Bases de Dados Relacionais

José Machado e António Abelha
Departamento de Informática
Escola de Engenharia
Universidade do Minho
jmac@di.uminho.pt

1. Introdução

Este texto didáctico é um apoio à disciplina de bases de dados, do quarto ano da licenciatura em Engenharia de Sistemas e Informática, e não dispensa à assistência às aulas teóricas e teorico-práticas da disciplina.

2. Sistemas de Informação

Podem ser dadas definições alternativas do conceito de Sistema de Informação (SI). Numa perspectiva estrutural, um 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 apoio à realização de inferências; assim, o SI facilita a criação e troca de significados que servem propósitos definidos socialmente tais como o controlo, o dar sentido e a argumentação.

Os dados constituem a forma mais primitiva de representar factos ou afirmações verdadeiras ou falsas. É um estabelecimento do nível de cognição muito próximo da realidade, sem a inclusão de qualquer heurística que visa a incorporação de formas de raciocíno. A informação é um conjunto de factos ou afirmações enquadradas num determinado contexto, veiculando intenções e dando algum significado aos factos. O conhecimento enquadra-se num contexto mais alargado, em comunidades e paradigma com história, cultura ou tradição, e é uma estrutura duradoura de factos com significado. Traduz-se num conjunto de crenças declaradas sobre um assunto com reinvindicações legítimas de verdade ou de correcção. Um SI é um sistema onde a informação deve ser capturada, representada, armazenada e processada no contexto da organização (estruturadas em secções, com ou sem níveis hierárquicos, com uma cultura própria e procedimentos comuns), das pessoas (com uma determinada formação, motivação e ergonomia) e da tecnologia, resolvendo os problemas criados por factores internos e externos. Deverá também interagir com o Ambiente Externo, com características políticas, demográficas, económicas e sociais.

O SI pode ser dividido em diferentes tipos, de acordo com a hierarquia da organização (Figura 1). Os utilizadores envolvidos estão ligados a cada um desses níveis de acordo com as funções exercidas. As áreas aplicacionais divergem na base da pirâmide mas convergem a medida que envolvem sistemas e pessoas de nível superior, integrando e partilhando dados, informação e procedimentos (Enterprise Resource Planning – ERP).

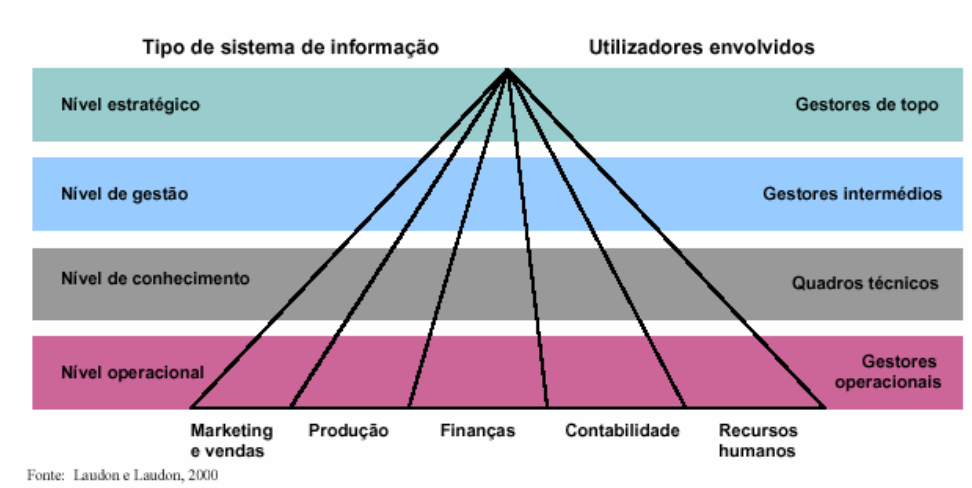


Figura 1 : Tipos de sistemas de informação

A base da pirâmide anterior envolve sistemas de processos transacionais (Transaction Processing System), enquanto os níveis superiores envolvem actividades ditas de gestão (Management Information System) (Figura 2)

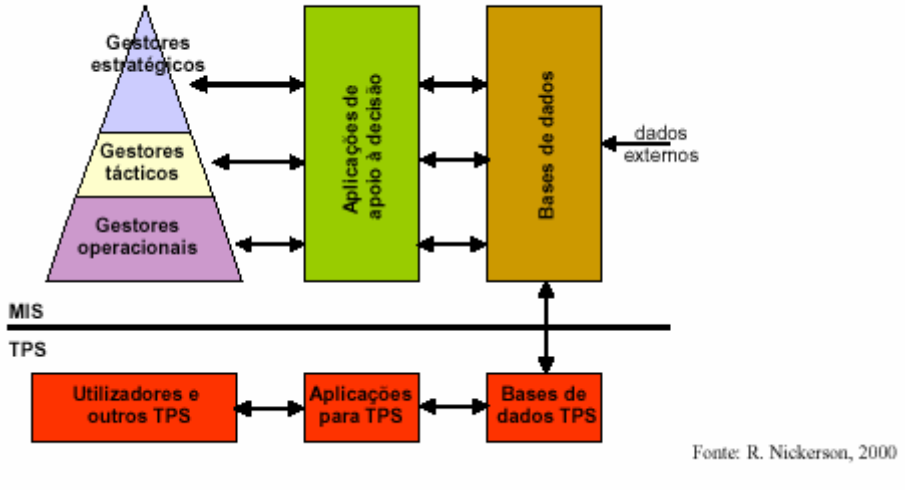


Figura 2 – TPS vs MIS

3. Enterprise Resource Planning (ERP)

Designa-se por **ERP** um conjunto de aplicações informáticas para o apoio de uma forma integrada às funções financeira, logística, produção ou de recursos humanos numa organização. Este assume uma perspectiva dos processos da organização e as várias funções ou actividades são integradas por intermédio de uma base de dados comum. O ERP beneficia:

- da integração de dados: a introdução/alteração de dados num módulo (função) é automaticamente reflectida nos outros módulos;

- da redução de dados redundantes, o que evita a reintrodução de dados;
- da melhoria da comunicação entre funções pela automação da transferência de dados;
- do acesso em tempo real a dados operacionais;
- do reforço do funcionamento fluído dos processos de negócio

O ERP é todavia um modelo que tem custos associados elevados, apesenta um processo de implementação crítico que exige muitos recursos em tempo e pessoas, além de novas metodologia de abordagem e resolução de problemas. A configurabilidade é limitada.

Um dos aspectos mais críticos nos relacionamentos inter-organizacionais tem a ver com o envio / recepção de informação entre sistemas de informação de empresas ou organizações, possivelmente heterógeneos e com objectivos distintos. Existem diferentes técnicas para facilitar essas operações, tais como o email, o voice mail, a partilha de ficheiros, a videoconferência, a webconferência, a Internet, as comunicações móveis e o Electronic Data Interchange (EDI)[1]. A diferença fundamental entre estes tipos de comunicação reside nas diferenças funcionais ao nível destas quatro características:

- **Transporte:** enviar e receber mensagens;
- **Linguagem:** significado da mensagem;
- **Ontologia:** estruturação das conversações;
- **Arquitectura:** interligação entre os sistemas de acordo com os protocolos

4. Resolução de Problemas

A resolução de problemas faz-se dividindo o problema em três sub-problemas:

1) a identificação do problema e os seus dados de entrada;
    a. capturar dados;
    b. introduzir dados;
    c. validar dados;
2) a transformação dos dados de entrada em resultados, com recurso à informação armazenada em bases de dados;
    a. processamento;

[1] O EDI tem como objectivo a comunicação electrónica de dados entre empresas ou organizações. A troca de dados é realizada entre aplicações informáticas heterogéneas, baseadas em ferramentas de controlo, gestão e bases de dados diferentes e em sistemas operativos possivelmente diferentes, com pouca intermediação humana. O EDI facilita a troca de dados e as transacções, usando protocolos (e.g., XML) que possibilitam as comunicações; i.e., envio e recepção de informação. O único requisito em termos de equipamento é a garantia de infra-estruturas de comunicação que permitam o envio de ficheiros com informação e transacções, através de redes de valor acrescentado, da internet ou ligações ponto a ponto.

i. processar;
ii. tomar decisões;

b. armazenamento;
i. guardar;
ii. aceder;
iii. ordenar;
iv. actualizar;

3) a apresentação dos resultados;
a. produzir saída para o monitor;
b. produzir saída para a impressora.

A Figura 3 ilustra esta metodologia numa perspectiva reactiva, em que o processador apenas reage a pedidos de resolução do problema.

Figura 3 – Resolução de Problemas (Método Tradicional)

Na figura 4 o processador tem um papel pró-activo e é capaz de desenvolver acções sem receber qualquer pedido do exterior.

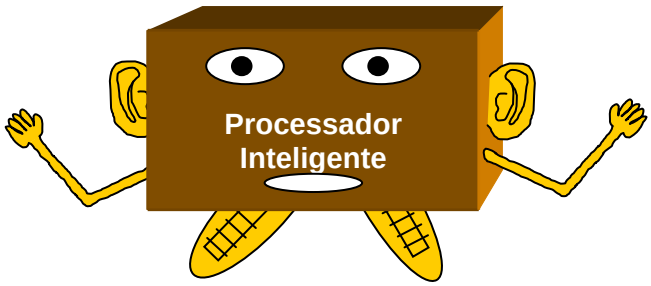


Figura 4 – Resolução de Problemas (Abordagem baseada na Inteligência Artificial)

O desenvolvimento de SI pode ser feito de duas formas diferentes, dependendo da existência ao não de soluções de mercado aceitáveis para o problema.

a) SI à medida

Partindo duma ideia concreta, avaliando as necessidades e aproveitando a oportunidade são realizadas quatro fases:

1) análise:
    i. análise de processos
    ii. análise organizacional
    iii. identificação de requisitos.
2) Especificação:
    i. especificação de requisitos funcionais;
    ii. especificação de requisitos não funcionais;
3) desenho:
    i. definição de tecnologias;
    ii. definição de arquitecturas;
4) implementaçao:
    i. programação;
    ii. integração.

No final do processo o sistema poderá em fase de funcionamento.

A implementação de um SI adaptado sofre apenas alterações nas fases:

- . desenho:
    - definição de infra-estruturas;
    - configuração;
- implementação:
    - teste;
    - integração.

O ciclo de vida de um SI é apresentado na Figura 5.

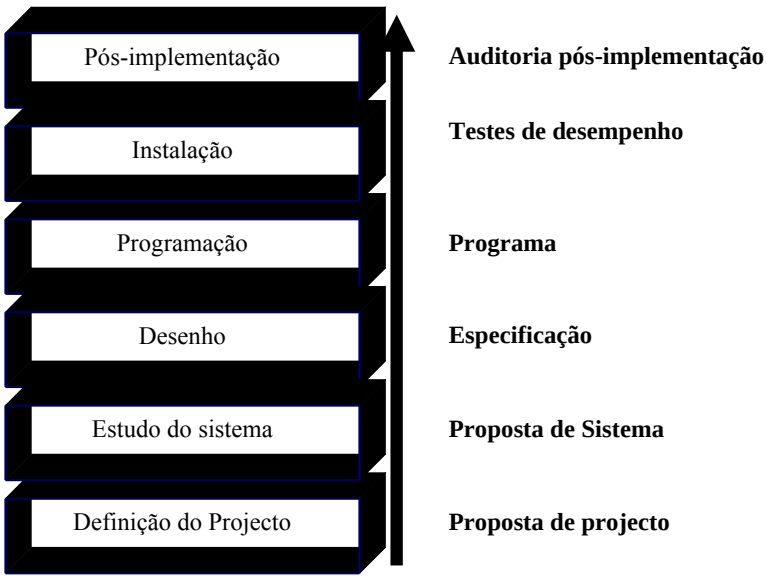


Figura 5 – Ciclo de Vida de um Sistema de Informação

Qualquer erro que possa surgir nas fases e nas operações descritas poderá levar a problemas de difícil resolução, que atrasarão o processo e poderão levar a prejuízos financeiros e logísticos graves (Figura 6).

5. Modelos básicos de comunicação

Os modelos básicos de comunicação são os seguintes:

- Troca de mensagens;
- Invocação de Procedimentos remotos;
- Avaliação remota; e
- Objectos móveis.

6. Governo Electrónico

O governo electrónico apresenta-se como uma rara oportunidade que se nos oferece para se tirar vantagem do aumento de produtividade e redução de custos que podem ser obtidos pela utilização de tecnologias Internet.
Por outro lado, pode-se afirmar que a Administração Pública tem como principal objectivo servir o cidadão, bem, rapidamente e ao menor custo possível. Sem a preciosa ajuda dos sistemas de informação, não será possível, porém, não só garantir esse objectivo, como também modernizar a própria estrutura em que esta assenta.

A sociedade da informação foi considerada uma prioridade para a Comissão Europeia durante a presidência portuguesa, em 2000. Em 2002, foi criada a Unidade de Missão Inovação e Conhecimento (UMIC), estrutura de apoio ao desenvolvimento da política governamental em matéria de inovação, sociedade da informação e governo electrónico, à qual compete actuar no âmbito das áreas da Inovação, do Governo electrónico, da Economia digital, do Apoio aos cidadãos com necessidades especiais na sociedade da informação, e no Acesso generalizado à Internet, de preferência através de banda larga. No domínio da sociedade da informação, da inovação tecnológica e do governo electrónico, foi também criada a Comissão Interministerial para a Inovação e Conhecimento, à qual compete propor estratégias, promover a articulação dos diversos programas e iniciativas, debater, aprovar e actualizar o elenco das responsabilidades dos diferentes ministérios e organismos públicos, e acompanhar a execução do "Plano de acção e-Europe 2005: Uma sociedade do conhecimento para *todos*, e de outros programas da União Europeia no âmbito da inovação, da sociedade da informação e do governo electrónico".

A Sociedade da Informação é aí definida como uma "alavanca para a mudança, capaz de reforçar a competitividade da economia portuguesa, modernizar a Administração Pública, qualificar os portugueses e aumentar a qualidade de vida".
A implementação do governo electrónico não pode seguir uma estratégia de rompimento com o passado, com as pessoas e com o funcionamento dos serviços. As soluções financeiras têm de aparecer, os sistemas devem ser integrados e integradores, as implementações devem ser suficientemente rápidas, adaptáveis e escaláveis, e os portais temáticos que são a fachada das instituições, organizações e repartições virtuais, são apenas a interface do sistema. Estes parâmetros estão longe de serem suficientes. Há que apostar nas comunicações, na gestão dos conteúdos e na integração dos meios físicos de interacção entre os humanos e os computadores, através de técnicas convencionais ou inovadoras, tais como o email, o voice mail, a partilha de ficheiros, a videoconferência, a webconferência, a Internet e as comunicações móveis, assim como a difusão e a integração da informação, do conhecimento e dos dados entre sistemas computacionais heterogéneos. É fundamental introduzir os mecanismos para garantir a confiança, a segurança, a fiabilidade no armazenamento e a integração dos dados, da imagem e da voz. É indispensável preparar os sistemas internos de gestão e

administração, para receber e enviar a informação aos gestores e aos cidadãos, assim como criar os procedimentos necessários à especificação e desenvolvimento de sistemas de apoio à decisão, capazes de proporcionar as respostas certas e atempadas (e.g., não se deve obrigar os homens do Direito a enviar documentos para os tribunais em email seguro e autenticado, para depois imprimirem-se esses documentos, sem que estes sejam processados e armazenados informaticamente. Não se deve construir um portal centralizado que permita ao cidadão pedir uma certidão para que depois o pedido chegue à conservatório via fax, a certidão sejam fotocopiada e entregue ao cidadão por correio postal muitos dias depois).

O governo electrónico é um multi-sistema de informação, onde a informação é difundido, armazenada e processada entre sistemas possivelmente heterogéneos que usam protocolos de comunicação compatíveis e sistemas de bases de dados distribuídos e heterogéneos. Os sistemas serão divididos em duas categorias:

- Sistemas internos (sistemas de informação complexos com capacidades em termos operacionais, planeamento e gestão de recursos, documentação e "back-office"), sub-divididos em duas sub-categorias:
  - Sistemas de apoio à decisão, tácticos e estratégicos;
  - Sistemas operacionais, em particular nas áreas administrativa e financeira;
- Sistemas de relação com o exterior;
  - Portais baseados em tecnologia Internet;
  - Centro de atendimento aos cidadãos;
  - Centrais de compras (e.g., através de sistemas de comércio electrónico usando mecanismos de negociação e argumentação);
  - Canais de comunicação entre governo, cidadãos e funcionários.

O ambiente a ser desenvolvido no âmbito do governo electrónico deve, assim, além da concepção e desenvolvimento dos sistemas de informação e das bases de dados, levar em linha de conta que todos os organismos públicos nele têm de ser integrados, e ser centrado em três componentes âncora: Governo e Cidadãos (G-e-C), Governo e Negócios (G-e-N), Governo e Funcionários (G-e-F).

As componentes G-e-C e G-e-N estão intimamente relacionadas com o público em geral e o mundo empresarial e de serviços. A componente G-e-F dirige-se, no essencial, aos funcionários públicos, por forma a se conseguir o seu melhor desempenho e se satisfaçam, assim, os desafios colocados pela Nova Economia.

Resumindo, poder-se-á afirmar que o governo electrónico faz-se aproveitando a dispersão geográfica, a independência das plataformas e a riqueza das soluções já implementadas, ligando-as e integrando-as por meio de comunicações rápidas e usando protocolos e normas de segurança, de difusão e de integração da informação, sem nunca esquecer que não são nem os donos das equipas de fórmula um, nem os mecânicos, nem os patrocinadores, que pilotam as máquinas, mas sim os próprios pilotos.

7. Sistemas Gestores de Bases de Dados

Definição 1: Uma base de dados é uma colecção de grande quantidade de dados.

Definição 2: Um Sistema Gestor de Bases de Dados (SGBD) é uma aplicação informática desenvolvida para armazenar e gerir bases de dados.

Os Sistemas Gestores de Bases de Dados organizam grandes quantidades de informação. Uma base de dados é uma colecção de relações, que se encontram relacionadas através de atributos comuns. A informação está contida em dispositivos chamados ficheiros. Os utilitários para gerir bases de dados disponibilizam ao utilizador uma linguagem simples, para fazer a manutenção da base de dados (através de inserções, modificações ou remoções de informação) e para responder a questões que podem ser colocadas pelo utilizador. Essa linguagem (normalmente de comandos) é uma linguagem estruturada que facilita a consulta aos dados organizados na base de dados (e.g., SQL).

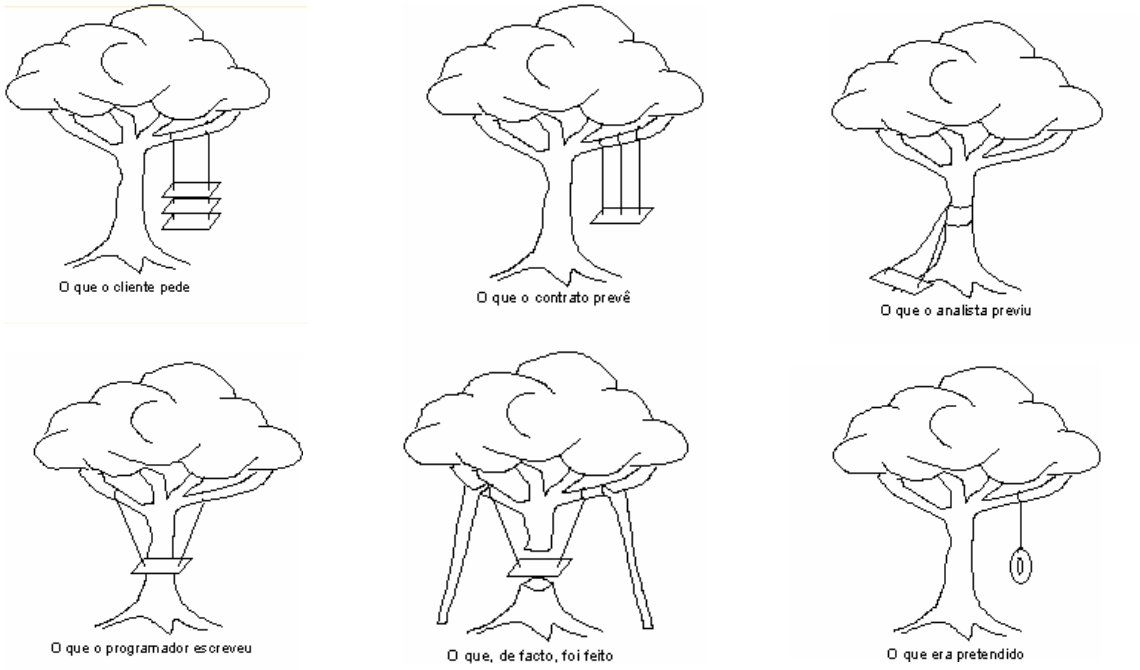


Figura 6 – Erros nas fases de desenvolvimento de SI

O modelo relacional de bases de dados é o resultado do trabalho de E.F.Codd na IBM durante a década de 70. Entre a publicação de um artigo com os fundamentos teóricos do modelo relacional e o aparecimento de um primeiro sistema baseado no modelo relacional passaram vários anos. Só em 1979/80 surgiu uma primeira implementação, através da Relational Software Inc. (actualmente Oracle Corp.). Apesar de o modelo relacional ser muito mais simples e flexível, muitos contrapunham que o seu desempenho nunca iria permitir a sua utilização comercial.

O tempo encarregou-se de provar o contrário, resolvidos os problemas iniciais. A sua implementação foi rápida, contribuindo decisivamente para a massificação da bases de dados nas organizações.
O Oracle Server da Oracle Corp., Informix SE e RDS, 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.

A estrutura relação (tabela) (Figura 7) constitui uma estrutura bidimensional, com os atributos da relação (uma ou mais colunas) e 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.

Uma base de dados é relacional se verificar as seguintes 12 regras denominadas 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.
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.

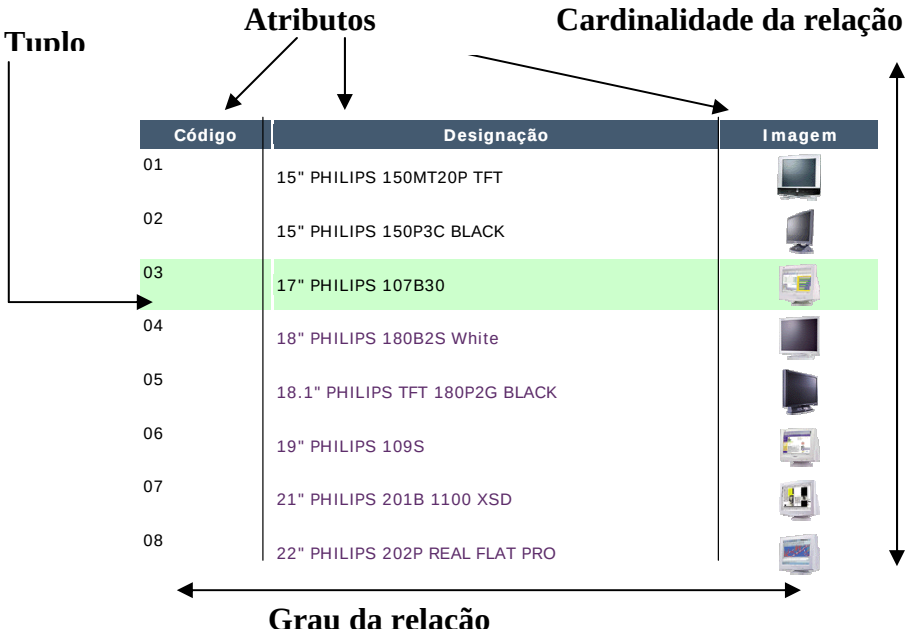


Figura 7 : Uma tabela

8. Objectos de uma base de dados relacional

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. O grau é o número de colunas de uma tabela. A cardinalidade é o número de tuplos da mesma.

As aplicações devem transferir conjuntos de dados de grande dimensão entre a memória principal dos computadores / servidores e as unidades externas de armazenamento (e..g, discos) através de técnicas sofisticadas de "buffering", paginação e endereçamento. As diferentes interrogações obedecem a codificações especiais e os dados devem estar protegidos da inconsistência às vezes resultante da intervenção de múltiplos utilizadores. Devem ser considerados routinas de recuperação em caso de falha e implementados sistemas de segurança e de controlo de acesso para os utilizadores.

A chave de uma tabela define uma forma mais fácil e mais rápida de aceder à informação. Normalmente, está implicito um sistema de codificação na definição de uma chave, podendo ser uma codificação sequencial, numérica, alfanumérica (e.g. siglas).
Podem ser definidas vários tipos de 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.

Uma **entidade** será representativa de uma classe de objectos sobre os quais se quer guardar informação (e.g., 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 (e.g., um médico tem o seu *nome, morada, especialidade* etc.).
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.

9. 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.

O processo de normalização (Figura 8) ocorre através de um conjunto de fases que conduzem a base de dados a estados onde a redundância se torna cada vez menor.
A cada um destes estados dá-se o nome de forma normal (FN).

Consideremos a ficha de consulta de uma clínica médica:

Cod.Medico:_____ Nome: _____
Especialidade : ___ --- _____
Morada Med. : _____ Contacto: _____
Idade : _____

Data	Hora	N.Benif	Paciente	Idade	Morada	Cod.Postal	Preço

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.

O primeiro passo do processo de normalização (1ªFN), 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.

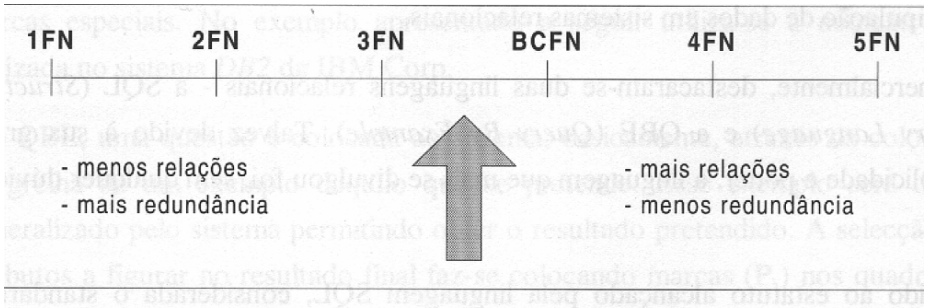


Figura 8 - Normalização

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 *1ªFN* 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*.

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→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 2ª FN, a 3ª FN e a *Boyce-Codd* FN).

Multivalor – numa relação **R(X,Y,Z)** diz-se que existe uma dependência multivalor **X–»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 4ª 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 5ª FN).

Existe uma dependência funcional X → 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 → Y então existe uma dependência funcional entre X e Y em que X determina Y ou Y depende de X. Por exemplo:

NºFuncionário → Nome_funcionário, Departamento
(NºFactura, Cod_produto) → Qtd_vendida, Preço_venda

Por definição se X → Y então a correspondência entre X e Y é do tipo 1:1 ou ∞:1

Axiomas de Armstrong :

Reflexividade (se X ⊇ Y então X → Y)
Aumentatividade (se X → Y então XZ → YZ)
Transitividade (se X → Y e Y → Z então X → Z)

Regras de Inferência :

Decomposição (se X → YZ então X → Y e X → Z)
União (se X → Y e X → Z então X → YZ)
Pseudotransitividade (se X → Y e YW → Z então XW → 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 4ª FN usa este tipo de dependência.

Uma relação na 2ªFN 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 2ªFN.

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 2ª FN.

Uma relação na 3ªFN 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 3ªFN.

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:

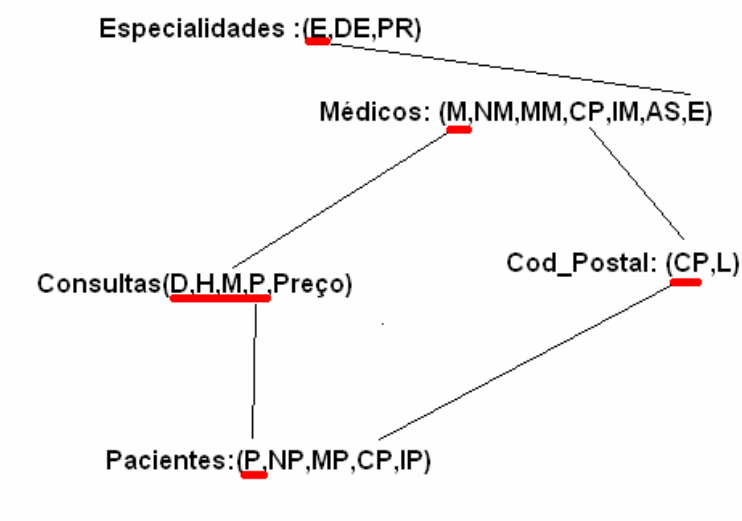
Paciente(n_ben ,paciente,idade,morada,cod_post,localidade)
n_ben → paciente,idade,morada,cod_pos
cod_post → localidade

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 3ª FN.

O exemplo está ilustrado na Figura 9.



E –Código da Especialidade
DE –Descrição da Especialidade
PR –Preço da consulta de especialidade
M –Código do Médico
NM–Nome do Médico
MM –Morada do Médico
CP –Código PostalL–Localidade
IM –Idade do Médico
AS –Anos de serviço do Médico
D -Dia da Consulta
H –Hora da Consulta
Preço –Preçopago pelo paciente
P –Código do Paciente
NP –Nome do Paciente
MP –Morada do Paciente
IP –Idade do Paciente

Figura 9

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 3ª FN é aquela em que, na maioria dos casos, termina o processo de normalização, em alguns casos a 3ªFN ainda transporta algumas anomalias que são resolvidas pela BCFN.

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:

(paciente,cod_serv) → 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 registar dois médicos do mesmo serviço para o mesmo paciente! Esta dependência deverá ser tratada ao nível aplicacional! ou então manter R juntamente com R2.

Por definição, dada uma relação R(X,Y,Z) diz-se que existe uma dependência multivalor X ->> 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.

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

Tabela 1

Na tabela 1, existem 2 dependências multivalor X ->> Y e X ->> 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.

Por exemplo: dada R(Agente,Produto,Zona) e a regra : todos os agentes vendem todos os produtos que representam em todas as zonas em que actuam; i.e.,

(Agente --> Produto e Agente --> Zona)

A solução é criar duas relações: R1(Agente,Produto) e R2(Agente,Zona)

Na tabela 2, 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 (5ª FN).

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

Tabela 2

10. Álgebra Relacional

Uma base de dados relaciona um conjunto de tabelas na forma R(A1,A2,...,Ar), com grau r. A cardinalidade é o número de elementos de R.

O conjunto dos possíveis valores de um atributo é o domínio dom(Ai) = Di (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,..,65}).

Regra 1: não pode haver 2 tuplos idênticos na mesma relação.
Regra 2: não existe uma ordem pré-definida numa relação.

Na Álgebra Relacional existem dois tipos de operações:

Operações básicas :

União: $R \cup S$
Diferença: $R - S$
Produto Cartesiano: $R \times S$
Proojecção: $\Pi_{Ai,...,Aj}(R) = \Pi_{i,...,j}(R)$
Selecção: $\sigma_{Cond}(R)$

Operações adicionais:
Intersecção: $R \cap S$
Divisão: $R \div S$
Θ-junção (e.g.,<-junção): $R \bowtie_{i\Theta j} S$
Junção natural: $R \bowtie S$
Semi-junção: $R \ltimes S$

São dados alguns exemplos de aplicação dos operadores relacionais:

União:

A	B	C
a1	b2	c1
a5	b1	c2
a2	b4	c4
a3	b3	c3

∪

A	B	C
a2	b3	c2
a1	b2	c1
a2	b4	c4

=

A	B	C
a1	b2	c1
a5	b1	c2
a2	b4	c4
a3	b3	c3
a2	b3	c2

Intersecção:

A	B	C
a1	b2	c1
a5	b1	c2
a2	b4	c4
a3	b3	c3

∩

A	B	C
a2	b3	c2
a1	b2	c1
a2	b4	c4

=

A	B	C
a1	b2	c1
a2	b4	c4

Diferença:

A	B	C
a1	b2	c1
a5	b1	c2
a2	b4	c4
a3	b3	c3

-

A	B	C
a2	b3	c2
a1	b2	c1
a2	b4	c4

=

A	B	C
a5	b1	c2
a3	B3	c4

Produto Cartesiano:

A	B
a1	b2
a5	b1
a2	b4

X

C	D
c2	d3
c1	d2

=

A	B	C	D
a1	b2	c2	d3
a1	b2	c1	d2
a5	b1	c2	d3
a5	b1	c1	d2
a2	b4	c2	d3
a2	b4	c1	d2

Projecção:

$\Pi_{A,C}$

A	B	C
a2	b3	c4
a1	b2	c1
a2	b4	c4

=

A	C
a2	c4
a1	c1

Selecção:

$\sigma_{A=a2}$

A	B	C
a2	b3	c4
a1	b2	c1
a2	b4	c4

=

A	B	C
a2	b3	c4
a2	b4	C4

Junção:

A	B	C
a1	b2	c1
a5	b1	c1
a2	b4	c4

$\bowtie$

C	D
c2	d3
c1	d2
c4	d1

=

A	B	C	D
a1	b2	c1	d2
a5	b1	c1	d2
a2	b4	c4	d1

Semi-junção:

A	B	C
a1	b2	c1
a5	b1	c1
a2	b4	c4

$\ltimes$

C	D
c2	d3
c1	d2
c4	d1

=

A	B	C
a1	b2	c1
a5	b1	c1
a2	b4	c4

C	D
c2	d3
c1	d2
c4	d1

$\ltimes$

A	B	C
a1	b2	c1
a5	b1	c1
a2	b4	c4

=

C	D
c1	d2
c4	d1

No caso da semi-junção note-se sendo:

R1 =

A	B	C
a1	b2	c1
a5	b1	c1
a2	b4	c4

e

R2 =

C	D
c2	d3
c1	d2
c4	d1

obtem-se:
R1$\ltimes$ R2 = R1 e R2$\ltimes$ R1 $\neq$ R2
porque todos os valores de C em R1 se relaciona com pelo menos um valor de C em R2, enquanto existe pelo menos um valor de C em R2 (c2) que não se relaciona com qualquer valor de C em R1.

Divisão:

A	B	C	D
a1	b2	c2	d3

÷

C	D
c2	d3

=

A	B
a1	b2

a5	b1	c1	d2
a2	b4	c1	d2
a3	b5	c4	d4
a2	b4	c2	d3
a1	b2	c1	d2

c1	d2

a2	b4

As seguintes regras de equivalência definem os operadores opcionais com base nos operadores básicos:

Sendo R e S duas tabelas do mesmo grau e com a mesma estrutura $R(A_1,...,A_r)$ e $S(A_1,...,A_r)$:

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

Sendo $R(A_1,...,A_r)$, $S(A_{r-s+1},..,A_s)$ $r > s$ e $S \neq \Phi$:

$$R \div S = \Pi_{1,...,r-s}(R) - \Pi_{1,...,r-s}((\Pi_{1,...,r-s}(R) \text{ X } S) - R)$$

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

$$R \bowtie S = \Pi_{i1,...,im}(\sigma_{R.Ai=S.Ai \text{ e } ... \text{ e } R.Ak=S.Ak}(R \text{ X } S))$$

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

A ponte entre a álgebra relacional e a programação em lógica pode ser feita usando as seguintes regras de conversão:

Sendo $R(A_1,...,A_r)$, $S(A_1,...,A_r)$ e $T(A_1,...,A_r)$

$R \cup S = T$ é definido por:

$t(A1,...,Ar) \leftarrow r(A_1,...,A_r).$
$t(A_1,...,A_r) \leftarrow s(A_1,...,A_r).$

$R - S = T$ é definido por:

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

Sendo $R(A_1,...,A_r)$, $S(A_{r+1},...,A_{r+s})$ e $T(A_1,...,A_{r+s})$

$T = R \text{ X } S$ é definido por:

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

Sendo $R(A_1,..,A_r)$, $T(A_i,...,A_j)$ com i<= r ej<= r

$\Pi_{Ai,...,Aj}(R) = T$ é definido por:

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

Sendo $R(A_1,...,A_r)$

$\sigma_{Cond}(R) = T$ é definido por:

$t(A_1,...,A_r) \leftarrow r(A_1,...,A_r) \wedge Cond$.

Sendo $R(A_1,...,A_r)$, $S(A_1,...,A_r)$ e $T(A_1,...,A_r)$

$R \cap S = T$ é definido por:

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

$R - (R - S) = T$ é definido por:

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

Aplicando a regra : $A \wedge \neg (A \wedge \neg B) = A \wedge (\neg A \vee B) = A \wedge B$

Então:

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

Sendo $R(A_1,...,A_r)$, $S(A_{r-s+1},...,A_r)$ e $T(A_1,...,A_{r-s})$:

Para calcular $R \div S = T$ define-se:

$R \div S = \Pi_{1,...,r-s}(R) - \Pi_{1,...,r-s}((\Pi_{1,...,r-s}(R) \times S) - R)$
$r_1(A_1,...,A_{r-s}) \leftarrow r(A_1,...,A_r)$.
$r_2(A_1,...,A_r) \leftarrow r_1(A_1,...,A_{r-s}) \wedge s(A_{r-s+1},...,A_r)$.
$r_3(A_1,...,A_r) \leftarrow r_2(A_1,...,A_r) \wedge \neg r(A_1,...,A_r)$.
$r_4(A_1,...,A_{r-s}) \leftarrow r_3(A_1,...,A_r)$.
$t(A_1,...,A_{r-s}) \leftarrow r_1(A_1,...,A_{r-s}) \wedge \neg r_4(A_1,...,A_{r-s})$.

sendo

$R_1 = \Pi_{1,...}(R)$
$R_2 = R_1 \times S$
$R_3 = R_2 - R$
$R_4 = \Pi_{1,...}(R_3)$

$R \bowtie_{i\Theta j} S = T$ define-se:

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

Sendo k o número de atributos comuns:

$R \bowtie S = T$ define-se:

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

11. A Linguagem SQL

O SQL é uma linguagem usada para comunicar com bases de dados. De acordo com a ANSI (American National Standards Institute) é a linguagem standard para os sistemas de gestão de bases de dados relacionais. Os comandos da SQL são usados para realizar operações tais como alterar / actualizar dados numa base de dados, ou responder a questões a partir da informação armazenada numa base de dados. Os seguintes motores de bases de dados usam SQL: Oracle, Informix, DB2, Sybase, Microsoft SQL Server, Access, Ingres, etc. A maior parte dos sistemas gestores de bases de dados usam SQL mas têm as suas próprias extensões ao SQL. No entanto os comandos mais comuns da SQL tais como "SELECT", "INSERT", "UPDATE", "DELETE", "CREATE" e "DROP" podem ser usados para fazer a maioria das operações necessárias numa base de dados.

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

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

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

=	Igual
>	Maior que
<	Menor que
>=	Maior ou igual
<=	Menor ou igual
<>	Diferente
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. Por exemplo:

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

O comando seleciona qualquer atribuído primeiro que comece por "Er". Os valores do tipo alfanumérico (strings) devem estar entre "".

Por exemplo:

```
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.

Outros exemplos da utilização do comando SELECT são:

```
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 '%s';

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

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

O comando “CREATE TABLE” é usado para criar uma tabela:

```
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
```

Por exemplo:

Exemplo:

```
CREATE TABLE empregado
(primeiro varchar(15),
```

```
ultimo varchar(20),
idade número(3),
morada varchar(30),
localidade varchar(20),
estado varchar(20),
primary key(primeiro,ultimo));
```

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 e/ou hora	
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 registos 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.
A seguinte script permite criar / recriar uma base de dados:

```
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;

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(
      fid number(9,0) primary key,
      fname varchar2(30),
      deptid number(2,0)
      );
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(
      snum number(9,0),
      cname varchar2(40),
      primary key(snum,cname),
      foreign key(snum) references student,
      foreign key(cname) references class(name)
      );
create table emp(
      eid number(9,0) primary key,
      ename varchar2(30),
      age number(3,0),
      salary number(10,2)
      );
create table dept(
      did number(2,0) primary key,
      dname varchar2(20),
      budget number(10,2),
      managerid number(9,0),
      foreign key(managerid) references emp(eid)
      );
create table works(
      eid number(9,0),
      did number(2,0),
      pct_time number(3,0),
      primary key(eid,did),
      foreign key(eid) references emp,
      foreign key(did) references dept
      );
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(
      aid number(9,0) primary key,
      aname varchar2(30),
      cruisingrange number(6,0)
      );
create table employees(
      eid number(9,0) primary key,
      ename varchar2(30),
      salary number(10,2)
```

```
      );
create table certified(
      eid number(9,0),
      aid number(9,0),
      primary key(eid,aid),
      foreign key(eid) references employees,
      foreign key(aid) references aircraft
      );
create table suppliers(
      sid number(9,0) primary key,
      sname varchar2(30),
      address varchar2(40)
      );
create table parts(
      pid number(9,0) primary key,
      pname varchar2(40),
      color varchar2(15)
      );
create table catalog(
      sid number(9,0),
      pid number(9,0),
      cost number(10,2),
      primary key(sid,pid),
      foreign key(sid) references suppliers,
      foreign key(pid) references parts
      );
create table sailors(
      sid number(9,0) primary key,
      sname varchar2(30),
      rating number(2,0),
      age number(4,1)
      );
```

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** é usado para actualizar registos:

```
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';
```

Exercício: desenvolva as produções em SQL que lhe permitam fazer as seguintes actualizações de estado da base de dados:

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

```
DROP TABLE "nometabela"
```

Exemplo:

```
DROP TABLE empregado;
```

A sintaxe completa do comando SELECT é a seguinte:

```
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;
```

O comando SELECT pode ser usados para calcular operações de agregação com grupos:

```
SELECT coluna-1ista, SUM(coluna2)
FROM "lista-de-tabelas"
GROUP BY "coluna-lista";
```

Por exemplo:

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

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

É possível estabelecer uma condição que restrinja o grupo:

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

Por exemplo:

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

SELECT dept, avg(Vencimento)
FROM empregado
```

```
GROUP BY dept
HAVING avg(Vencimento) > 20000;
```

A ordenação é implementada usando a claúsula ORDER BY.

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

Por omissão, a ordem é decrescente, por isso a palavra DESC é dispensável enquanto isto não é verdade para a ordem ascendente (ASC).

Por 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 ASC;
```

As funções de agregação são enunciadas na tabela 3.

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

Por exemplo as seguintes operações retornam um valor e não uma tabela como foi sempre visto até este momento.

```
SELECT AVG(Vencimento)
FROM empregado;

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

SELECT Count(*)
FROM empregados;
```

É possível usar-se condições lógicos via operadores lógicos:

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

Ou

```
 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');
```

Os operadores IN e BETWEEN simplificam também a escrita das condições:

```
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;
```

Exemplo:

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

É equivalente a:

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

Assim como:

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

é equivalente a:

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

No SQL, podem-se usar operadores tais como:

+	Adição
-	Subtração
*	Multiplicação
/	Divisão

%	Módulo

Assim como:

ABS(x)	retorna o valor absoluto de x
SIGN(x)	retorna o sinal de x (-1, 0, ou 1) (negativo, zero, ou positivo)
MOD(x,y)	módulo - retorna o resto da divisão inteira de x por (x%y)
FLOOR(x)	retorna o maior inteito que é menor ou igual a x
CEILING(x) or CEIL(x)	retorna o menor inteiro que é maior ou igual a x
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.

Todas as questões colocadas até agora têm 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;
```

As operações da álgebra relacional podem ser traduzidas para SQL:

União:

```
(select * from R) union (select * from S)
```

Diferença:

```
(select * from R) except (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)
      ou
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
```

Exercício:

Dado o esquema da base de dados da Figura 10, resolva:

1. Qual é o nome dos médicos com mais de 10 anos de serviço?
   $\sigma_{AS > 10}$ *medicos*

   ```
   Select NM from medicos where AS > 10
   ```

2. Indique o nome dos médicos e respectiva especialidade.
   $\Pi_{NM,DE}$ *(medicos* $\bowtie_E$ *especialidades)*

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

3. Qual é o par (nome e morada) dos pacientes residentes em Braga?
   $\Pi_{NP,MP}$ *(pacientes* $\bowtie_{CP}$ $\sigma_{L = \text{“Braga”}}$ *cod_postal)*

   ```
   Select NP,MP from pacientes,cod_postal
   where pacientes.cp = cod_postal.cp and cod_postal.L = “Braga”
   ```

4. Qual é o nome dos médicos da especialidade de oftalmologia?
   Π_{NM} *(medicos* $\bowtie_E$ $\sigma_{DE = \text{“Oftalmologia”}}$ *especialidades)*

   ```
   select NM from medicos,especialidades
   where medicos.E = especialidades.E and especialidades.DE =
   “Oftalmologia”
   ```

5. Quais são os médicos com mais de 40 anos com a especialidade de Clínica Geral?
   $\sigma_{IM > 40}$ *medicos* $\ltimes_E$ $\sigma_{DE = \text{“Clínica Geral”}}$ *especialidades*

   ```
   select me.* from medicos me,especialidades es
   where me.E = es.E and me.IM > 40 and es.DE = “Clínica Geral”
   ```

6. Quais são os médicos de Oftalmologia que consultaram pacientes de Braga?
   medicos $\ltimes_E$ $\sigma_{DE = \text{“Oftalmologia”}}$ *especialidades* $\ltimes_M$
   (consultas $\bowtie_P$ *pacientes* $\bowtie_{CP}$ $\sigma_{L = \text{“Braga”}}$ *cod_postal)*

   ```
   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”
   ```

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

   $\sigma_{IM > 50}$ *medicos* $\ltimes_M$ *(*$\sigma_{H > \text{“12:00”}}$ *consultas* $\bowtie_P$ $\sigma_{IP < 20}$ *pacientes)*

   ```
   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
   ```

8. Quais os pacientes com mais de 10 anos que <u>nunca</u> foram consultados a Oftalmologia?

$\sigma_{IP > 10}$ *pacientes* – ($\sigma_{IP > 10}$ *pacientes* $\ltimes_P$
(*consultas* $\rtimes_M$ *medicos* $\rtimes_E$ $\sigma_{DE="Oftalmologia"}$ *especialidades*))

```
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")
```

9. Quais são as especialidades consultadas no mês de Janeiro de 2002?
especialidades $\ltimes_E$ (*medicos* $\rtimes_M$ $\sigma_{month(D) = 1 and year(D) = 2002"}$ *consultas*)

```
select es.* from especialidades es,medicos me, consultas co
where year(co.D) = 2002 and month(co.D) = 1 and
es.E = me.E and co.M = me.M
```

10. Qual é o nome dos médicos com mais de 30 anos ou menos de 5 anos de serviço?
Π_{NM} ($\sigma_{IM>30 and AS < 5}$ *medicos*)

```
select me.NM from medicos me
where me.AS < 5 and me.IM >30
```

11. Quais são os médicos de Clínica Geral que não consultaram em Janeiro de 2002?
medicos $\ltimes_E$ $\sigma_{DE="Clínica Geral"}$ *especialidades* -
(*medicos* $\ltimes_M$ $\sigma_{month(D) = 1 and year(D) = 2002"}$ *consultas*)

```
select me.* from medicos me,especialidades es
where DE = "Clínica Geral" and es.E = me.E
except
select me.* from medicos me, consultas co
where year(co.D) = 2002 and month(co.D) = 1
```

12. Quais são os pacientes que já foram consultados por todos os médicos?

($\Pi_{pacientes.*,M}$ (*pacientes* $\rtimes_P$ *consultas* $\rtimes_M$ *medícos*) ÷ Π_M *medicos*

```
select pa.* from pacientes pa where not exists
      (select * from medicos me
      where me.M not in
(select co.M from consultas co where cp.P = pa.P)
```

13. Quais são as especialidades que não foram consultadas durante os meses de Janeiro e Março de 2002?
especialidades –
especialidades $\ltimes_E$ (*medicos* $\rtimes_E$ $\sigma_{momth(D) = 1 and year(D) = 2002}$ *consultas*)

```
select es.* from especialidades es
where not exists
      (select co.* from consultas co where co.M in
```

```
            (select me.M from medicos me where me.E = es.E)
      )
```

14. Quais são os médicos que nunca consultaram pacientes de Braga?
    medicos –
    medicos $\ltimes_M$ *(consultas* $\rtimes_P$ *pacientes* $\rtimes_{CP}$ $\sigma_{L="Braga"}$ *cod_postal)*

```
select me.* from medicos me
where not exists
      (select co.* from consultas co where co.M = me.M and
            co.P in (select pa.P from pacientes pa where pa.cp in
                        (select cp.cp from cod_postal cp
                        where cp.L = "Braga")
                  )
      )
```

15. Quais são os pacientes que só foram consultados a Clínica Geral?
    pacientes $\ltimes_P$ *(consultas* $\rtimes_P$ *medicos* $\rtimes_E$ $\sigma_{DE="Clínica\ Geral"}$ *especialidades)*
    - pacientes $\ltimes_P$ *(consultas* $\rtimes_P$ *medicos* $\rtimes_E$ $\sigma_{DE<>"Clínica\ Geral"}$ *especialidades)*

```
select pa.* from pacientes pa where pa.P not in
      (select co.P from consultas co where co.M in
            (select me.M from medicos me where me.E in
                  (select es.E from especialidades es
 where es.DE <> "Clínica Geral"))) and exists
 (select co2.* from consultas co2 where co2.P = pa.P)
```

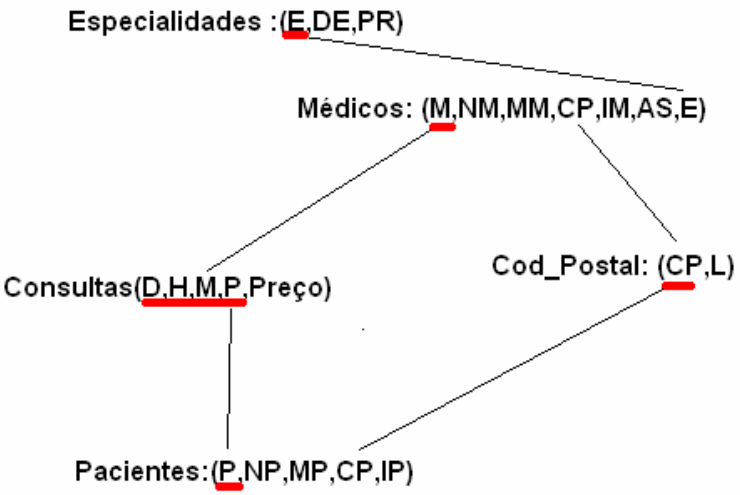


Figura 10

Exercícios

1. Qual é a média de idades dos médicos com mais de 15 anos de serviço?
2. Qual é a média de anos de serviço dos médicos para cada uma especialidade?
3. Quantas consultas estão registadas por médico?

4. Qual é a média de idades dos médicos por especialidade?
5. Para cada médico apresentar o valor facturado em 2003.
6. Qual é o número de médicos de cada especialidade?
7. Para cada especialidade com menos de dois médicos listar o valor máximo e mínimo facturado por consulta, bem como o valor médio.
8. Quais são os médicos cujo valor facturado em 2002 é superior à média?
9. Actualizar o preço de referência das especialidades em 10%.
10. Remover os pacientes residentes em Braga.
11. Incremente em 15% o preço das consultas dos médicos com mais de 40 anos de idade.
12. Actualize o preço de refencia de Clínica Geral com o valor médio cobrado nas consultas de especialidade em 2003.
13. Remover os médicos que nunca consultaram.
14. Remover os códigos postais sem ligaçoes a médicos e/ou paciente
15. Remover as consultas executadas depois das18 horas pormédicos com mais de 60 anos.

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

É dado o registo R_0 com a informação não normalizada:

R_0: médicos(m,nm,md,im,as,ce,de,pr,p,np,mp,ip,d,h,preço)

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

Assim como o tamanho de cada atributo:

Col.	Tamanho (em bytes)
m	2
nm	40
mm	50
im	2
as	2
ce	2
de	40
pr	4

Col.	Tamanho (em bytes)
p	2
np	40
mp	50
ip	2
d	8
h	5
preço	4

Obtem-se:

Tamanho inicial da relação R_0 : 253 bytes
Cardinalidade de R_0 : 50*6*1000 = 300 000
Tamanho R_0: 300 000 * 253 = 75 900 000 b = 72.38 Mb

Existe um grupo repetido <p,np,mp,ip,d,h,preço> em R_0. Deste modo, na primeira forma normal R_0 é dividido em $R_{1,1}$ e $R_{1,2}$ tal que:

$R_{1,1}$ = medico(m,nm,mm,im,as,ce,de,pr)
$R_{1,2}$ = consultas(m,p,np,mp,ip,d,h,preço)

Obtem-se:

Tamanho inicial da relação $R_{1,1}$: 142 bytes
Cardinalidade de $R_{1,1}$: 50
Tamanho $R_{1,1}$: 50 * 142 = 7 100 b = 0.01 Mb

Tamanho inicial da relação $R_{1,2}$: 113 bytes
Cardinalidade de $R_{1,1}$: 50*6*1000 = 300 000 registos
Tamanho $R_{1,1}$: 300 000 * 113 = 33 900 000 b = 32.33 Mb

Tamanho da base de dados na 1ª FN : 33 907 100 b = 32.34 Mb

Na relação $R_{1,2}$ existe a dependência p -> np,mp,ip que é uma dependência funcional de parte da chave total <m,p> . Deste nodo da 2ª FN resultam as relações:

$R_{2,1}$ = $R_{1,1}$
$R_{2,2}$ = consultas(m,p,d,h,preço)
$R_{2,3}$ = pacientes(p,np,mp,ip)

Tamanho inicial da relação $R_{2,1}$: 142 bytes
Cardinalidade de $R_{2,1}$: 50
Tamanho $R_{2,1}$: 50 * 142 = 7 100 b = 0.01 Mb

Tamanho inicial da relação $R_{2,2}$: 21 bytes
Cardinalidade de $R_{2,2}$: 50*6*1000 = 300 000
Tamanho $R_{2,2}$: 300 000 * 21 = 6 300 000 b = 6.01 Mb

Tamanho inicial da relação $R_{3,2}$: 94 bytes
Cardinalidade de $R_{3,2}$: 10 000
Tamanho $R_{3,2}$: 10 000 * 94 = 940 000 b = 0.90 Mb

Tamanho da base de dados na 2ª FN : 7 247 100 b = 6.91 Mb

Na relação $R_{2,1}$ = medicos(m,nm,mm,im,as,ce,de,pr) existem as dependências
m -> nm,mm,im,as,ce
ce->de,pr
dependências transitivas. Deste modo, após a aplicação da 3ª FN ontem-se:

$R_{3,1}$ = medicos(m,nm,mm,im,as,ce)
$R_{3,2}$ = especialidades(ce,de,pr)
$R_{3,3}$ = consultas(m,p,d,h,preço)
$R_{4,3}$ = pacientes(p,np,mp,ip)

Tamanho inicial da relação $R_{3,1}$: 98 bytes
Cardinalidade de $R_{3,1}$: 50
Tamanho $R_{3,1}$: 50 * 98 = 4 900 b = 0.005 Mb

Tamanho inicial da relação $R_{3,2}$: 46 bytes
Cardinalidade de $R_{3,2}$: 10
Tamanho $R_{3,2}$: 10 * 46 = 460 b = 0.0004 Mb

Tamanho inicial da relação $R_{3,3}$: 21 bytes
Cardinalidade de $R_{3,3}$: 50*6*1000 = 6 300 000
Tamanho $R_{3,3}$: 10 000 * 21 = 6 300 000 b = 6.01 Mb

Tamanho inicial da relação $R_{3,4}$: 94 bytes
Cardinalidade de $R_{3,4}$: 10000
Tamanho $R_{3,4}$: 10 000 * 94 = 940 000 = 0.90 Mb

Tamanho da base de dados na 1ª FN : 7 245 360 b = 6.91 Mb

Os resultados da normalização encontram-se resumidos no quadro seguinte:

Fase	Espaço (em Bytes)	Espaço (em Mbytes)	Ganho Total	Ganho entre fases
Início	**75 900 000**	**72.38**		
1ª FN	**33 907 100**	**32.34**	**55.33%**	**55.33%**
2ª FN	**7 247 100**	**6.91**	**90.45%**	**78.63%**
3ª FN	**7 245 360**	**6.91**	**90.45%**	**0.02%**

Exercício Resolvido:

Dado o esquema da Figura 11, resolva:

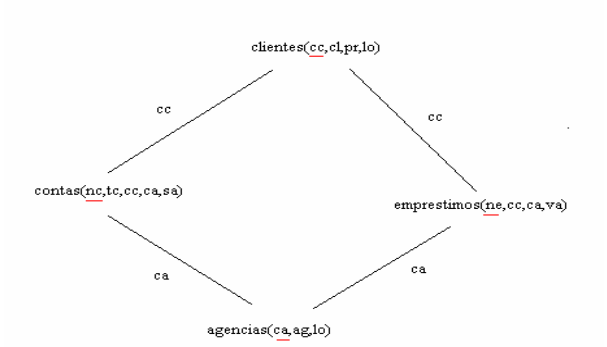


Figura 11

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

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

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?**

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

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

```
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)
```

13. 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
EXCEPT
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 ca FROM contas where 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(*) FROM contas 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 co.cc = c.cc AND co.ca = ag.ca))
```

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
```

Funções:

```
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;
```

Execuçao da função em Microsoft Visual Basic 6.0:

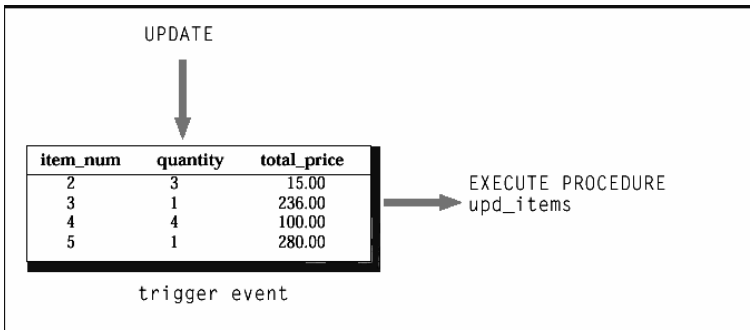
```
SQL = "Declare n1 Number(8) := " + trus(1) + "; n2 Number(8) := " +
trus(2) + "; n3 Number(9):=" + trus(3) + "; n4 Number(8):=" + trus(4)
+ "; n5 char(3):=" + trus(5) + "; n6 varchar2(10):=" + trus(6) + "; n7
varchar2(100) := " + Trim(trus(7)) + "; n8 Char(1) := " +
Trim(trus(8)) + "; n9 Integer:=" + 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)
```

Procedimentos em Informix:

```
CREATE PROCEDURE read_address (lastname CHAR(15)) – one argument
RETURNING CHAR(15), CHAR(15), CHAR(20), CHAR(15),CHAR(2), CHAR(5); --
6 items
DEFINE p_lname,p_fname, p_city CHAR(15); --define each procedure
variable
DEFINE p_add CHAR(20);
DEFINE p_state CHAR(2);
DEFINE p_zip CHAR(5);

SELECT fname, address1, city, state, zipcode
INTO p_fname, p_add, p_city, p_state, p_zip
FROM customer WHERE lname = lastname;
RETURN p_fname, lastname, p_add, p_city, p_state, p_zip; --6 items
END PROCEDURE
DOCUMENT 'This procedure takes the last name of a customer as', --
brief description
'its only argument. It returns the full name and address of the
customer.'
WITH LISTING IN '/acctng/test/listfile'
-- compile-time warnings go here
; -- end of the procedure read_address

EXECUTE PROCEDURE read_address ('Putnum');

CREATE PROCEDURE address_list ()
      DEFINE p_lname, p_fname, p_city CHAR(15);
      DEFINE p_add CHAR(20);
      DEFINE p_state CHAR(2);
      DEFINE p_zip CHAR(5);
      .
      .
```

```
      .
      LET p_fname, p_lname,p_add, p_city, p_state, p_zip =
      read_address ('Putnum');
      .
..
      -- use the returned data some way
END PROCEDURE;

CREATE PROCEDURE tryit()
.
.
.
      CREATE TABLE gargantuan (col1      INT, col2 INT, col3 INT);
      CREATE TABLE libby.tiny (col1      INT, col2 INT, col3 INT);
END PROCEDURE;

CREATE PROCEDURE scope()
DEFINE x,y,z INT;
LET x = 5; LET y = 10;
LET z = x + y; --z is 15
BEGIN
      DEFINE x, q INT; DEFINE z CHAR(5);
      LET x = 100;
      LET q = x + y; -- q = 110
      LET z = 'silly'; -- z receives a character value
END
LET y = x; -- y is now 5
LET x = z; -- z is now 15, not 'silly'
END PROCEDURE;

EXECUTE PROCEDURE read_address('Smith')
INTO p_fname, p_lname, p_add, p_city, p_state, p_zip;
CALL read_address('Smith')
RETURNING p_fname, p_lname, p_add, p_city, p_state, p_zip;

CREATE PROCEDURE call_test()
RETURNING CHAR(15), CHAR(15);
DEFINE fname, lname CHAR(15);
CALL read_name('Putnum') RETURNING fname, lname;
IF fname = 'Eileen' THEN RETURN 'Jessica', lname;
ELSE RETURN fname, lname;
END IF
END PROCEDURE;
```

Triggers:

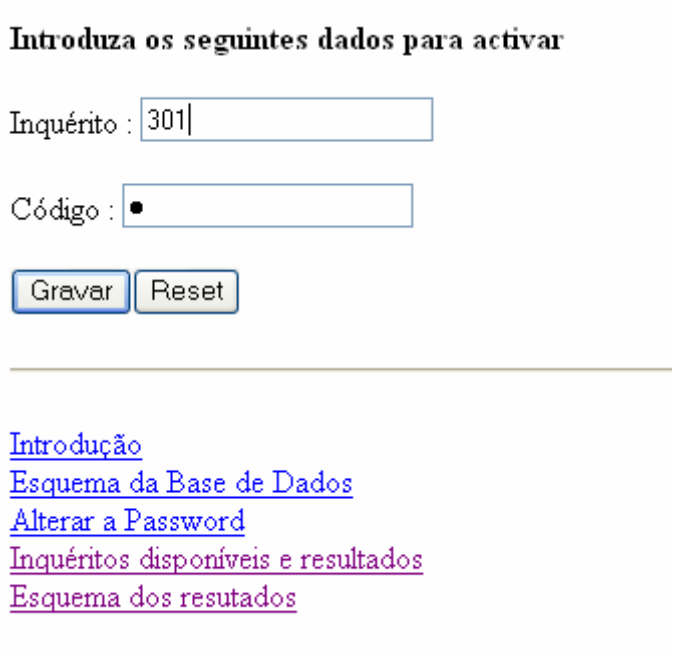



```
CREATE TRIGGER upqty
UPDATE OF quantity ON items-- an UPDATE trigger event
BEFORE(EXECUTE PROCEDURE upd_items_p1)-- a BEFORE action

CREATE TRIGGER ins_qty
INSERT ON items -- an INSERT trigger event

CREATE PROCEDURE upd_items_p1()
DEFINE GLOBAL old_qty INT DEFAULT 0;
LET old_qty = (SELECT SUM(quantity) FROM items);
END PROCEDURE;

CREATE PROCEDURE upd_items_p2()
      DEFINE GLOBAL old_qty INT DEFAULT 0;
      DEFINE new_qty INT;
      LET new_qty = (SELECT SUM(quantity) FROM items);
      IF new_qty > old_qty * 1.50 THEN
            RAISE EXCEPTION -746, 0, 'Not allowed - rule violation';
      END IF
END PROCEDURE;

CREATE PROCEDURE upd_items_p1()
DEFINE GLOBAL old_qty INT DEFAULT 0;
LET old_qty = (SELECT SUM(quantity) FROM items);
END PROCEDURE;

CREATE TRIGGER up_items
UPDATE OF quantity ON items
BEFORE(EXECUTE PROCEDURE upd_items_p1())
AFTER(EXECUTE PROCEDURE upd_items_p2());

UPDATE items SET quantity = quantity * 2 WHERE manu_code = 'KAR‘

CREATE TABLE log_record
(item_num SMALLINT,
ord_num INTEGER,
username CHARACTER(8),
update_time DATETIME YEAR TO MINUTE,
old_qty SMALLINT,
new_qty SMALLINT);

FOR EACH ROW(INSERT INTO log_record
VALUES    (pre_upd.item_num,    pre_upd.order_num,    USER,    CURRENT,
pre_upd.quantity, post_upd.quantity));
```

```
CREATE TRIGGER up_price
UPDATE OF unit_price ON stock
REFERENCING OLD AS pre NEW AS post
FOR EACH ROW WHEN(post.unit_price > pre.unit_price * 2)
(INSERT INTO warn_tab VALUES(pre.stock_num, pre.order_num,
pre.unit_price, post.unit_price, CURRENT

CREATE TRIGGER upd_totpr
UPDATE OF quantity ON items
REFERENCING OLD AS pre_upd NEW AS post_upd
FOR    EACH    ROW(EXECUTE    PROCEDURE    calc_totpr(pre_upd.quantity,
post_upd.quantity, pre_upd.total_price) INTO total_price)
```

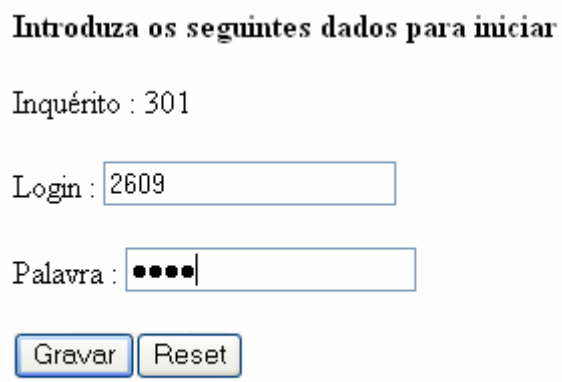
12. PHP e Bases de dados

O seguinte programa em PHP permite construir a seguinte página:



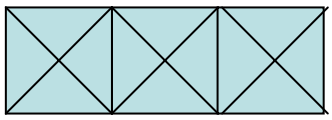

```
<?php
require("local.inc");
?>
<html>

<head>
<title>Construção de Inquérito </title>
</head>

<body>

<?php
echo "<form method='POST' action='inq.php'>";

echo "</ol>
  <p><b> Introduza os seguintes dados para activar</b></p>
  <p>Inquérito : <input type='text' name='q' size ='20'></p>
  <p>Código    : <input type='password' name='palavra' size ='20'></p>
```

```
  <p><input type='submit' value='Gravar' name='b1'><input type='reset'
value='Reset' name='b2'></p>";
?>
</form>
<hr><br>
<a href=/apsi/infos/trb.html>Introdução</a><br>
<a href=inquerito.zip>Esquema da Base de Dados</a><br>
<a href=mudapass.php>Alterar a Password</a><br>
<a href=lista.php>Inquéritos disponíveis e resultados</a><br>
<a href=resultados.zip>Esquema dos resutados</a>
</body>

</html>
```




```
<?php
require("local.inc");
?>
<html>
<head>
<title></title>
</head>
<body>
<?php
$palavra = $_POST["palavra"];
$q = $_POST["q"];
if ($palavra == '9') {
      echo "<form method='POST' action='inquerito.php'>";
      echo "</ol>
       <p><b> Introduza os seguintes dados para iniciar</b></p>
      <p>Inquérito : ".$q."</p>
      <p>Login          :  <input  type='textbox'  name='usern'  size
='20'></p>
      <p>Palavra       :  <input  type='password'  name='passw'  size
='20'></p>
       <input type='hidden' name='palavra' value=".md5($palavra).">
       <input type='hidden' name='q' value=".$q.">
        <p><input   type='submit'   value='Gravar'   name='b1'><input
      type='reset' value='Reset' name='b2'></p>
      </form>";
}

else {
      echo "Código Errado - Contacte o Docente";
}
?>
```

```
</body>
</html>
```



Inquérito : 301

José Manuel Ferreira Machado

1.Qual é o seu sexo?

1. Feminino ☐
2. Masculino ☐

Qual é o seu Estado Civil?

1. Casado ☐
2. Solteiro ☐
3. Viúvo ☐
4. Divorcidado ☐

Quantos filhos tem?

1. Nenhum ☐
2. Um ☐
3. Dois ☐
4. Três ☐
5. Mais de três ☐

```
<?php
require("local.inc");
?>
<html>
<head>
<title></title>
</head>
<body>
<?php
$usern = $_POST["usern"];
$passw = $_POST["passw"];
$palavra = $_POST["palavra"];
$q = $_POST["q"];
if ($palavra == md5("9")) {
        $db = odbc_connect(BASE, DBUSER, DBPASSWORD)
        $sql ="select * from alunos where t1 = '".$usern."'";
        $rs = odbc_exec($db,$sql);
        if (odbc_fetch_row($rs)) {
                $PWD1 = odbc_result($rs,"t3“);
                $nome = odbc_result($rs,"t2“);
        }
else {
```

```
                $PWD1 = "";
}
if (trim($passw) == trim($PWD1)) {
        $qx = $q - 300;
        echo "<form method='POST' action='gravainq2.php'>";
        $sql = "select * from alunos_grupos where grupo = ".$qx;
        odbc_exec($db,$sql);
        while (odbc_fetch_row($rs)) {
                $alu = odbc_result($rs,"aluno“);
        echo "<img
src='http://labia.di.uminho.pt/apsi/photos/".$alu.".jpg' width=60>";
        }
        echo "<h2>Inquérito : ".$q."</h2><br>";
        echo "<h2> ".$nome."</h2>";
        echo "<img
src='http://labia.di.uminho.pt/apsi/photos/".$usern.".jpg'
width=100>";
        $sql = "select * from questoes where quest = '".$q."' order by
linha";
        $rs = new recordset($db,$sql);
        echo "<ol>";
        while (odbc_fetch_row($rs)) {
                $texto = odbc_result($rs,"texto“);
                $caixa = odbc_result($rs,“caixa“);
                $nomecaixa = trim(odbc_result($rs,“caixa“));
                if (substr($nomecaixa,0,1) == "c") {
                        $j = substr($nomecaixa,1,3);
                        $c[$j] = $nomecaixa;
                }
                if ($caixa == 0) {
                        echo "</ol>";
                        echo "<p><b>".$texto."</b></p>";
                        echo "<ol>";
                }
elseif ($caixa == 1) {
echo "<li>".$texto." <input type='checkbox' name='c[".$j."]'
value='ON'></li>"; }
                elseif ($caixa == 3) {
                        echo  "<li>".$texto; }
                elseif ($caixa == 4) {
echo  $texto." <input type='checkbox' name='c[".$j."]'
value='ON'></li>"; }
                else {
                        echo "</ol><p><b>".$texto."</b></p>
                        <p><input type='textbox' name='".$nomecaixa."'
size ='120'></p><ol>";
                }
}
echo "</ol>
<p><b> Grave o seu inquérito</b></p>
<p><input type='hidden' name='palavra' value = '".$palavra."' size
='10'></p>
<p><input type='hidden' name='usern' value = '".$usern."' size
='10'></p>
<p><input type='hidden' name='passwd' value = '".$passw."' size
='10'></p>
<p><input type='hidden' name='q' value = '".$q."' size ='10'></p>
<p><input type='submit' value='Gravar' name='b1'><input type='reset'
value='Reset' name='b2'></p>";
echo "</form>";
}
```

```
else {  echo "Password Errada"; }

$db->disconnect();
}
else { echo "Parâmetros Ilegais - Contacte o Docente"; }
?>
</body>
</html>
```

Inquéritos

301 Preenchidos : 39 Quem Resultados
302 Preenchidos : 39 Quem Resultados
303 Preenchidos : 39 Quem Resultados
304 Preenchidos : 38 Quem Resultados
305 Preenchidos : 37 Quem Resultados
306 Preenchidos : 38 Quem Resultados
307 Preenchidos : 36 Quem Resultados
308 Preenchidos : 36 Quem Resultados
309 Preenchidos : 36 Quem Resultados
310 Preenchidos : 35 Quem Resultados
311 Preenchidos : 35 Quem Resultados
313 Preenchidos : 35 Quem Resultados
314 Preenchidos : 34 Quem Resultados
315 Preenchidos : 7 Quem Resultados
316 Preenchidos : 34 Quem Resultados

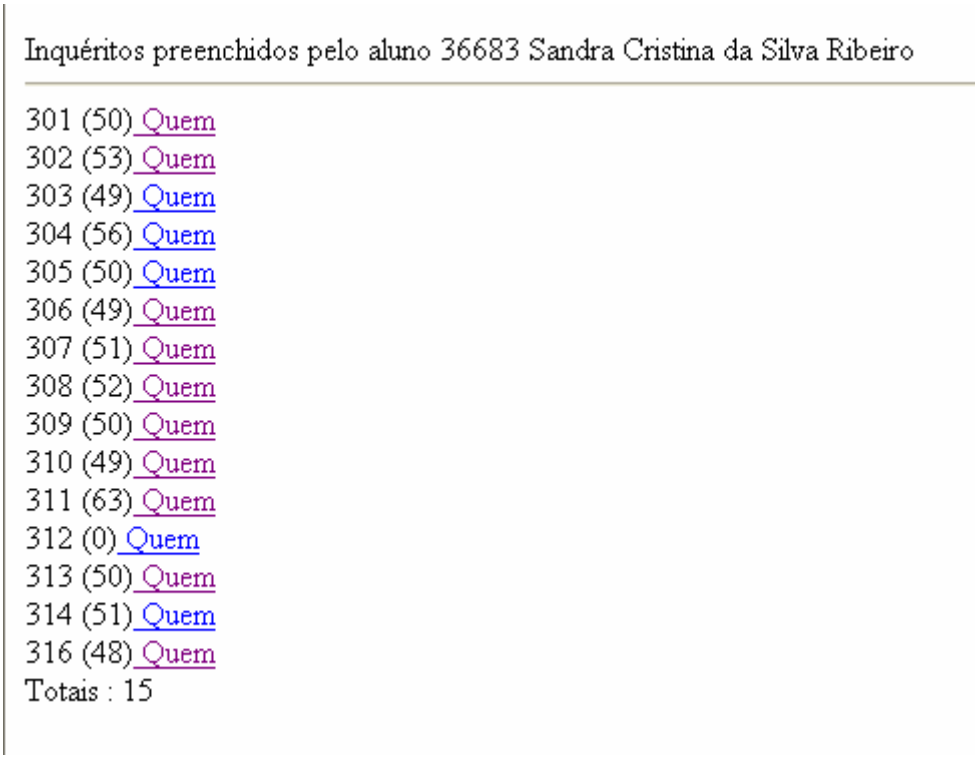
```
<?php
require("local.inc");
?>
<html>
<head>
<title>Lista de Inquéritos </title>
</head>
<body>
<?php
$db= odbc_connect(BASE, DBUSER, DBPASSWORD));
$sql ="select distinct quest as q from questoes where quest > 100";
echo "Inquéritos<br><hr>";
$rs = odbc_exec($db,$sql);
while (odbc_fetch_row($rs)) {
        $q = odbc_exec($sql,"q“);
        $sql ="select count(*) as q1 from respostas where inquerito =
".$q;
        $rs2= odbc_exec($db,$sql);
        if odbc_fetch_row($rs) ) {
                $q1 = odbc_result($rs,”q1”);
        }
echo $q." Preenchidos : ".(integer) $q1."<a href=quem.php?q=$q>
Quem</a><a href=respostas.php?q=$q> Resultados </a><br>";
}
odbc_close($db);
?>
</body>
```

Alunos que preencheram o Inquérito 301

30456 Luis Miguel Aires Nogueira Cerqueira (51) Quais
36683 Sandra Cristina da Silva Ribeiro (50) Quais
39940 Nilza Helena Neves Fonseca (53) Quais
42224 João Ricardo Costa Esteves da Silva (50) Quais
42399 Diana Filipa Anjos Batista (50) Quais
42401 Nuno Filipe Miranda (54) Quais
42402 Angela Margarida Carvalho Lima (49) Quais
42404 Juliana José Novais Pinheiro (50) Quais
42405 Liliana Patricia Pereira de Sousa (51) Quais
42406 Carine Morgado Teixeira (49) Quais
42407 Paulo Renato Barbosa da Rocha (50) Quais
42408 Catarina Isabel Magalhães Ribeiro (53) Quais
42411 Fátima Isabel da Silva Lopes (52) Quais
42412 João Manuel da Costa Duarte (50) Quais
42413 Fernando Barros Machado (49) Quais
42418 Mónica Alves Cabeleira (50) Quais
42420 Daniela Filipa Pinto de Sousa (51) Quais
42421 Ana Raquel Lopes Quintas (51) Quais
42422 Brandon Alexandre Domingues Gaspar (50) Quais
42424 Patricia Liliane Carvalhal Cavaco (49) Quais
42428 Tânia Catarina Sá de Brito Esteves (48) Quais
42429 Sonia Nogueira da Silva (50) Quais
42431 Nuno Ricardo Silva Sousa (50) Quais
42432 Aurora Sendão Fernandes (55) Quais
42433 Maria Susana Faria Pereira (50) Quais
42434 Gilbert de Castro Rodrigues (49) Quais
42435 Jorge Selénio Portugal Ribeiro Marques (48) Quais
42437 Nuno Miguel Gomes Agra da Silva (50) Quais
44133 José Maria Esteves de Faria Couto (50) Quais
44143 Cátia Patricia Ribeiro da Silva (54) Quais
44156 Pedro Miguel Xavier Ferreira Antunes da Cunha (50) Quais
44215 Ana Sofia Faria Cunha dos Santos (50) Quais
44238 Joana de Fátima Peixoto Martins (53) Quais
44303 Marta Filipa Araújo da Silva (49) Quais
44469 Daniela Gonçalves Arada (56) Quais
44471 José Manuel Parente Rodrigues (50) Quais

```
<?php
require("local.inc");
?>
<html>
<head>
<title>Lista de Inquéritos </title>
</head>
<body>
<?php
$q = $_GET["q"];
$db = odbc_connect(BASE, DBUSER, DBPASSWORD);
$sql ="select aluno,t2,res1 from respostas,alunos where
respostas.aluno = alunos.t1 and inquerito= ".$q." order by aluno";
echo "Alunos que preencheram o Inquérito $q<br><hr>";
$rs = odbc_exec($db,$sql);
$k = 0;
while (odbc_fetch_row($rs)) {
        $q1 = $rs->row["aluno"];
        $q2 = $rs->row["res1"];
        $t2 = $rs->row["t2"];
        $c1 = 0;
        for ($i=0; $i<256;$i++) {
                if (substr($q2,$i,1) == "1") {
                        $c1 = $c1 + 1;
}
        }
        echo $q1." ".$t2." "."($c1)<a href=quais.php?a=$q1> Quais
</a><br>";
```

```
        $k = $k + 1;
}
echo "Totais : $k";
odbc_close($db);
?>
</body>
</html>
```




```
<?php
require("local.inc");
require("/www/".DIR."/psw.inc");
require("/www/".DIR."/rs.inc");
?>
<html>
<head>
<title>Lista de Inquéritos </title>
</head>
<body>
<?php
$a = $_GET["a"];
$db=odbc_connect(BASE, DBUSER, DBPASSWORD) {
$sql ="select t2 from alunos where t1 = '".$a."'";
$rs = odbc_exec($db,$sql);
if (odbc_fetch_row($rs)) {
        $t2 = odbc_result($rs,”t2");
}
$sql ="select inquerito,res1 from respostas where aluno= ".$a." order
by inquerito";
echo "Inquéritos preenchidos pelo aluno $a $t2<hr>";
$rs = odbc_exec($db,$sql);
$k = 0;
while (odbc_fetch_row($rs)) {
        $q1 = odbc_exec($rs,"inquerito“);
        $q2 = odbc_fetch_row($rs,"res1“);
        $c1 = 0;
```

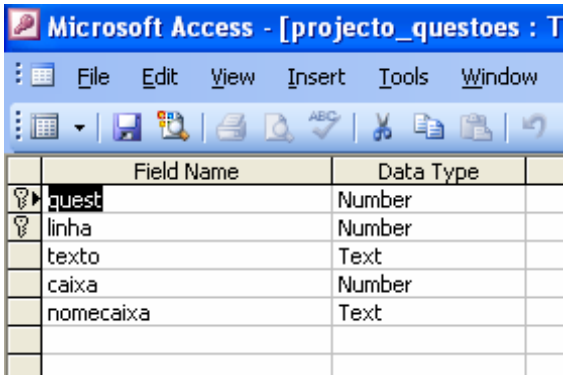
```
        for ($i=0; $i<256;$i++) {
                if (substr($q2,$i,1) == "1") {
                        $c1 = $c1 + 1;
                }
}
echo $q1." ($c1)<a href=quem.php?q=$q1> Quem </a><br>";
$k = $k + 1;
}
echo "Totais : $k";
odbc_close($db);
?>
</body>
</html>
```

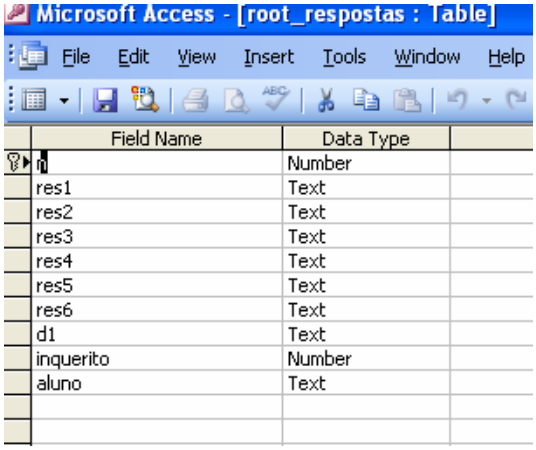
Gravação do inquérito

```
<?php
require("local.inc");
?>
<html>
<head>
<title>Gravação de Inquérito </title>
<font face="verdana">
<?php
$c = $_POST["c"];
$passwd = $_POST["passwd"];
$palavra = $_POST["palavra"];
$usern = $_POST["usern"];
$q = $_POST["q"];
$db = odbc_connect(BASE, DBUSER, DBPASSWORD);
$sql ="select * from alunos where t1 = '".$usern."'";
$rs = odbc_exec($db,$sql);
if (odbc_fech_row($rs) {
        $PWD1 = odbc_rsult($rs,"t3“);
}
else {
        $PWD1 = "";
}
if ((trim($passwd) == trim($PWD1)) and ( $palavra == md5('9'))) {
        $sql ="delete from respostas where inquerito = ".$q." and aluno
= '".$usern."'";
        $rs = new recordset($db);
        odbc_do($db,$sql);
        for ($k = 1; $k <= 6; $k++) {
                $resulta[$k] = "";
                for ($i = ($k-1)*250+1; $i <= 250*$k; $i++) {
                        if ($c[$i] == "ON") {
                        $resulta[$k] = $resulta[$k]."1";
                        }
                        else {
                        $resulta[$k] = $resulta[$k]."0";
                        }
                }
        }
        $date = date("d-m-Y H:i");
        $at1 = "res1,res2,res3,res4,res5,res6,d1,inquerito,aluno";
$resulta = "'". $resulta[1] . "','". $resulta[2] . "','" . $resulta[3]
. "','" . $resulta[4] ."','" .$resulta[5].   "','". $resulta[6]."'";
$vt1 = $resulta.",'".$date."',".$q.",'".$usern."'";
$sql ="insert into respostas(".$at1.") values (".$vt1.")";
odbc_do($db,$sql);
```

```
echo "Inquérito gravado com sucesso - Obrigado";

}
else
{
echo "Password errada";
}
odbc_close($db);
 ?>
</font>
</body>
</html>
```

Estrutura das tabelas





13. Ligação em .NET

Establecer ligação com o Oracle usando ODP.NET:

C#

```
stringconstr= "UserId=Scott;Password=tiger;DataSource=orcl9i";
OracleConnectioncon= newOracleConnection(constr);
con.Open();
Console.WriteLine("Connectedto database!");
```

Visual Basic.NET'

```
Dim constrAs String= "UserId=Scott;Password=tiger;DataSource=orcl9i"
Dim con As OracleConnection= NewOracleConnection(constr)
con.Open()
Console.WriteLine("Connectedto database!")
```

Criar e executar comandos SQL sem retorno:

C#

```
Stringblock= "INSERT INTO testblob(id, name) VALUES (100, 'José')";
OracleCommandcmd= newOracleCommand();
cmd.CommandText= block;
cmd.Connection= con;
cmd.ExecuteNonQuery();
```

Visual Basic.NET

```
Dim sql as string
Dim my Command as OracleClient.Oracle
Commandsql= "INSERT INTO testblob(id, name) VALUES (100, ‘José’)”
 myCommand= NewOracleClient.OracleCommand(sql, oracledb)
myCommand.ExecuteNonQuery()
```

Criar e executar comandos SQL com retorno (SELECT):

Visual Basic.NET

```
Dim sql as string
dim myCommand as OracleClient.OracleCommand
dim myReader As OracleClient.OracleDataReader
sql=  “selectid, nome fromtabela "
myCommand= NewOracleClient.OracleCommand(sql, oracledb)
myReader= myCommand.ExecuteReader()
If myReader.Read() Then
      If myReader.IsDBNull(0) Then
                A = myReader. GetValue(0)
      Endif
      If myReader.IsDBNull(1) Then
                B = myReader. GetString(1)
      End if
End if
```

EXEMPLO DE UM SELECT (Visual Basic.NET)

```
oracledb = New OracleClient.OracleConnection
oracledb.ConnectionString = "User ID=sil;Data Source=MARTE;password=x"
oracledb.Open()

sql = "select id_termos,de_termos from termos_xml order by id_termos"

myCommand = New OracleClient.OracleCommand(sql, oracledb)
myReader = myCommand.ExecuteReader
Dim codigo, descricao As String
While myReader.Read
        codigo = myReader.GetString(0)
        If myReader.IsDBNull(1) Then
                descricao = ""
        Else
                descricao = myReader.GetString(1)
        End If
        ...........
End While
myReader.Close()
oracledb.Close()
```

EXEMPLO DE UM UPDATE (Visual Basic.NET)

```
oracledb = New OracleClient.OracleConnection
oracledb.ConnectionString = "User ID=sil;Data Source=MARTE;password=x"
oracledb.Open()

sql = "update tabigif set digif='" & desc.Text & "' where cigif='" &
TC & "'"

myCommand = New OracleClient.OracleCommand(sql, oracledb)
myCommand.ExecuteNonQuery()
```

```
oracledb.Close()
```

EXEMPLO COM UTILIZAÇÃO DE LOB TEMPORARIO (Visual Basic.NET)

```
oracledb = New OracleClient.OracleConnection
oracledb.ConnectionString = "User ID=pce;Data Source=MARTE;password=x"
oracledb.Open()
Dim AE As New System.Text.UnicodeEncoding

Dim ByteArray As Byte() = AE.GetBytes(pxml.DocumentElement.OuterXml)

Dim pdata As New OracleClient.OracleParameter
Dim tx As OracleClient.OracleTransaction
Dim sql As String

tx = oracledb.BeginTransaction
myCommand = New OracleClient.OracleCommand
myCommand = oracledb.CreateCommand
myCommand.Transaction = tx
sql = "declare xx Clob; begin dbms_lob.createtemporary(xx, false, 0);
:tempClob := xx; end;"

pdata.Direction = ParameterDirection.Output
pdata.OracleType = OracleClient.OracleType.Clob
pdata.ParameterName = "tempClob"

myCommand.Parameters.Add(pdata)
myCommand.CommandText = sql
myCommand.ExecuteNonQuery()

Dim tempLob As OracleClient.OracleLob
tempLob = myCommand.Parameters(0).Value

tempLob.BeginBatch(OracleClient.OracleLobOpenMode.ReadWrite)

tempLob.Write(ByteArray, 0, ByteArray.Length)

tempLob.EndBatch()

myCommand.Parameters.Clear()
myCommand.CommandText = "pce.updateXMLtoTermos"

myCommand.CommandType = CommandType.StoredProcedure

Dim oraparameter As New OracleClient.OracleParameter("strXML",
OracleClient.OracleType.Clob)
oraparameter.Direction = ParameterDirection.Input
oraparameter.SourceVersion = DataRowVersion.Original
oraparameter.SourceColumn = "MSGDATA"
oraparameter.Value = tempLob
myCommand.Parameters.Add(oraparameter)
myCommand.Parameters.Add("codigo_", OracleClient.OracleType.VarChar,
20, ParameterDirection.Input).Value = TC
myCommand.ExecuteNonQuery()
tx.Commit()
oracledb.Close()
```

14. 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 DELETEME.SQL
select 'drop table ', table_name, 'cascade constraints \;' from user_tables;
SPOOL OFF
@DELETEME
```

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.
```

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;
```

Exemplo de Sessão

```
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>
```

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;
```

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;
```

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>
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.
```

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;
```

"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`

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.

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

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

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 contruções 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 galinha (cID INT PRIMARY KEY,
                      eID INT REFERENCES ovo(eID));

CREATE TABLE ovo(eID INT PRIMARY KEY,
                 cID INT REFERENCES galinha(cID));
```

Estas definições provocam um erro no Oracle, porque na criação da tabela `galinha` é feita referência à tabela `ovo` que ainda não existe. Alterar a ordem não resolve o problema.

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

```
CREATE TABLE galinha(cID INT PRIMARY KEY,
                     eID INT);
CREATE TABLE ovo(eID INT PRIMARY KEY,
                 cID INT);
```

E depois adicionar as restrições (chaves estrangeiras):

```
ALTER TABLE galinha ADD CONSTRAINT galinhaREFovo
    FOREIGN KEY (eID) REFERENCES ovo(eID)
    INITIALLY DEFERRED DEFERRABLE;
ALTER TABLE ovo ADD CONSTRAINT ovoREFgalinha
    FOREIGN KEY (cID) REFERENCES galinha(cID)
    INITIALLY DEFERRED DEFERRABLE;
```

`INITIALLY DEFERRED DEFERRABLE` permite adiar a verificação de restrições até ao comando `COMMIT`. Por exemplo:

```
INSERT INTO galinha VALUES(1, 2);
INSERT INTO ovo VALUES(2, 1);
COMMIT;
```

Para eliminar as tabelas deve-se eliminar primeiro as "constraints".

```
ALTER TABLE ovo DROP CONSTRAINT ovoREFgalinha;
ALTER TABLE galinha DROP CONSTRAINT galinhaREFovo;
DROP TABLE ovo;
DROP TABLE galinha;
```

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>
```

De notar:

- 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 claúsula `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 is *row-level*; senão é do tipo *statement-level*.

- 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 claúsula `WHEN`, tal não se verifica.
  - A claúsula `REFERENCING` é usada para atribuir sinónimos à variáveis `NEW` e `OLD`.
  - Uma restrição pode ser especificada na claúsula `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.

Exemplos

```
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 esrros 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>';
```

Elimação 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;
```

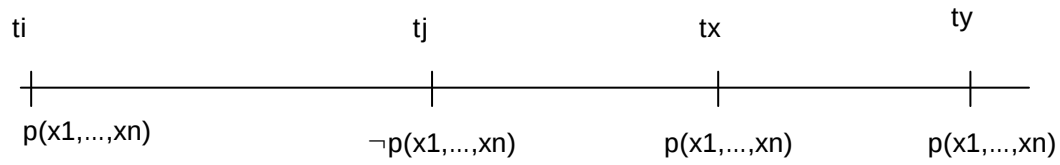
Ao executar:

```
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'
```

15. Bases de dados dependentes da dimensão temporal



- **Abordagem baseada em informação negativa**

  Tabela p+

T	A1	…	An
ti	x1	…	xn
tx	x1	…	xn

  Tabela p-

T	A1	…	An
tj	x1	…	xn
ty	x1	…	xn

  A1,…,An ∈ p no ponto tk sse
  $\exists t1 \mid t1,A1,\ldots,An \in p+ \wedge t1 < tk$ e
  $\neg \exists t2 \mid t2,A1,\ldots,An \in p- \wedge t2 < tk \wedge t2 > t1$

  A1,…,An ∉ p no ponto tk sse
  $\exists t1 \mid t1,A1,\ldots,An \in p- \wedge t1 < tk$ e
  $\neg \exists t2 \mid t2,A1,\ldots,An \in p+ \wedge t2 < tk \wedge t2 > t1$
  ou $\neg \exists t2 \mid t2,A1,\ldots,An \in p+ \wedge t2 < tk$

- **Abordagem baseada em intervalos**

  Tabela p*

From	To	A1	…	An
ti	tj	x1	…	xn
tx	ty	x1	…	xn

  A1,…,An ∈ p no ponto tk sse
  $\exists t1,t2 \mid t1,t2,A1,\ldots,An \in p* \wedge t1 <= tk <= t2$
  A1,…,An ∉ p no ponto tk sse
  $\neg \exists t1,t2 \mid t1,t2,A1,\ldots,An \in p* \wedge t1 <= tk <= t2$

16. Bases de dados distribuídas

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.

No sistema centralizado (Figura 12), os dados e o processamento estão centralizados. No sistema cliente-servidor (Figura 13) os dados estão centralizados enquanto o processamento é distribuído pelo diferentes clientes. O servidor é responsável apenas pela execução de interrogações em linguagem SQL.

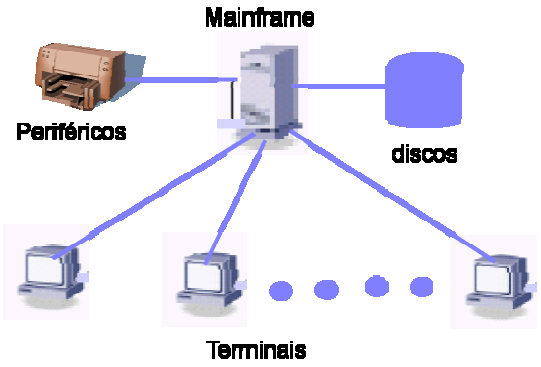


Figura 12: Sistema centralizado

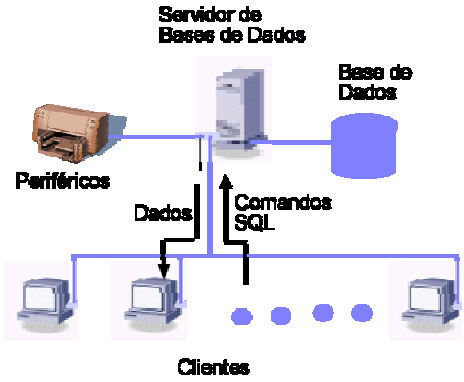
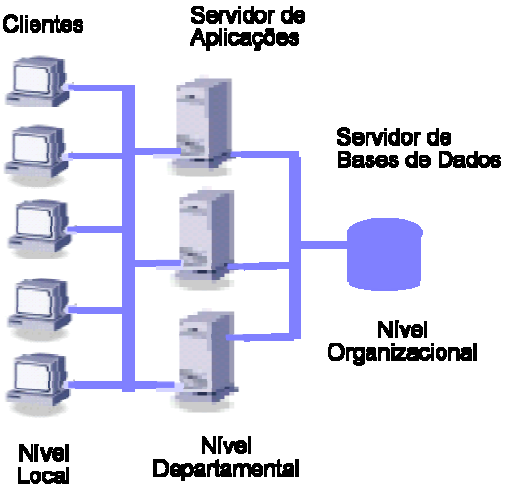


Figura 13 : Sistema cliente-servidor



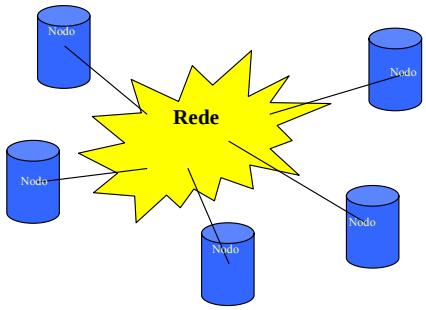


Figura 15 : Base de dados distribuída

Figura 14: Arquitectura 3 camadas

A arquitectura 3-camadas suporta o processamento pela camada dos clientes e pela camada aplicacional (Figura 14). Os comandos em SQL são também executados na camada organizacional. Não existe qualquer tipo de ligação física entre os clientes e a base de dados, funcionando o nível aplicacional como uma camada de segurança e de verificação de permissões. O nível aplicacional pode ser implementado usando tecnologia Web (via Intranet) fazendo com que não seja necessário instalar qualquer software de apoio na camada de clientes onde um simples "browser" é suficiente para executar as operações.

Na figura 15 é apresentado o esquema duma base de dados distribuída onde o processamento e os dados são distribuídos. Em cada nodo há uma base de dados capaz de sincronizar-se, fragmentar-se e distribuir interrogações de forma optimizada e transparente por todos os nodos.

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.

17. Sistemas Gestores de Bases de dados distribuídas

Por definição, 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, de uma forma mais abrangente, 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.

Existem dois tipos de SGBDs distribuídos:

> **Homogéneos.** Neste caso, todos os nodos usam o mesmo SGBD, fazendo lembrar um único sistema de bases de dados mas em que, ao contrário 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 relaciona!), até SGBDs baseados em modelos de dados diferentes (relacionais, rede, QO, etc.).

Os SGBDs distribuídos podem também ser caracterizados de acordo com o tipo de infra-estruturas de comunicação que utilizam:

> **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".

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.

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 colidi com a necessidade de manter todas as réplicas sincronizadas, desde falhas nas comunicações, até falhas dos próprios nodos individuais.

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 replica ção 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.

A fragmentação de dados pode assumir três formas:

- **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 e dividida em varias partes, cada uma delas com o mesmo esquema da tabela original. Continuando a analogia com as operações da álgebra relacional, o exemplo da Figura pode ser obtido executando três operações de selecção sobre a tabela lógica e armazenando as tabelas resultantes (tabelas físicas) em nodos diferentes.
- **Fragmentação mista**. Sobre uma mesma tabela lógica são definidos fragmentos verticais e horizontais..

Idealmente, um sistema de bases de dados distribuídas deveria apresentar-se ao nível aplicacional como um só sistema correndo sobre uma "grande máquina". Por outras palavras, o nível aplicacional deveria aceder a dados localizados algures na rede com a mesma facilidade com que acederia se esses dados 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.

O desenvolvimento de bases de dados distribuídas tem as seguintes características:

- **Dificuldades de desenvolvimento**. Já que a localização dos dados e o código necessário para a navegação na rede teriam que ser especificados nos acessos aos dados.
- **Dificuldades de manutenção**. Se, por qualquer razão, a localização dos dados fosse alterada para outro(s) nodo(s) na rede, seria necessário actualizar todas as aplicações por forma a referenciar as novas localizações.
- **Disponibilidade permanente**. Alterações na arquitectura da rede, por adição de novos nodos ou remoção de nodos existentes, assim como falhas de nodos, não devem impedir os restantes nodos de continuarem a sua operação, se não total, pelo menos parcialmente.
- **Fragmentação transparente**. Uma mesma tabela pode estar fragmentada por vários nodos. Essa fragmentação deve ser transparente aos utilizadores que deverão continuar a "ver" uma só tabela lógica.
- **Duplicação transparente.** Alguns dados, por razões de prevenção e/ou desempenho, podem estar duplicados em vários nodos. Da mesma forma, essa duplicação deve ser transparente aos utilizadores. O SGBD é o único responsável por manter a sua coerência.
- **Processamento de questões distribuído**. O processamento de uma questão pode necessitar da cooperação de vários nodos. Todo esse processo é coordenado pelo sistema sem intervenção dos utilizadores.
- **Independência do hardware, sistema operativo e tecnologias de rede**. Não deverá ser obrigatório que o hardware, o sistema operativo ou a tecnologia de rede utilizadas sejam iguais em todos os nodos. O mesmo deverá acontecer com os SGBDs utilizados. Ainda que SGBDs diferentes cooperem seria conveniente que o utilizador pudesse aceder aos dados presentes noutras máquinas utilizando o mesmo interface que para os dados presentes localmente. O utilizador deveria estar abstraído destes detalhes.

Relativamente à concepção de bases de dados distribuídas existem, basicamente, duas 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 mais que provável heterogeneidade das várias bases de dados em presença, esta será a abordagem que mais dificuldades oferece.

O processamento distribuído de uma questão consiste na decomposição dessa questão em sub questões, de acordo com a distribuição dos dados pretendidos pelos vários nodos. Posteriormente, nos nodos respectivos, a resolução de cada uma destas sub questões é optimizada localmente, de forma idêntica aos sistemas centralizados.
Resumindo, a execução de uma questão global consiste, normalmente, num conjunto de execuções de sub questões locais, provavelmente em paralelo, cujos resultados

intermédios são movimentados entre os nodos participantes e finalmente reunidos para constituir a resposta final.

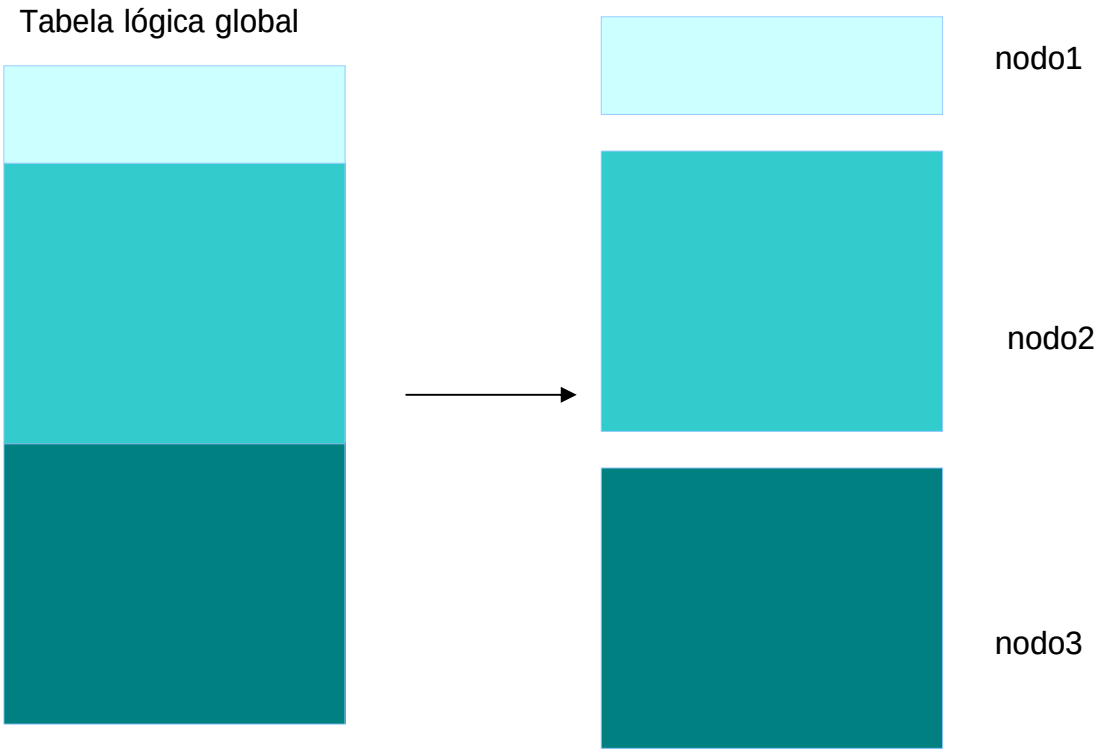


Figura 16 – Fragmentação horizontal

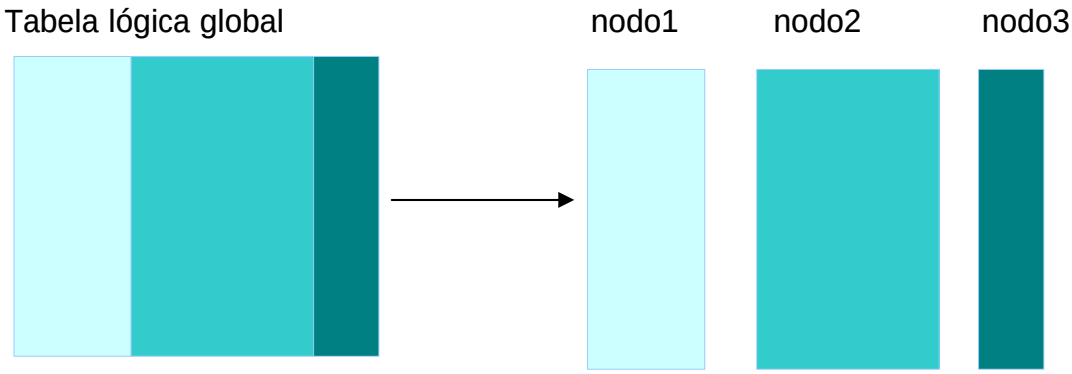


Figura 17 – Fragmentação Vertical

18. 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.

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...

Exemplo:

```
<encomenda numero="12345">
        <cliente nclie="123">
                <nomecli> Empresa lda</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 lda	Braga
…	…	…

artiencom

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

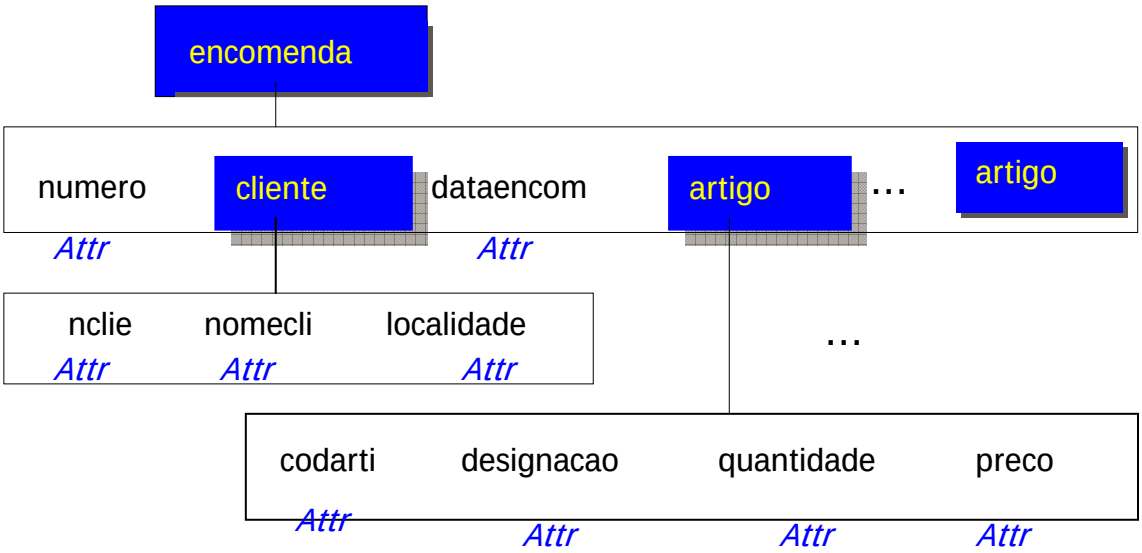
…

<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 lda</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>
```



19. XML e Oracle

O Oracle disponibiliza o tipo de dados sys.XMLTYPE para graver 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);
```

No primeiro caso além do atributo do tipo XMLTYPE existem outros atributos, no segundo a tabela apenas tem um atributo.
Para inserir um texto em XML deve-se utilizar a função sys.XMLTYPE.createXML().

Por exemplo:

```
insert into PURCHASEORDER (PODOCUMENT) values (
sys.XMLTYPE.createXML('
<PurchaseOrder>
        <Reference>BLAKE-20010625140342 98PDT</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>
 '));
```

Ou

```
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>'));
```

A interrogação é feita usando o comando SELECT. Para obter todo o texto em XML, na utiliza-se, por exemplo[2]:

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

O resultado, considerando a informação inserida anteriormente é:

```
P.PODOCUMENT.GETCLOBVAL()
---------------------------------
<PurchaseOrder>
  <Reference>BLAKE-20010625140342 98PDT</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>
```

Outro exemplo é:

select t.documento.getClobVal() from tabxml t;

A função EXTRACT permite instanciar parte do texto XML.

```
select P.PODOCUMENT.extract('/PurchaseOrder/ShippingInstructions').getClobVal()
from PURCHASEORDER P
where
P.PODOCUMENT.extract('/PurchaseOrder/Reference/text()').getStringVal() =
```

[2] No SQLPlus, tendo em conta o tamanho da string rsultante use o comando SET LONG 10000.

```
'BLAKE-20010625140342980PDT')
```

O resultado é:

```
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>
```

Outro exemplo é:

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

20. Datawarehousing

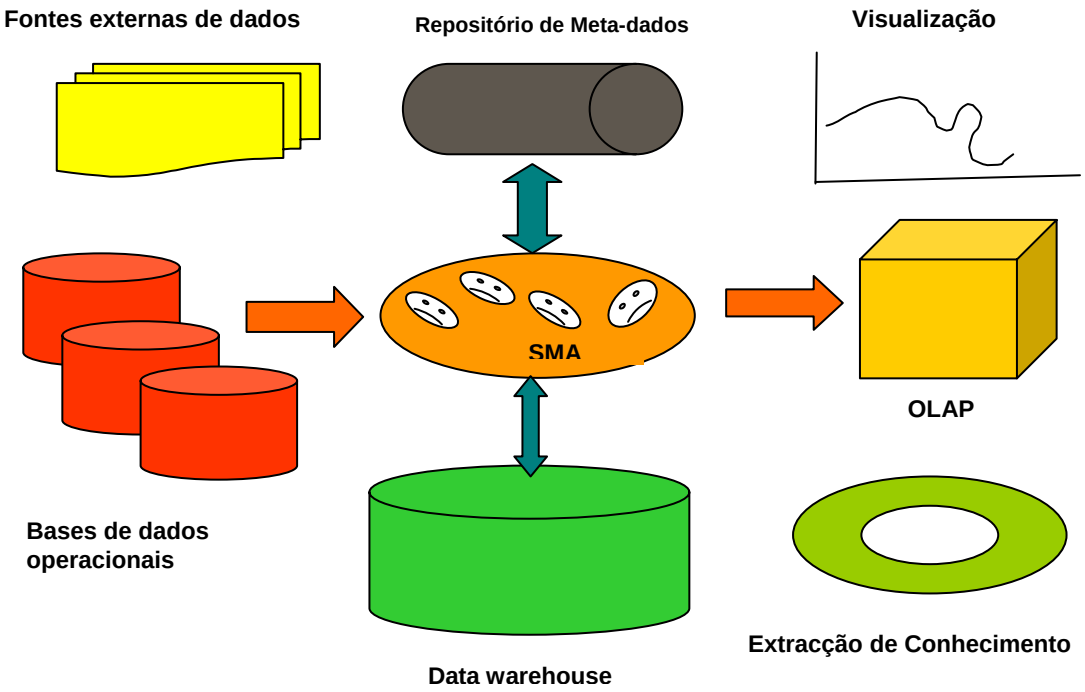


Figura 16

A figura 16 ilustra o modelo clássico de um datawarehouse com particular atenção para:

- As fontes externas de dados que devem ser integrados no sistema.
- As bases de dados operacionais onde os dados são arquivados;
- O repositório de meta-dados, fundamental para a compreensaõ dos dados a arquivar;
- As ferramentas de visualização, por exemplo visualização do estado do sistema em tempo real, a partir da informação armazenada ou processada no datawarehouse;
- O Sistema Multiagente (SMA);

- As ferrramentas OLAP (Online Analytic Processing).

Os Sistemas Multiagente (SMA) definem um novo conceito de computação e de 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.

As propriedades dos SMA são:
- 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.
Os 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).

21. Administração de bases de dados Oracle

Tipos de utilizadores

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 mantem 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

A avaliação do hardware do servidor de bases de dados consiste em:

- 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 computdores 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 eficiciência 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 papeis ("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.

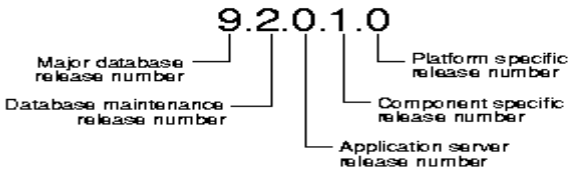
Tarefa 8: Copiar as funcionalidades da base de dados

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.

Identificação da versão de software Oracle



Major Database Release Number
É o principal identificador. Representa uma versão de software que contem um número significativo de novas funcionalidades.
Database Maintenance Release Number
Este dígito representa um nível adicional para versões de correcção, incluíndo algumas novas funcionalidades.
Application Server Release Number
Este dígito acrescenta uma nível adicional para o Oracle Application Server (e.g. Oracle9*i*AS).
Component Specific Release Number
Este dígito identifica um nível de versão específico para um componente. Diferentes componentes podem apresentar um dígito diferente nesta posição. **Platform Specific Release Number**
Este dígito identifica a plataforma.

Verificação do número da versão do software Oracle

```
COL PRODUCT FORMAT A35
COL VERSION FORMAT A15
COL STATUS FORMAT A15
SELECT * FROM PRODUCT_COMPONENT_VERSION;

PRODUCT                                  VERSION      STATUS
---------------------------------------- -----------  --------------
NLSRTL                                   9.2.0.1.0    Production
Oracle9i Enterprise Edition              9.2.0.1.0    Production
 PL/SQL                                  9.2.0.1.0    Production
TNS for Solaris:                         9.2.0.1.0    Production
```

Administrador da base de dados – Segurança e provilégios

A conta do sistema operativo para o DBA

O administrador deve ter acesso a comandos do sistema operativo para executar operações fndamentais 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-directorias 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 utilisadores estão bloqueados (locked) e desactivados (expired), excepto SYS, SYSTEM, SCOTT, DBSNMP, OUTLN, AURORAJISUTILITY$, AURORA$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 ciado 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 tabela no

esquema SYS, podendo apenas alterar-se os parametros 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 ciada com o papel de DBA.
- O utilizador SYSTEM serve para criar tabelas e views adicionais para mostrar informação de caracter administrativo e tabelas ou views internas usadas em ferramentas ou opções do Oracle. Não se deve usar esta valência para o interesse individual de uitilizadores.

O papel de DBA (DBA role)

- Um papel pré-definido, DBA, é automaticamente criado para cada base de dados. Este papel contem 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 incluí os provilégios SYSDBA e SYSOPER. Estes privilégios são especiais e além de permtir 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 • Incluí o privlégio 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 • Incluí o privlégio RESTRICTED SESSION

Ligação com privilégio de Administração

Assuma que o utilizador *scott* executa:

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

Mais tarde executa o commando 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

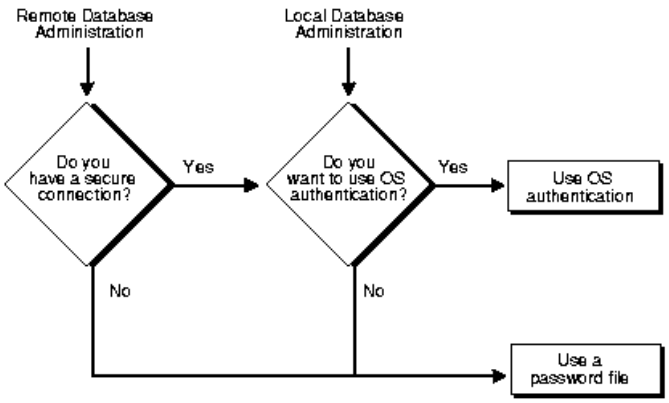
A tabela foi criada no esquema *scott* e não no esquema *SYS*. O comando correcto seria:

*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).

A ligação usando autenticação OS é feita, 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 dadis que corram em máquinas com o mesmo sistema operativo ou não.

Exemplos (de Oracle 10g e 9i – António Rodrigues – FCA)

Utilização de uma máscara e da função TO_DATE :

```
INSERT INTO empregado (n_emp, nome, categoria, reporta, entrada,
salario)
  VALUES (1234, 'MARCO', 'CHEFE DE DIVISÂO', 7566,
  TO_DATE ('24-Julho-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)
      );
```

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'))
         TABLESPACE users,
      PARTITION vendas_fev   VALUES LESS THAN
         (TO_DATE ('1-Mar-2002','DD-MON-YYYY'))
         TABLESPACE users,
      PARTITION vendas_mar    VALUES LESS THAN
         (TO_DATE ('1-Abr-2002','DD-MON-YYYY'))
         TABLESPACE users,
      PARTITION vendas_q2 VALUES LESS THAN
         (TO_DATE ('1-Jul-2002','DD-MON-YYYY'))
         TABLESPACE users,
      PARTITION vendas_q3 VALUES LESS THAN
         (TO_DATE ('1-Out-2002','DD-MON-YYYY'))
         TABLESPACE users,
      PARTITION vendas_q4 VALUES LESS THAN
         (TO_DATE ('1-Jan-2003','DD-MON-YYYY'))
```

```
          TABLESPACE users);
```

Criação de sequências:

```
CREATE SEQUENCE cod_venda_seq
      INCREMENT BY 1
      START WITH 1000 MAXVALUE 999999999 MINVALUE 1
       NOCYCLE CACHE 20 NOORDER;
```

Utilização de sequências:

```
INSERT INTO venda2
      (cod_venda, vendedor, produto, distrito, data_da_venda,
       valor )
  VALUES
      (cod_venda.nextval, 'João Dias', 'Bomba B', 'Viseu',
      TO_DATE('5-Abr-2002','DD-MON-YYYY'), 1500);
```

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;
```

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
```

Utilização de *triggers*:

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

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

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;
```

Consultar a informação sumária sobre a SGA:

```
SELECT * FROM v$sga;

NAME                    VALUE
--------------------    ----------
Fixed Size                  453492
Variable Size             109051904
Database Buffers           25165824
Redo Buffers                 667648
```

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 * dba$tablespaces;
```

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

```
SELECT * FROM dba_free_space;
```

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:

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

Atribuir um *temporary tablespace* a um utilizador já existente:

```
ALTER USER fernando TEMPORARY TABLESPACE TEMP2;
```

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 ;
```

Alterar um *tablespace* para *read-Only*:

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

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;
```

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;
```

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';
```

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;
```

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;
```

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
```

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
```

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.
```

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;
```

Mover *datafiles* entre *tablespaces*:

1. Antes de mais, fazer *backup* da base de dados, ver próximos capítulos.
2. Partido do princípio que se pretende mover um dos datafiles do *tablespace* DATA1, ver a lista dos *datafile*s 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
```

3. Pôr o *tablespace* DATA1 *offline.*

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

4. Copiar o *datafile* pretendido para o novo local, usando os comandos do sistema operativo.
5. 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';
```

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

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

7. Fazer novo *backup* da base de dados.

Remover *datafiles*:

1. Criar um novo *tablespace*.

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

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

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

```
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
```

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”;
```

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
```

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

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

```
SQL > ALTER TABLESPACE DADOS5 READ ONLY;
```

2. 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.
```

3. 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.

4. 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.
```

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

```
SQL> ALTER TABLESPACE DADOS5 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
```

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
```

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;
```

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;
```

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_TIME           END_TIME             UNDOTSN   UNDOBLKS  TXNCOUNT 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.
```

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';
```

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.
```

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

```
SET ECHO off
REM NAME:    TFSPRVVW.SQL
REM USAGE:"@path/tfsprvvw"
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 DBA_USER_PRIVS
REM    which may be queried to obtain information about which
REM    privileges are available to a given user, or which users
REM    enjoy a particular privilege.
REM -----------------------------------------------------------
REM EXAMPLE:
REM    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
REM Main text of script follows:
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
```

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;
```

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";
```

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;
```

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

-- 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);
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 simlar
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;

   --
   -- 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
  -- 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;
     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>>

    --
    -- 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;
/

-- 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>
```

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*:

1. 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
2. Verificar o parâmetro de inicialização da instância:

   REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE

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
```

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;
```

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;
```

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;
```

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;
```

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
```

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
```

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

```
SQL> AUDIT SESSION WHENEVER SUCCESSFUL;

SQL> AUDIT SESSION WHENEVER NOT SUCCESSFUL;
```

Registar todas as operações que afectem tabelas:

```
AUDIT TABLE;
```

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

```
SQL> SELECT * FROM AUDIT_ACTIONS;

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;
```

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
---------  --------  ----------  -----------       -----------  SYSTEM
02.09.28     CLIENTE       DELETE                 0
SYSTEM     02.09.28     CLIENTE       DELETE                0
SCOTT      02.09.28     CLIENTE       DELETE             2004
```

Registar 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;
```

22. Bibliografia

Ullman J., *Principles of Database and Knowledge-Base Systems*, Volumes I and II, Computer Science Press, 1988.

Pereira J.,*Tecnologia de Bases de Dados* , FCA - Editora de Informática, 1997.

Laudon, Kenneth e Laudon, Jane Price, "*Sistemas de Informação com Internet*"; LTC – Livros Técnicos e Científicos, S.A.; ISBN 85-216-1182-X; 4ª edição; Rio de Janeiro, 1999

Bell D., and Grimson J.*, "Distributed Database Systems",* Addison-Wesley Publishing Company, New York, USA, 1992.

Ramakrishman R. and Gehrke G., *Database Management System*, 3rd Edition, Mc Graw Hill High Edication, 2003.

Date C., *An Introduction to Database Systems*, Volume I, VI Edição, Addison-Wesley Systems Programming Series, 1996.

Date, C.J., Darwen, H., *A Guide to the SQL Standard* , IV Edição, Addison-Wesley Inc, 1997.

Brathwaite K., *Object-Oriented Database Design, Concepts and Application*, Academic Press, 1993.

Damas L., *SQL*, FCA – Editora de Informática, 1997.