Universidade de Coimbra - DEI/FCTUC
Copyright © Pedro Bizarro | bizarro@dei.uc.pt | 2001 | Versão 1.6 Beta

O Fabuloso Copiança de SQL Oracle!

Tudo o que não queres saber mas a que és obrigado! ☺

Intro aos clientes SQL

SQL Worksheet

Shortcut	Start→Programs→Oracle Enterprise Manager→SQL Worksheet
Executável	C:\orant\BIN\VAW.EXE
Executar comandos	F5 ou ícone Execute.
Aceder à history	CTRL+H
Executar um ficheiro	File→Open
Gravar resultados	File→Save
Observações	Comando único pode ou não ter o ';' final. Vários comandos separados por ';' ou por ';' e '/'.

SQL Plus

Shortcut	Start→Programs→Oracle for Windows NT→SQL Plus 8.0
Executável	C:\orant\BIN\PLUS80W.EXE
Executar comandos	ENTER
Aceder à history	Bem... é muito mais complicado! Veja o manual! ☺
Executar um ficheiro	start 'nome_ficheiro_com_path';
Gravar resultados	spool 'nome_ficheiro_com_path';
Observações	Comandos acabam com ';'.Vários comandos separados por ';' e '/'.

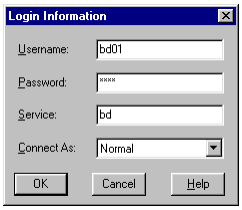
Parâmetros configuráveis no SQL Worksheet e SQL Plus

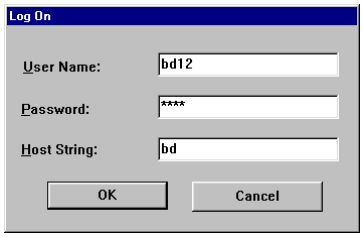
Ver os parâmetros: `SHOW ALL;`

Configurar tamanho coluna de números, caracteres, longs e datas

`SET NUMWIDTH n; / SET CHARWIDTH n; / SET LONGWIDTH n; / SET DATEWIDTH n;`

Login & pwd





Login / pwd / service

bd01/bd01/bd (ou bd.dei.uc.pt)

a

bd12/bd12/bd

Código das tabelas

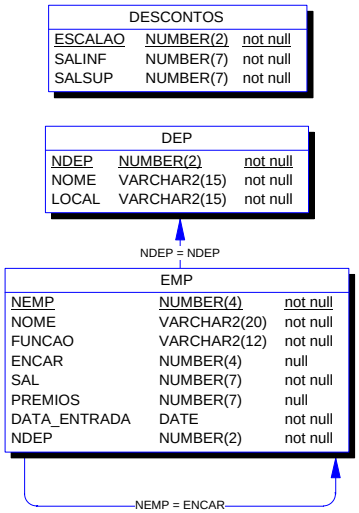
criaTabelas.sql

```
/* Cria a tabela dos descontos */
CREATE TABLE descontos
(escalao      NUMBER(2)      CONSTRAINT pk_esc_descontos PRIMARY KEY,
 salinf       NUMBER(7)      CONSTRAINT nn_inf_descontos NOT NULL   ,
 salsup       NUMBER(7)      CONSTRAINT nn_sup_descontos NOT NULL   ,
 CONSTRAINT ck_salinf_salsup CHECK (salinf < salsup));

/* Cria a tabela dos departamentos */
CREATE TABLE dep
(ndep         NUMBER(2)      CONSTRAINT pk_ndep_dep      PRIMARY KEY,
 nome         VARCHAR2(15)   CONSTRAINT nn_nome_dep      NOT NULL   ,
 local        VARCHAR2(15)   CONSTRAINT nn_local_dep     NOT NULL);

/* Cria a tabela dos empregados */
CREATE TABLE emp
(nemp          NUMBER(4)      CONSTRAINT pk_nemp_emp     PRIMARY KEY,
 nome          VARCHAR2(20)   CONSTRAINT nn_nome_emp     NOT NULL   ,
 funcao        VARCHAR2(12)   CONSTRAINT nn_funcao_emp   NOT NULL   ,
 encar         NUMBER         CONSTRAINT fk_encar_emp
                                 REFERENCES emp(nemp)    NULL       ,
 data_entrada  DATE           DEFAULT SYSDATE
                              CONSTRAINT nn_data_emp     NOT NULL   ,
 sal           NUMBER(7)      CONSTRAINT nn_sal_emp      NOT NULL   ,
 premios       NUMBER(7)      DEFAULT NULL                          ,
 ndep          NUMBER(2)      CONSTRAINT nn_ndep_emp     NOT NULL
                              CONSTRAINT fk_ndep_emp
                                 REFERENCES dep(ndep));
```

Modelo Físico



Home Install

1. Para instalar a BD em casa fazer os seguintes passos:
2. Instalar o servidor Oracle
3. Ler as instruções e executar os seguintes scripts:
   http://www.dei.uc.pt/~bizarro/aulas/bd1/2000/criaBD01.sql
   http://www.dei.uc.pt/~bizarro/aulas/bd1/2000/criaTabelas.sql
   http://www.dei.uc.pt/~bizarro/aulas/bd1/2000/insereDados.sql

Insere dados nas tabelas

insereDados.sql

```
/* Insere os descontos */
INSERT INTO descontos VALUES (1,   55000,    99999);
INSERT INTO descontos VALUES (2, 100000,   210000);
INSERT INTO descontos VALUES (3, 210001,   350000);
INSERT INTO descontos VALUES (4, 350001,   550000);
INSERT INTO descontos VALUES (5, 550001, 9999999);

/* Insere os departamentos */
INSERT INTO dep VALUES (10, 'Contabilidade', 'Condeixa');
INSERT INTO dep VALUES (20, 'Investigação',  'Mealhada');
INSERT INTO dep VALUES (30, 'Vendas',        'Coimbra');
INSERT INTO dep VALUES (40, 'Planeamento',   'Montemor');

/* Insere os empregrados */
ALTER SESSION SET NLS_DATE_FORMAT = 'yyyy.mm.dd';
INSERT INTO emp VALUES(1839, 'Jorge Sampaio',  'Presidente'  ,NULL, '1984.02.11', 890000,  NULL, 10);
INSERT INTO emp VALUES(1566, 'Augusto Reis',   'Encarregado' ,1839, '1985.02.13', 450975,  NULL, 20);
INSERT INTO emp VALUES(1698, 'Duarte Guedes',  'Encarregado' ,1839, '1991.11.25', 380850,  NULL, 30);
INSERT INTO emp VALUES(1782, 'Silvia Teles',   'Encarregado' ,1839, '1986.11.03', 279450,  NULL, 10);
INSERT INTO emp VALUES(1788, 'Maria Dias',     'Analista'    ,1566, '1982.11.07', 565000,  NULL, 20);
INSERT INTO emp VALUES(1902, 'Catarina Silva', 'Analista'    ,1566, '1993.04.13', 435000,  NULL, 20);
INSERT INTO emp VALUES(1499, 'Joana Mendes',   'Vendedor'    ,1698, '1984.10.04', 145600, 56300, 30);
INSERT INTO emp VALUES(1521, 'Nelson Neves',   'Vendedor'    ,1698, '1983.02.27', 212250, 98500, 30);
INSERT INTO emp VALUES(1654, 'Ana Rodrigues',  'Vendedor'    ,1698, '1990.12.17', 221250, 81400, 30);
INSERT INTO emp VALUES(1844, 'Manuel Madeira', 'Vendedor'    ,1698, '1985.04.21', 157800,     0, 30);
INSERT INTO emp VALUES(1900, 'Tome Ribeiro',   'Continuo'    ,1698, '1994.03.05',  56950,  NULL, 30);
INSERT INTO emp VALUES(1876, 'Rita Pereira',   'Continuo'    ,1788, '1996.02.07',  65100,  NULL, 20);
INSERT INTO emp VALUES(1934, 'Olga Costa',     'Continuo'    ,1782, '1986.06.22',  68300,  NULL, 10);
INSERT INTO emp VALUES(1369, 'Antonio Silva',  'Continuo'    ,1902, '1996.12.22',  70800,  NULL, 20);
```

Comparadores

=	Igualdade.	>	Maior que.
!=	Diferença.	>=	Maior ou igual a.
<>	Diferença.	<	Menor que.
		<=	Menor ou igual a.

`[NOT] IN` [Não] Pertence ao conjunto. Equivalente a '= ANY'.

`ANY` Precedido por =, >, >=, < ou <= e seguido por uma lista de valores. Devolve TRUE se o comparador anterior devolver TRUE para pelo menos um dos valores da lista.

`SOME` Equivalente a ANY.

`ALL` Semelhante a ANY, mas devolve TRUE apenas se o comparador anterior devolver TRUE para TODOS os valores da lista.

`[NOT] BETWEEN ... AND ...` Testa se o valor está contido no intervalo.

`EXISTS` Devolve TRUE se a subquery (correlacionada) retornar pelo menos uma linha.

`LIKE [NOT] string [ESCAPE 'c']` Testa se a variável (do tipo caracter) está contida no padrão definido por string. Na string, o caracter especial '%' corresponde a um qualquer conjunto de 0, 1 ou mais caracteres e o caracter especial '_' corresponde a 1 caracter. O caracter de ESCAPE 'c' (definido pelo utilizador) quando colocado antes de '%' ou de '_' torna-os caracteres normais.

`IS [NOT] NULL` Testa se o valor é [não é] nulo.

Nulos

Funções e Operações

- Todas as funções de linha devolvem NULL excepto as funções NVL e TRANSLATE. Exemplo: `NVL(null,10) = 10`.
- Todas as funções de grupo **ignoram** os nulos excepto o `COUNT(*)`.
- Todas as operações numéricas com nulos produzem nulos.

Condições

Condição	Resultado
10 IS NULL	FALSE
10 IS NOT NULL	TRUE
NULL IS NULL	TRUE
NULL IS NOT NULL	FALSE
10 = NULL	UNKNOWN
10 != NULL	UNKNOWN
NULL = NULL	UNKNOWN
NULL != NULL	UNKNOWN

Nota: Um WHERE com uma condição UNKNOWN devolve zero linhas. Nesse aspecto é igual a FALSE. Mas NOT FALSE é TRUE enquanto que NOT UNKNOWN é UNKNOWN. Ver a parte dos operadores lógicos aqui ao lado.

Algumas vistas úteis

Vistas ALL_	Vistas USER_
Todos os objectos que o utilizador pode ver.	Objectos do utilizador.
ALL_CATALOG	USER_CATALOG
ALL_CONSTRAINTS	USER_CONSTRAINTS
ALL_CONS_COLUMNS	USER_CONS_COLUMNS
ALL_ERRORS	USER_ERRORS
ALL_EXTENTS	USER_EXTENTS
ALL_FREE_SPACE	USER_FREE_SPACE
ALL_INDEXES	USER_INDEXES
ALL_IND_COLUMNS	USER_IND_COLUMNS
ALL_OBJECTS	USER_OBJECTS
ALL_OBJECT_SIZE	USER_OBJECT_SIZE
ALL_SEGMENTS	USER_SEGMENTS
ALL_SEQUENCES	USER_SEQUENCES
ALL_SNAPSHOTS	USER_SNAPSHOTS
ALL_SOURCE	USER_SOURCE
ALL_SYNONYMS	USER_SYNONYMS
ALL_TABLES	USER_TABLES
ALL_TABLESPACES	USER_TABLESPACES
ALL_TRIGGERS	USER_TRIGGERS
ALL_VIEWS	USER_VIEWS

Operadores

Operadores de conjunto

Ver secção Junções→Operadores de Conjunto.

Operadores Aritméticos

`+, -, *, /`

Operadores de Concatenação

`||` Para concatenar strings; ver função concat

Outros operadores

`(+)` Junção externa.

`PRIOR` Para SELECTs hierárquicos (CONNECT BY).

User-defined operators

Operadores lógicos

NOT

	TRUE	FALSE	UNKNOWN
NOT	FALSE	TRUE	UNKNOWN

AND

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

OR

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

SELECTs simples

Sintaxe

```
SELECT [DISTINCT] lista_cols
  FROM lista_tabs
 [WHERE condições]
 [GROUP BY lista_cols]
[HAVING condições_grupo]
 [ORDER BY lista_cols]
```

Ver dados do sistema

- Ver a data do sistema

```
SELECT sysdate FROM dual;
```

Ver dados do sistema

- Nome do utilizador da BD

```
SELECT user FROM dual;
```

- Objectos do utilizador

```
SELECT * FROM
user_objects;
```

- As vistas a que tem acesso

```
SELECT view_name
  FROM all_views;
```

WHERE, ORDER BY, concatenação, *alias*, NVL

- Vendedores do dep 10, ordenados desc/mente por sal e em caso de empate por nome asc/mente.

```
SELECT * FROM emp
 WHERE ndep = 10 AND funcao = 'Vendedor'
 ORDER BY sal DESC, nome ASC;
```

Nota: As *strings* aparecem limitadas por plicas.

- Concatenação de colunas e uma constante. Uso do *alias* "Descrição".

```
SELECT nome||' trabalha no dep '||ndep "Descrição"
  FROM emp;
```

Nota: Os alias aparecem limitadas por aspas.

- Uso de NVL (Null VaLue). Se 1º parâm. For NULL substitui devolve o 2º. Caso contrário devolve o primeiro.

```
SELECT nome,
       sal*14 + NVL(premios, 0) "Remuneracao Annual"
  FROM emp;
```

Nota: Os nulos são sempre um caso especial em funções de linha.

DISTINCT

- Uso de DISTINCT. Mostra as funções, elimina repetidos. Também se usa em funções de grupo.

```
SELECT DISTINCT funcao
  FROM emp;
```

Pattern matching e uso de *wildcards*

- Encontra todos os nomes que comecem por 'A', tenham um número qq de letras, tenham um espaço, e tenham um 'R' na penúltima posição da string. A última letra pode ser uma qq.

```
SELECT *
  FROM emp
 WHERE nome LIKE 'A% R_';
```

- Encontra nomes que começam por 'A' e terminam em '_'.

```
SELECT *
  FROM emp
 WHERE nome LIKE 'A%#_' ESCAPE '#';
```

Junções

Junção interna com equi-junção

- Mostra os departamentos e os seus empregados. Uso de pseudónimos ou alias.

```
SELECT e.nome, d.nome
  FROM emp e, dep d
 WHERE e.ndep = d.ndep;
```

Nota: Se FROM tem n tabelas, WHERE tem que ter n-1 restrições de junção (assumindo que não há chaves concatenadas).

Junção interna com não equi-junção

- Mostra os empregados e os seus escalões.

```
SELECT e.nome, escalao
  FROM emp e, descontos
 WHERE sal BETWEEN salinf AND salsup;
```

Junção externa

- Mostra os departamentos e os seus empregados mesmo para departamentos sem nenhum empregado. Aparecem linhas a NULL na tabela com o símbolo de junção externa, (+), para todas as linhas da outra tabela que ainda não tinham sido seleccionadas.

```
SELECT e.nome, d.nome
  FROM emp e, dep d
 WHERE e.ndep (+) = d.ndep;
```

Junção com a própria tabela ou *self-join*

- Mostra os empregados e os seus escalões. Usa junção externa para mostrar os empregados sem subordinados. Neste caso, o uso de pseudónimos é obrigatório.

```
SELECT e.nome Chefe, e.nome Subordinado
  FROM emp e1, emp e2
 WHERE e1.nemp = e2.encar (+);
```

Operadores de conjuntos

- UNION — Junta 2 SELECTs. Elimina repetidos.
  UNION ALL — Junta 2 SELECTs. **NÃO** elimina repetidos.
  INTERSECT — As linhas em comum de 2 SELECTs.
  MINUS — As linhas do 1º SELECT menos as do 2º.

Exemplo:

```
SELECT funcao
  FROM emp e
 WHERE ndep = 10
UNION | UNION ALL | INTERSECT | MINUS
SELECT funcao
  FROM emp e
 WHERE ndep = 30;
```

Nota: Os SELECTs têm que concordar em número e tipo de colunas. Só existe um ORDER BY no fim de tudo. Só o primeiro SELECT define os pseudónimos a mostrar. O ORDER BY pode usar esses pseudónimos.

Funções de grupo

Regras

- Devolvem apenas um valor para um conjunto de linhas.
- Podem ser usadas nas cláusulas SELECT e HAVING.
- Cada um dos valores devolvidos é calculado para o conjunto de registos definidos pela cláusula GROUP BY.
- Sem GROUP BY os valores são calculados para a tabela toda.
- Todas as funções de grupo ignoram os nulos excepto o COUNT(*).
- A cláusula DISTINCT permite eliminar os repetidos.

Nota: Não se pode misturar funções de grupo com funções que não são de grupo.

Nota: HAVING apenas para restrições grupo, WHERE apenas para de linha.

Lista de funções de grupo

AVG	COUNT	GROUPING	MAX
MIN	STDDEV	SUM	VARIANCE

Exemplos

- Salário máximo e quantidade de pessoas para cada função para funções com mais de 2 pessoas.

```
SELECT funcao,
       max(sal) "Max Sal",
       count(*) "Quantidade"
  FROM emp
 GROUP BY funcao
HAVING count(*) >= 2;
```

- Quantidade diferente de funções

```
SELECT COUNT(DISTINCT funcao) FROM emp;
```

Funções de linha

Numéricas

ABS	EXP	SIGN
ACOS	FLOOR	SIN
ATAN	LN	SINH
ATAN2	LOG	SQRT
CEIL	MOD	TAN
COS	POWER	TANH
COSH	ROUND (number)	TRUNC (number)

De caracter devolvendo caracter

CHR	NLS_LOWER	SUBSTR
CONCAT	NLSSORT	SUBSTRB
INITCAP	NLS_UPPER	TRANSLATE
LOWER	REPLACE	TRIM
LPAD	RPAD	UPPER
LTRIM	RTRIM	
NLS_INITCAP	SOUNDEX	

De caracter devolvendo número

ASCII	INSTRB	LENGTHB
INSTR	LENGTH	

Data

ADD_MONTHS	NEW_TIME	SYSDATE
LAST_DAY	NEXT_DAY	TRUNC (date)
MONTHS_BETWEEN	ROUND (date)	

Conversão

CHARTOROWID	TO_CHAR (date)	TO_NUMBER
CONVERT	TO_CHAR (number)	TO_SINGLE_BYTE
HEXTORAW	TO_DATE	TRANSLATE ... USING
RAWTOHEX	TO_LOB	
ROWIDTOCHAR	TO_MULTI_BYTE	

Miscelâneas

BFILENAME	NVL
DUMP	SYS_CONTEXT
EMPTY_[B | C]LOB	SYS_GUID
GREATEST	UID
LEAST	USER
NLS_CHARSET_DECL_LEN	USERENV
NLS_CHARSET_ID	VSIZE
NLS_CHARSET_NAME	

Object Reference

DEREF	REF	VALUE
MAKE_REF	REFTOHEX	

Nota: As funções não podem ser executadas na linha de comando directamente. Têm que ser executadas sempre através de um comando SQL (ou com PL/SQL). Podem ser chamadas umas dentro das outras.

Nota: Quando se pretende chamar uma função cujos parâmetros de entrada (se os tiver) não estão associados a nenhuma tabela em particular pode-se usar a tabela **DUAL**. Por exemplos:

```
SELECT sysdate FROM dual;
SELECT userenv('CLIENT_INFO') FROM dual;
```

Subconsultas

- **Uma coluna**. Empregado(s) com o salário mais alto.

```
SELECT * FROM emp
 WHERE sal = (SELECT max(sal) FROM emp);
```

- **Duas colunas**. Empregado(s) com o salário mais alto por função.

```
SELECT * FROM emp
 WHERE (funcao, sal) IN (SELECT funcao, max(sal)
                           FROM emp
                          GROUP BY funcao)
```

- Subconsulta **correlacionada**. Empregados com salário superior à média do seu departamento.

```
SELECT nemp, nome, sal, ndep FROM emp e
 WHERE sal > (SELECT avg(sal) FROM emp
                WHERE ndep = e.ndep)
 ORDER BY ndep ASC, sal DESC;
```

Correlação

- Subconsulta **correlacionada e operador EXISTS**. Departamentos sem empregados.

```
SELECT * FROM dep d
 WHERE NOT EXISTS (SELECT * FROM emp
                     WHERE ndep = d.nep);
```

Correlação

- Subconsulta na **cláusula FROM**.

```
SELECT e.nome, e.sal, depmax.maxsal, depmax.ndep
  FROM emp e,
       (SELECT max(sal) maxsal, ndep
          FROM emp
         GROUP BY ndep) depmax
 WHERE e.ndep = depmax.ndep
 ORDER BY e.ndep, e.sal DESC;
```

Alias da coluna

Alias da tabela