SQL> SELECT USERNAME, ACCOUNT_STATUS FROM DBA_USERS;
```

USERNAME	ACCOUNT_STATUS
SYS	OPEN
SYSTEM	OPEN
OUTLN	EXPIRED & LOCKED
DBSNMP	OPEN
SYSMAN	OPEN
MGMT_VIEW	OPEN
MDSYS	EXPIRED & LOCKED
ORDSYS	EXPIRED & LOCKED
EXFSYS	EXPIRED & LOCKED
DMSYS	EXPIRED & LOCKED
WMSYS	EXPIRED & LOCKED
WKSYS	EXPIRED & LOCKED
WK_TEST	EXPIRED & LOCKED
CTXSYS	EXPIRED & LOCKED
ANONYMOUS	EXPIRED & LOCKED
XDB	EXPIRED & LOCKED
WKPROXY	EXPIRED & LOCKED
ORDPLUGINS	EXPIRED & LOCKED
SI_INFORMTN_SCHEMA	EXPIRED & LOCKED
OLAPSYS	EXPIRED & LOCKED
SCOTT	OPEN
BI	EXPIRED & LOCKED
PM	EXPIRED & LOCKED
MDDATA	EXPIRED & LOCKED
IX	EXPIRED & LOCKED
SH	EXPIRED & LOCKED
DIP	EXPIRED & LOCKED
OE	EXPIRED & LOCKED
HR	EXPIRED & LOCKED

29 rows selected.



Exemplos

Procedimento que cria uma vista com o nome de **DBA_USER_PRIVS** que lista todos os utilizadores e respectivos privilépios:

```

SET ECHO off
REM NAME: TFSPPRVW.SQL
REM USAGE:"@path/tfsprvw"
REM -----
REM REQUIREMENTS:
REM Should be run as SYS
REM -----
REM AUTHOR:
REM Anonymous
REM Copyright 1995, Oracle Corporation
REM -----
REM PURPOSE:
REM The following script will generate a view
REM DBA_USER_PRIVS
REM which may be queried to obtain information
REM about which
REM privileges are available to a given user, or which
REM users
REM enjoy a particular privilege.
REM -----

```

REM EXAMPLE:	USERNAME	ROLENAME	PRIVILEGE
REM -----	-----	-----	-----
REM SCOTT	CONNECT		ALTER SESSION
REM SCOTT	CONNECT		CREATE CLUSTER
REM SCOTT	CONNECT		CREATE DATABASE LINK
REM SCOTT	CONNECT		CREATE TABLE
REM SCOTT	CONNECT		CREATE VIEW
REM SCOTT	DBA		ALTER ANY CLUSTER
REM SCOTT	DBA		ALTER ANY INDEX
REM SCOTT	DBA		ALTER ANY PROCEDURE
REM SCOTT	DBA		ALTER ANY ROLE
REM SCOTT	DBA		ALTER ANY SEQUENCE
REM SCOTT	DBA		ALTER PROFILE
REM SCOTT	DBA		UPDATE ANY TABLE
REM SCOTT	RESOURCE		CREATE CLUSTER
REM SCOTT	RESOURCE		CREATE PROCEDURE
REM SCOTT	RESOURCE		CREATE SEQUENCE
REM SCOTT	RESOURCE		CREATE TABLE
REM SCOTT	RESOURCE		CREATE TRIGGER
REM SCOTT			UNLIMITED TABLESPACE
REM			
REM -----			

REM DISCLAIMER:

```

REM This script is provided for educational purposes only. It
REM is NOT supported by Oracle World Wide Technical Support.
REM The script has been tested and appears to work as
REM intended. You should always run new scripts on a test
REM instance initially.
REM -----

```

REM Main text of script follows:



Exemplos

```
CREATE OR REPLACE VIEW DBA_USER_PRIVS (USERNAME, ROLENAME,
PRIVILEGE) AS
SELECT DECODE(SA1.GRANTEE#, 1, 'PUBLIC', U1.NAME), SUBSTR(U2.NAME,
1,20),
SUBSTR(SPM.NAME,1,27)
FROM SYS.SYSAUTH$ SA1, SYS.SYSAUTH$ SA2, SYS.USER$ U1,
SYS.USER$ U2, SYS.SYSTEM_PRIVILEGE_MAP SPM
WHERE SA1.GRANTEE# = U1.USER#
AND SA1.PRIVILEGE# = U2.USER#
AND U2.USER# = SA2.GRANTEE#
AND SA2.PRIVILEGE# = SPM.PRIVILEGE
UNION
SELECT U.NAME, NULL, SUBSTR(SPM.NAME,1,27)
FROM SYS.SYSTEM_PRIVILEGE_MAP SPM, SYS.SYSAUTH$ SA,
SYS.USER$ U
WHERE SA.GRANTEE#=U.USER#
AND SA.PRIVILEGE#=SPM.PRIVILEGE
```



Exemplos

Criação de utilizadores:

```
CREATE USER "RUI"
PROFILE "CONTABILIDADE"
IDENTIFIED BY "password" PASSWORD EXPIRE
DEFAULT TABLESPACE "USERS"
TEMPORARY TABLESPACE "TEMP"
QUOTA 5 M ON "DADOS2"
QUOTA 1 M ON "INDICES"
QUOTA 1 M ON "USERS"
ACCOUNT UNLOCK;
GRANT "CONNECT" TO "RUI";
```

Alterar a quota do utilizador:

```
ALTER USER "RUI" QUOTA 50 M ON "DADOS2";
```

Alterar a password do utilizador:

```
ALTER USER "RUI" IDENTIFIED BY "xpto";
```

Remover utilizadores:

```
DROP USER "RUI" CASCADE;
```

Bloquear o acesso a uma conta:

```
ALTER USER "RUI" ACCOUNT LOCK;
```

Consultar a informação sobre todos os utilizadores da base de dados:

```
SELECT * FROM DBA_USERS;
```

Limitar utilização recursos da base de dados:

```
ALTER SYSTEM SET RESOURCE_LIMIT = TRUE;
```



Exemplos

Criação de perfis:

```
CREATE PROFILE "CONTABILIDADE" LIMIT  
    CPU_PER_SESSION 3600  
    CPU_PER_CALL DEFAULT  
    CONNECT_TIME UNLIMITED  
    IDLE_TIME 15  
    SESSIONS_PER_USER 1;
```

Alteração de perfis:

```
ALTER PROFILE "CONTABILIDADE" LIMIT  
    LOGICAL_READS_PER_SESSION 1000;
```

Remover um *profile* existente:

```
DROP PROFILE "CONTABILIDADE" CASCADE;
```

Atribuir um perfil a um utilizador:

```
ALTER USER "MIGUEL" PROFILE "CONTABILIDADE";
```



Exemplos

Visualizar informação sobre perfis e sua utilização:

```
SELECT * FROM DBA_PROFILES;
```

PROFILE	RESOURCE_NAME	RESOURCE	LIMIT
DEFAULT	COMPOSITE_LIMIT	KERNEL	UNLIMITED
CONTABILIDADE	COMPOSITE_LIMIT	KERNEL	DEFAULT
DEFAULT	FAILED_LOGIN_ATTEMPTS	PASSWORD	UNLIMITED
CONTABILIDADE	FAILED_LOGIN_ATTEMPTS	PASSWORD	DEFAULT
DEFAULT	SESSIONS_PER_USER	KERNEL	UNLIMITED
CONTABILIDADE	SESSIONS_PER_USER	KERNEL	1
DEFAULT	PASSWORD_LIFE_TIME	PASSWORD	UNLIMITED
CONTABILIDADE	PASSWORD_LIFE_TIME	PASSWORD	DEFAULT
DEFAULT	CPU_PER_SESSION	KERNEL	UNLIMITED

Impor um limite de 60 dias antes que a mesma *password* possa ser reutilizada:

```
ALTER PROFILE "CONTABILIDADE"
  LIMIT PASSWORD_REUSE_MAX 60;
```



Exemplos

Procedimento para impor complexidade às palavras-chave:

```

Rem
Rem $Header: utlpwdmg.sql 31-aug-2000.11:00:47 nireland Exp $
Rem
Rem utlpwdmg.sql
Rem
Rem Copyright (c) Oracle Corporation 1996, 2000. All Rights
Rem
Rem NAME
Rem utlpwdmg.sql - script for Default Password Resource
Rem
Rem DESCRIPTION
Rem This is a script for enabling the password management
Rem features by setting the default password resource limits.
Rem
Rem NOTES
Rem This file contains a function for minimum checking of
Rem password complexity. This is more of a sample function
Rem that the customer can use to develop the function for
Rem actual complexity checks that the customer wants to make Rem on the new password.
Rem
Rem MODIFIED (MM/DD/YY)
Rem nireland 08/31/00 - Improve check for
Rem           username=password. #1390553
Rem nireland 06/28/00 - Fix null old password test.
Rem           #1341892
Rem asurpur 04/17/97 - Fix for bug479763
Rem asurpur 12/12/96 - Changing the name of
Rem           password_verify_function
Rem asurpur 05/30/96 - New script for default password
Rem           management
Rem asurpur 05/30/96 - Created
Rem

```



Exemplos

```
-- This script sets the default password resource parameters
-- This script needs to be run to enable the password features.
-- However the default resource parameters can be changed based
-- on the need.
-- A default password complexity function is also provided.
-- This function makes the minimum complexity checks like
-- the minimum length of the password, password not same as the
-- username, etc. The user may enhance this function according --- to the need.
-- This function must be created in SYS schema.
-- connect sys/<password> as sysdba before running the script
CREATE OR REPLACE FUNCTION verify_function
(username varchar2,
password varchar2,
old_password varchar2)
RETURN boolean IS
n boolean;
m integer;
differ integer;
isdigit boolean;
ischar boolean;
ispunct boolean;
digitarray varchar2(20);
punctarray varchar2(25);
chararray varchar2(52);
```



Exemplos

```

BEGIN
    digitarray:= '0123456789';
    chararray:=
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
    punctarray:='!"#$%&(``*+,/-;:<=>?_';
    --
    -- Esta condição verifica se a password é igual ao username,
    -- uma das regras básicas para impedir a utilização de
    -- passwords óbvias
    --
    -- Check if the password is same as the username
    IF NLS_LOWER(password) = NLS_LOWER(username) THEN
        raise_application_error(-20001, 'Password same as or similar
to user');
    END IF;
    --
    -- Verificação do tamanho mínimo da password, para uma
    -- política
    -- mais rigorosa podemos impor um tamanho de 8 caracteres
    -- substituindo o 4 pelo 8
    --
    -- Check for the minimum length of the password
    IF length(password) < 4 THEN
        raise_application_error(-20002, 'Password length less than 4');
    END IF;

```



Exemplos

```
--  
-- Verifica se uma password é demasiado simples; é possível  
-- manter um  
-- dicionário de palavras consideradas óbvias, no nosso caso  
-- poderíamos  
-- acrescentar 'benfica', 'sporting', 'porto', 'boavista',  
-- etc  
--  
-- Check if the password is too simple. A dictionary of words  
-- may be  
-- maintained and a check may be made so as not to allow the  
-- words  
-- that are too simple for the password.  
IF NLS_LOWER(password) IN ('welcome', 'database', 'account',  
'user',  
    'password', 'Oracle', 'computer', 'abcd') THEN  
    raise_application_error(-20002, 'Password too simple');  
END IF;  
--  
-- Verifica se a password contém pelo menos um algarismo e um  
-- carácter de pontuação de forma a tornar mais difícil  
-- adivinhar  
--  
-- Check if the password contains at least one letter, one
```



Exemplos

```
-- digit and one
-- punctuation mark.
-- 1. Check for the digit
  isdigit:=FALSE;
  m := length(password);
  FOR i IN 1..10 LOOP
    FOR j IN 1..m LOOP
      IF substr(password,j,1) = substr(digitarray,i,1) THEN
        isdigit:=TRUE;
        GOTO findchar;
      END IF;
    END LOOP;
  END LOOP;
  IF isdigit = FALSE THEN
    raise_application_error(-20003, 'Password should \ contain at least one digit,
one character and one punctuation');
  END IF;
-- 2. Check for the character
<<findchar>>
  ischar:=FALSE;
```



Exemplos

```

FOR i IN 1..length(chararray) LOOP
    FOR j IN 1..m LOOP
        IF substr(password,j,1) = substr(chararray,i,1) THEN
            ischar:=TRUE;
            GOTO findpunct;
        END IF;
    END LOOP;
END LOOP;
IF ischar = FALSE THEN
    raise_application_error(-20003, 'Password should contain
\ at least one digit, one character and one punctuation');
END IF;
-- 3. Check for the punctuation
<<findpunct>>
ispunct:=FALSE;
FOR i IN 1..length(punctarray) LOOP
    FOR j IN 1..m LOOP
        IF substr(password,j,1) = substr(punctarray,i,1) THEN
            ispunct:=TRUE;
            GOTO endsearch;
        END IF;
    END LOOP;
END LOOP;
IF ispunct = FALSE THEN
    raise_application_error(-20003, 'Password should contain at least one \
digit, one character and one punctuation');
END IF;
<<endsearch>>

```



Exemplos

-- Verifica se a *password* é diferente da anterior em pelo menos 3 caracteres

```
--  
-- Check if the password differs from the previous password by at least 3 letters  
IF old_password IS NOT NULL THEN  
    differ := length(old_password) - length(password);  
    IF abs(differ) < 3 THEN  
        IF length(password) < length(old_password) THEN  
            m := length(password);  
        ELSE  
            m := length(old_password);  
        END IF;  
        differ := abs(differ);  
        FOR i IN 1..m LOOP  
            IF substr(password,i,1) != substr(old_password,i,1)  
            THEN  
                differ := differ + 1;  
            END IF;  
        END LOOP;  
        IF differ < 3 THEN  
            raise_application_error(-20004, 'Password should \  
            differ by at least 3 characters');  
        END IF;  
    END IF;  
END IF;  
-- Everything is fine; return TRUE ;  
RETURN(TRUE);  
END;
```



Exemplos

```
-- This script alters the default parameters for Password
-- Management
-- This means that all the users on the system have Password
-- Management
-- enabled and set to the following values unless another
-- profile is
-- created with parameter values set to different value or
-- UNLIMITED is created and assigned to the user.
```

```
ALTER PROFILE DEFAULT LIMIT
PASSWORD_LIFE_TIME 60
PASSWORD_GRACE_TIME 10
PASSWORD_REUSE_TIME 1800
PASSWORD_REUSE_MAX UNLIMITED
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LOCK_TIME 1/1440
PASSWORD_VERIFY_FUNCTION verify_function;
```

Alterar a própria *password* usando o comando **PASSWORD**:

```
SQL> password
A alterar a senha para MIGUEL
Senha antiga:
Senha nova:
Reintroduza a senha nova:
Senha foi alterada
SQL>
```



Exemplos

Alterar a *password* de outro utilizador usando o comando PASSWORD:

```
SQL> password miguel
A alterar a senha para MIGUEL
Senha nova:
Reintroduza a senha nova:
Senha foi alterada
SQL>
```

Criação de um utilizador que será validado pelo sistema operativo:

```
SQL> CREATE USER "OPS$RUI" PROFILE "DEFAULT"
```

```
IDENTIFIED EXTERNALLY
DEFAULT TABLESPACE "USERS"
ACCOUNT UNLOCK;
```

```
SQL> GRANT "CONNECT" TO "OPS$RUI";
```

Criação de um ficheiro de *passwords*:

Criar o ficheiro de *passwords* usando o utilitário **ORAPWD**, numa janela *Command Prompt* no *Windows* ou no utilizador *Oracle* no *Linux*:

```
$ orapwd file=ficheiro password=plim entries=10
```

Verificar o parâmetro de inicialização da instância:

REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE



Exemplos

Entrar na base de dados com privilégios AS SYSDBA usando a validação do sistema operativo:

```
$ sqlplus /nolog
```

```
SQL > connect / as sysdba
```

Entrar na base de dados com privilégios AS SYSDBA usando o ficheiro de *passwords*:

```
$ sqlplus /nolog
```

```
SQL > connect user/password as sysdba
```

Dar a outros utilizadores da base de dados a possibilidade de ligação como SYSDBA, usando o ficheiro de *passwords*:

```
GRANT SYSDBA TO RUI;
```

Revogar o privilégio de ligação AS SYSDBA:

```
REVOKE SYSDBA FROM RUI;
```

Consultar a lista de utilizadores com privilégio SYSDBA:

```
SQL> SELECT * FROM V$PWFILE_USERS;
```

USERNAME	SYSDB	SYSOP
SYS	TRUE	TRUE

USERNAME	SYSDB	SYSOP
SYS	TRUE	TRUE

Entrar na base de dados com privilégios AS SYSOPER usando a validação do sistema operativo:

```
$ sqlplus /nolog
```

```
SQL > connect / as sysoper
```

Entrar na base de dados com privilégios AS SYSOPER usando o ficheiro de *passwords*:

```
$ sqlplus /nolog
```

```
SQL > connect user/password as sysoper
```

Dar a outros utilizadores da base de dados a possibilidade de ligação como SYSOPER, usando o ficheiro de *passwords*:

```
GRANT SYSOPER TO RUI;
```

Revogar o privilégio de ligação AS SYSOPER:

```
REVOKE SYSOPER FROM RUI;
```



Exemplos

Consultar a lista de utilizadores com privilégio SYSOPER:

```
SQL> SELECT * FROM V$PWFILE_USERS;
USERNAME          SYSDB      SYSOP
-----
SYS                TRUE       TRUE
OPERADOR           FALSE      TRUE
```

Entrar na base de dados com privilégios normais de utilizador usando a validação do sistema operativo:

```
$ sqlplus /nolog
SQL > connect /
$ sqlplus /nolog
SQL > connect rui/"password_do_rui"
$ sqlplus rui/"password_do_rui"@ora920
```

Atribuir privilégios de sistema:

```
GRANT ALTER ANY TABLE TO "MIGUEL";
```

Revogar privilégios de sistema:

```
REVOKE ALTER ANY TABLE FROM "MIGUEL";
```

Atribuir um privilégio sobre um determinado objecto a um utilizador:

```
GRANT SELECT ON "SCOTT"."EMP" TO "MIGUEL";
```

Dar permissão de cedência de permissões:

```
GRANT UPDATE (SAL) ON "SCOTT"."EMP" TO "MIGUEL"
    WITH GRANT OPTION;
```



Exemplos

Criação um *role*:

```
CREATE ROLE "RECURSOS_HUMANOS";
```

Atribuir privilégios ao *role* sobre um ou mais objectos:

```
GRANT SELECT, INSERT, UPDATE ON "HR"."COUNTRIES"
    TO "RECURSOS_HUMANOS";
```

```
GRANT SELECT, INSERT, UPDATE ON "HR"."DEPARTMENTS"
    TO "RECURSOS_HUMANOS";
```

```
GRANT SELECT, INSERT, UPDATE ON "HR"."EMPLOYES"
    TO "RECURSOS_HUMANOS";
```

```
GRANT SELECT, INSERT, UPDATE ON "HR"."JOBS"
    TO "RECURSOS_HUMANOS";
```

Atribuir privilégios de sistema a *roles*:

```
GRANT CREATE SESSION TO "RECURSOS_HUMANOS";
```

Atribuir outros *roles* a *roles*:

```
GRANT "RESOURCE" TO "RECURSOS_HUMANOS";
```

Atribuir *passwords* a *roles*:

```
CREATE ROLE "GESTOR_DE_RH" IDENTIFIED BY password;
```

Alterar os *roles* activos numa sessão:

```
SET ROLE GESTOR_DE_RH IDENTIFIED BY password;
```



Exemplos

Consultar quais os *roles* que tem disponíveis na sessão:

```
SQL> SELECT * FROM SESSION_ROLES;
```

```
ROLE
```

```
GESTOR_DE_RH
```

Consultar a lista de privilégios na sessão:

```
SQL> SELECT * FROM SESSION_PRIVS;
```

```
PRIVILEGE
```

```
CREATE SESSION
```

```
ALTER SESSION
```

```
CREATE TABLE
```

```
ALTER ANY TABLE
```

```
CREATE CLUSTER
```

```
CREATE SYNONYM
```

```
CREATE VIEW
```

```
CREATE SEQUENCE
```

```
CREATE DATABASE LINK
```

Alterar a lista de *roles* por omissão para um determinado utilizador:

```
ALTER USER MIGUEL
```

```
    DEFAULT ROLE CONNECT, GESTOR_DE_RH;
```

Atribuir *roles* a utilizadores juntamente com o privilégio de poderem, por sua vez, atribuí-los a outros utilizadores:

```
GRANT "GESTOR_DE_RH" TO "MIGUEL"
```

WITH ADMIN OPTION;

Remover *roles*:

```
SQL> DROP ROLE GESTOR_DE_RH;
```



Exemplos

Consultar a lista de utilizadores com privilégios na tabela “EMPLOYEES”:

```
SQL> SELECT GRANTEE, OWNER, PRIVILEGE
      FROM DBA_TAB_PRIVS
     WHERE TABLE_NAME = 'EMPLOYEES';
```

GRANTEE	OWNER	PRIVILEGE
OE	HR	SELECT
OE	HR	REFERENCES
RECURSOS_HUMANOS	HR	INSERT
RECURSOS_HUMANOS	HR	SELECT
RECURSOS_HUMANOS	HR	UPDATE
RECURSOS_HUMANOS	HR	ALTER
RECURSOS_HUMANOS	HR	DELETE

Consultar a lista de privilégios de sistema do role “DBA”:

```
SELECT * FROM ROLE_SYS_PRIVS WHERE ROLE = 'DBA';
```

ROLE	PRIVILEGE	ADM
DBA	AUDIT ANY	YES
DBA	DROP USER	YES
DBA	RESUMABLE	YES
DBA	ALTER USER	YES
DBA	ANALYZE ANY	YES
DBA	BECOME USER	YES
DBA	CREATE ROLE	YES
DBA	CREATE RULE	YES
DBA	CREATE TYPE	YES
DBA	CREATE USER	YES



Exemplos

Activar o *audit* na base de dados:

```
ALTER SYSTEM SET AUDIT_TRAIL = DB SCOPE=SPFILE;
```

Activar a auditoria às operações de *login*:

```
AUDIT SESSION;
```

Consultar os registos de auditoria:

```
SELECT USERNAME, ACTION_NAME, RETURNCODE, TIMESTAMP
      FROM DBA_AUDIT_SESSION WHERE USERNAME = 'SCOTT';
```

USERNAME	ACTION_NAME	RETURNCODE	TIMESTAMP
----------	-------------	------------	-----------

SCOTT	LOGON	1017	02.09.25
-------	-------	------	----------

~

```
SELECT USERNAME, ACTION_NAME, RETURNCODE, TIMESTAMP
      FROM DBA_AUDIT_SESSION WHERE USERNAME = 'PEDRO';
```

USERNAME	ACTION_NAME	RETURNCODE	TIMESTAMP
----------	-------------	------------	-----------

PEDRO	LOGOFF	0	02.09.25
-------	--------	---	----------

Registrar só as tentativas com êxito ou só as tentativas falhadas:

```
SQL> AUDIT SESSION WHENEVER SUCCESSFUL;
```

```
SQL> AUDIT SESSION WHENEVER NOT SUCCESSFUL;
```

Registrar todas as operações que afectem tabelas:

```
AUDIT TABLE;
```



Exemplos

Consultar a lista de códigos e seus significados das várias operações sujeitas a auditoria:

SQL> SELECT * FROM AUDIT_ACTIONS;

ACTION NAME

ACTION	NAME
0	UNKNOWN
1	CREATE TABLE
2	INSERT
3	SELECT
4	CREATE CLUSTER
5	ALTER CLUSTER

Activar o registo de ocorrências especificando o utilizador e a operação a registar:

AUDIT DELETE TABLE BY MIGUEL;

Desactivar uma determinada auditoria:

NOAUDIT TABLE;



Exemplos

Registo de acções sobre objectos da base de dados (*object audits*):

```
AUDIT DELETE ON PEDRO.CLIENTE BY ACCESS;
```

Consultar a informação sobre o resultado do *audit*:

```
SELECT USERNAME, TIMESTAMP, OBJ_NAME,
       ACTION_NAME, RETURNCODE
     FROM DBA_AUDIT_OBJECT WHERE OBJ_NAME = 'CLIENTE';
----- ----- ----- ----- -----
USERNAME   TIMESTAM  OBJ_NAME   ACTION_NAME   RETURNCODE
----- ----- ----- ----- -----
DELETE          0
SYSTEM 02.09.28  CLIENTE    DELETE          0
SCOTT 02.09.28  CLIENTE    DELETE        2004
```

Registrar somente as tentativas sem sucesso de acessos ou alterações aos dados da tabela:

```
AUDIT SELECT, INSERT, UPDATE, DELETE ON PEDRO.CLIENTE
      BY ACCESS
      WHENEVER NOT SUCCESSFUL
```



INSERT INTO ... SELECT

```
create table klinhas
(ki number(3),
ano varchar2(4),
mes varchar2(2),
cod_servico_unidade varchar2(1),
cod_servico_unidade_prev varchar2(1),
cod_unidade_inter number(3),
cod_unidade_inter_prev number(3),
cod_especialidade number(5),
cod_especialidade_prev number(5),
tint number(8),
tnom number(8),
tdenom number(8)
);
```

```
insert into klinhas(ki,ano,mes,cod_servico_unidade,
cod_servico_unidade_prev,cod_unidade_inter,
cod_unidade_inter_prev,
cod_especialidade,cod_especialidade_prev,tint)
select 1,to_char(dta_saida,'yyyy'),to_char
(dta_saida,'mm'),cod_servico_unidade,
cod_servico_unidade_prev,cod_unidade_inter,
cod_unidade_inter_prev,cod_especialidade,
cod_especialidade_prev, count(distinct int_episodio)
from transferencias
where to_char(dta_saida,'yyyy') = '2009'
group by 1,to_char(dta_saida,'yyyy'),
to_char(dta_saida,'mm'),
cod_servico_unidade,cod_servico_unidade_prev,
cod_unidade_inter,cod_unidade_inter_prev,
cod_especialidade,cod_especialidade_prev;
```



CREATE PROCEDURE

```
CREATE OR REPLACE PROCEDURE CRIA_KI (iniki_ number, finiki_ number) is
Begin
declare ki_ number(3); ano_ varchar2(4);
mes_ varchar2(2); cod_servico_unidade_ varchar2(1);
cod_servico_unidade_prev_ varchar2(1); cod_unidade_inter_ number(5);
cod_unidade_inter_prev_ number(5); cod_especialidade_ number(5);
cod_especialidade_prev_ number(5);
tnom_ number(8); tdenom_ number(8);
exp1_ varchar2(200); exp2_ varchar2(200); exp3_ varchar2(200);
tipo_ number(2); interid_ number(8); interid1_ number(8); interid2_ number(8);
Begin
....
End;
End;
```



Begin ... End

Begin

```
ki_ := iniki_ - 1;  
while (ki_ < finiki_) loop  
    ki_ := ki_ + 1;  
    if (kpi_ = 1) then  
        exp1_ := '.....';  tipo_ := 1;  
    end if;  
  
    ....  
    if (kpi_ = 12) then  
        exp1_ := '...';  
        tipo_ := 5;  
    end if;  
    begin ... end;  
end;
```



Cursos

begin

```
declare cursor c1 is select ano, mes, cod_servico_unidade, cod_servico_unidade_prev, nvl
(cod_unidade_inter,0), nvl(cod_unidade_inter_prev,0), cod_especialidade,
cod_especialidade_prev from kilinhas where ki = ki_;      begin
```

open c1;

fetch c1 into ano_,mes_,cod_servico_unidade_,cod_servico_unidade_prev_,
cod_unidade_inter_, cod_unidade_inter_prev_, cod_especialidade_, cod_especialidade_prev_;

while c1%FOUND **loop**

 SELECT ...

 UPDATE ...

 COMMIT

```
        fetch c1 into ano_, mes_, cod_servico_unidade_, cod_servico_unidade_prev_,
cod_unidade_inter_, cod_unidade_inter_prev_, cod_especialidade_,
cod_especialidade_prev_;
```

end loop;

close c1;

end;

