

Manipulação de dados relacionais

Sistemas Informáticos
Licenciatura em Design e Multimédia

Sumário

- Consultas
- Criação/eliminação de tabelas e inserção, remoção, atualização de registo
- Transações

Manipulação e utilização de dados relacionais

- **Definição de dados** (criar e alterar tabelas, vistas, etc)
- **Integridade** (definir regras de integridade dos dados)
- **Interrogação** (pesquisar dados em tabelas)
- **Manipulação de dados** (inserir, alterar, apagar dados)
- **Autorizações e segurança** (definir privilégios, perfis, etc)
- **Controlo de transacções** (iniciar e terminar transacções)

SQL
(Structured
Query Language)

SQL é um standard ANSI e ISO desde 1986

SQL: Exemplo de alguns comandos

SELECT

Comando usado para pesquisar dados na base de dados.

CREATE

ALTER

DROP

Comandos utilizados para criar, alterar ou eliminar qualquer estrutura de dados, tais como tabelas, vistas e índices (comandos DDL).

INSERT

UPDATE

DELETE

Comandos para inserir novos registos em tabelas, alterar registos já existentes ou eliminá-los (comandos DML).

COMMIT

SAVEPOINT

ROLLBACK

Comandos utilizados para controlar explicitamente transacções.

GRANT

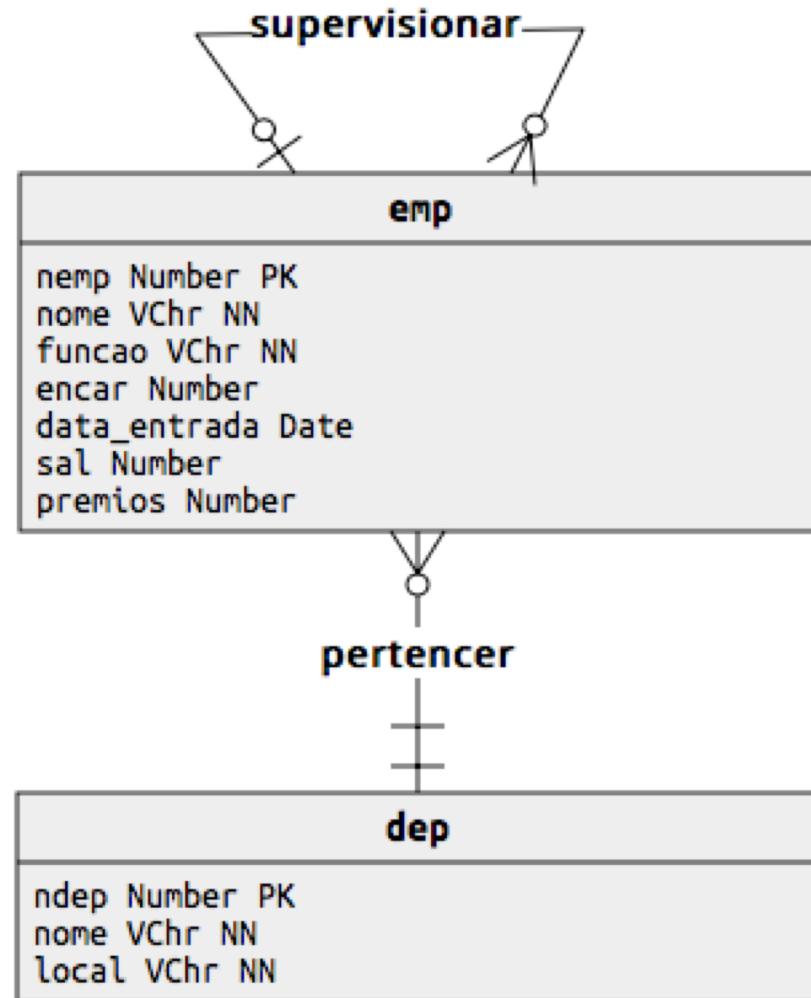
REVOKE

Comandos para atribuir ou retirar direitos de acesso à base de dados ou a estruturas específicas da base de dados.

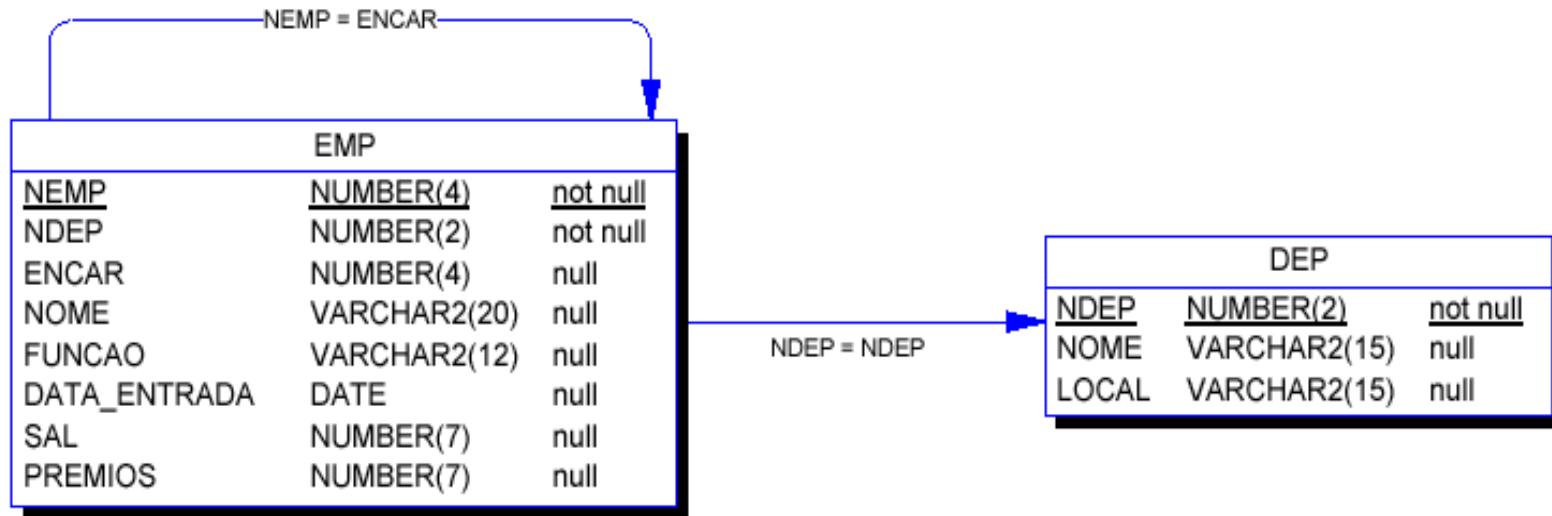
O problema

- Uma empresa é composta por vários departamentos. Cada empregado trabalha num departamento e pode supervisionar vários outros empregados (mesmo que trabalhem noutras departamentos). Um empregado é supervisionado no máximo por um outro empregado. O Presidente desta empresa não tem nenhum supervisor.

Modelo conceptual



Modelo físico



O Comando SELECT (simplificado)

- Para consultar a informação presente nas diversas tabelas da base de dados podemos usar o comando SELECT:

SELECT que atributos (i.e., colunas) queremos ver
FROM em que tabela(s) se encontra a informação
WHERE qual o(s) critério(s) para a escolha dos dados
(e.g., queremos ver apenas 5 registos, ou
apenas registos com características especiais)

Exemplo simples do comando SELECT

- Ver todos os dados de uma tabela

```
SQL> select * from emp;
```

NEMP	NOME	FUNCAO	ENCAR	DATA_ENT	SAL	PREMIOS	NDEP
1839	Jorge Sampaio	Presidente		84.02.11	890000		10
1566	Augusto Reis	Encarregado	1839	85.02.13	450975		20
1698	Duarte Guedes	Encarregado	1839	91.11.25	380850		30
1782	Silvia Teles	Encarregado	1839	86.11.03	279450		10
1788	Maria Dias	Analista	1566	82.11.07	565000		20
1902	Catarina Silva	Analista	1566	93.04.13	435000		20
1499	Joana Mendes	Vendedor	1698	84.10.04	145600	56300	30
1521	Nelson Neves	Vendedor	1698	83.02.27	212250	98500	30
1654	Ana Rodrigues	Vendedor	1698	90.12.17	221250	81400	30
1844	Manuel Madeira	Vendedor	1698	85.04.21	157800	0	30
1900	Tome Ribeiro	Continuo	1698	94.03.05	56950		30
1876	Rita Pereira	Continuo	1788	96.02.07	65100		20
1934	Olga Costa	Continuo	1782	86.06.22	68300		10
1369	Antonio Silva	Continuo	1902	96.12.22	70800		20

14 rows selected.

Exercício 1

Selecione toda a informação da tabela DEP. O resultado deve ser o seguinte:

NDEP	NOME	LOCAL
10	Contabilidade	Condeixa
20	Investigação	Mealhada
30	Vendas	Coimbra
40	Planeamento	Montemor

Operadores relacionais (breve introdução)



Operadores relacionais

Operadores unários

- Restrição
- Projeção

Operadores binários

- União
- Intersecção
- Diferença

- Divisão
- Prod. cartesiano
- Junção

Projeção

- Quando não queremos ver todas as colunas
- Operação que permite selecionar uma ou mais colunas de uma tabela e criar uma nova tabela (resultado).

```
SELECT nome, funcao  
FROM emp;
```

Mostra apenas os atributos **nome** e **funcao** da tabela **emp**

NOME	FUNCAO
Jorge Sampaio	Presidente
Augusto Reis	Encarregado
Duarte Guedes	Encarregado
Silvia Teles	Encarregado
Maria Dias	Analista
Catarina Silva	Analista
Joana Mendes	Vendedor
Nelson Neves	Vendedor
Ana Rodrigues	Vendedor
Manuel Madeira	Vendedor
Tome Ribeiro	Continuo
Rita Pereira	Continuo
Olga Costa	Continuo
Antonio Silva	Continuo

14 rows selected.

Mostra todos os registos existentes na tabela pois não há qualquer restrição (i.e., não especificámos uma cláusula *where*).

Exercício 2

Mostre a lista de todos os empregados contendo o nome de cada empregado, a sua função, o salário e o número do departamento a que pertence.

NOME	FUNCAO	SAL	NDEP
Jorge Sampaio	Presidente	890000	10
Augusto Reis	Encarregado	450975	20
Duarte Guedes	Encarregado	380850	30
Silvia Teles	Encarregado	279450	10
Maria Dias	Analista	565000	20
Catarina Silva	Analista	435000	20
Joana Mendes	Vendedor	145600	30
Nelson Neves	Vendedor	212250	30
Ana Rodrigues	Vendedor	221250	30
Manuel Madeira	Vendedor	157800	30
Tome Ribeiro	Continuo	56950	30
Rita Pereira	Continuo	65100	20
Olga Costa	Continuo	68300	10
Antonio Silva	Continuo	70800	20

Projeção e valores únicos

```
SELECT funcao  
FROM emp;
```

FUNCAO

Presidente
Encarregado
Encarregado
Encarregado
Analista
Analista
Vendedor
Vendedor
Vendedor
Vendedor
Continuo
Continuo
Continuo
Continuo

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

FUNCAO

Presidente
Encarregado
Continuo
Vendedor
Analista

Restrição

- Operação que permite seleccionar regtos de uma tabela que satisfazem um dada condição.

```
SELECT *  
FROM emp  
WHERE ndep = 10;
```

Mostra todos os atributos (i.e., todas as colunas) da tabela **emp**

Mas apenas para os regtos que verificam a condição de **ndep = 10**

Resultado:

NEMP	NOME	FUNCAO	ENCAR	DATA_ENT	SAL	PREMIOS	NDEP
1839	Jorge Sampaio	Presidente		84.02.11	890000		10
1782	Silvia Teles	Encarregado	1839	86.11.03	279450		10
1934	Olga Costa	Continuo	1782	86.06.22	68300		10

Restrição: Operadores (1)

- É possível também usar outros operadores para além do '='

Operador	Descrição
=	Igual a
<>	Diferente de
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a
BETWEEN	Dentro de um intervalo
LIKE	Igual a um padrão
IN	Dentro de uma lista de valores

```
SELECT *
FROM emp
WHERE ndep <> 10;
```

Restrição: Operadores (2)

- Para escrevermos cláusulas *where* válidas é necessário formatar **dados** correctamente com o uso de **apóstrofos**
- Dados numéricos não necessitam de apóstrofos, ao contrário de outros tipos de dados (e.g., character, varchar, date, timestamp, etc.).

```
SELECT *  
FROM emp  
WHERE ndep = 10;
```

```
SELECT *  
FROM emp  
WHERE nome = 'Manuel Madeira';
```

- Em alguns SGBDs podemos também usar os operadores ‘<’ e ‘>’ para comparar datas

Restrição: Operadores (3)

- Utilização do operador **IN**

```
SELECT *  
FROM emp  
WHERE ndep IN (10, 30)
```

- O operador **LIKE** é usado para pesquisar um padrão
- Os padrões definem-se com alguns caracteres especiais
- Para especificar zero ou mais letras desconhecidas no padrão usa-se o caracter %
- O que fará a seguinte query?

```
SELECT *  
FROM emp  
WHERE nome LIKE '%Silva'
```

Exercício 3

- Apresente a lista de todos os empregados (toda a informação) que recebem prémios.

NEMP	NOME	FUNCAO	ENCAR	DATA_ENTRADA	SAL	PREMIOS	NDEP
1499	Joana Mendes	Vendedor	1698	4/10/84	145600	56300	30
1521	Nelson Neves	Vendedor	1698	27/2/83	212250	98500	30
1654	Ana Rodrigues	Vendedor	1698	17/12/90	221250	81400	30
1844	Manuel Madeira	Vendedor	1698	21/4/85	157800	0	30

Restrição: Operadores lógicos

- Podem ser usadas condições múltiplas usando os operadores **AND, OR e NOT**.

```
SELECT *
FROM emp
WHERE ndep = 10 OR ndep = 30;
```

NEMP	NOME	FUNCAO	ENCAR	DATA_ENT	SAL	PREMIOS	NDEP
1839	Jorge Sampaio	Presidente		11/2/84	890000		10
1698	Duarte Guedes	Encarregado	1839	25/11/91	380850		30
1782	Silvia Teles	Encarregado	1839	3/11/86	279450		10
1499	Joana Mendes	Vendedor	1698	4/10/84	145600	56300	30
1521	Nelson Neves	Vendedor	1698	27/2/83	212250	98500	30
1654	Ana Rodrigues	Vendedor	1698	17/12/90	221250	81400	30
1844	Manuel Madeira	Vendedor	1698	21/4/85	157800	0	30
1900	Tome Ribeiro	Continuo	1698	5/3/94	56950		30
1934	Olga Costa	Continuo	1782	22/6/86	68300		10

- Os operadores podem ser combinados em condições mais complexas (use parêntesis para agrupar as sub-condições)

Restrição: Verificação de valores nulos

- Voltando ao exercício 3, queremos listar todos os empregados que recebem prémios. Solução imediata:

```
SELECT *
FROM emp
WHERE premios >= 0
```

- Mas... como listar todos os empregados que **não** recebem prémios?
- Por outras palavras, queremos obter todas as linhas onde a coluna **premios** esteja vazia (**nula**), isto é, tenha o valor especial **NULL**

```
SELECT *
FROM emp
WHERE premios IS NULL
```

- Valores **NULL** são tratados de forma diferente dos restantes
- NULL não é equivalente a zero** (não são comparáveis)

Restrição: Verificação de valores não nulos

- Solução alternativa para o exercício 3 (listar todos os empregados que recebem prémios):

```
SELECT *
FROM emp
WHERE premios IS NOT NULL
```

Restrição e projeção

- Podemos combinar as operações de restrição e de projeção.

```
SELECT nome, funcao, sal  
FROM emp  
WHERE ndep = 10 AND sal > 200;
```

Projeção: mostra apenas os atributos **nome**, **funcao** e **sal** da tabela **emp**

Restrição: Mostra apenas os registos que satisfazem a condição de ser do departamento 10 e ter salário maior do que 200.

Resultado:

NOME	FUNCAO	SAL
Jorge Sampaio	Presidente	890000
Silvia Teles	Encarregado	279450
Olga Costa	Continuo	68300

Exercício 4

- Mostre nome, função e salário dos empregados que são vendedores ou contínuos.

NOME	FUNCAO	SAL
Joana Mendes	Vendedor	145600
Nelson Neves	Vendedor	212250
Ana Rodrigues	Vendedor	221250
Manuel Madeira	Vendedor	157800
Tome Ribeiro	Continuo	56950
Rita Pereira	Continuo	65100
Olga Costa	Continuo	68300
Antonio Silva	Continuo	70800

Ordenar resultados (1)

```
SELECT *  
FROM emp  
ORDER BY nome
```

Nome da(s) coluna pela qual queremos ordenar os resultados.
Se for necessário especificar duas ou mais colunas, devemos separá-las por vírgula

NEMP	NOME	FUNCAO	ENCAR	DATA_ENT	SAL	PREMIOS	NDEP
1654	Ana Rodrigues	Vendedor	1698	17/12/90	221250	81400	30
1369	Antonio Silva	Continuo	1902	22/12/96	70800		20
1566	Augusto Reis	Encarregado	1839	13/2/85	450975		20
1902	Catarina Silva	Analista	1566	13/4/93	435000		20
1698	Duarte Guedes	Encarregado	1839	25/11/91	380850		30
...							

- A cláusula **ORDER BY** coloca-se a seguir à cláusula **WHERE**, caso esta exista

Ordenar resultados (2)

- Pode-se adicionar **ASC** ou **DESC** à consulta para indicar que os resultados devem ser apresentados por ordem **crescente** ou **decrescente**, respectivamente
- O que faz a seguinte query?

```
SELECT *
FROM emp
ORDER BY funcao DESC
```

Ordenar resultados (2)

- Pode-se adicionar **ASC** ou **DESC** à consulta para indicar que os resultados devem ser apresentados por ordem **crescente** ou **decrescente**, respectivamente
- O que faz a seguinte query?

```
SELECT *
FROM emp
ORDER BY funcao DESC
```

NEMP	NOME	FUNCAO	ENCAR	DATA_ENT	SAL	PREMIOS	NDEP
1499	Joana Mendes	Vendedor	1698	4/10/84	145600	56300	30
1844	Manuel Madeira	Vendedor	1698	21/4/85	157800	0	30
1654	Ana Rodrigues	Vendedor	1698	17/12/90	221250	81400	30
1521	Nelson Neves	Vendedor	1698	27/2/83	212250	98500	30
1839	Jorge Sampaio	Presidente		11/2/84	890000		10
1698	Duarte Guedes	Encarregado	1839	25/11/91	380850		30
1566	Augusto Reis	Encarregado	1839	13/2/85	450975		20
1782	Silvia Teles	Encarregado	1839	3/11/86	279450		10
1369	Antonio Silva	Continuo	1902	22/12/96	70800		20
1876	Rita Pereira	Continuo	1788	7/2/96	65100		20
1934	Olga Costa	Continuo	1782	22/6/86	68300		10
1900	Tome Ribeiro	Continuo	1698	5/3/94	56950		30
1788	Maria Dias	Analista	1566	7/11/82	565000		20
1902	Catarina Silva	Analista	1566	13/4/93	435000		20

Ordenar resultados (3)

- O que fará esta query?

```
SELECT *
FROM emp
ORDER BY funcao ASC, nome DESC
```

Ordenar resultados (3)

- O que fará esta query?

```
SELECT *
FROM emp
ORDER BY funcao ASC, nome DESC
```

NEMP	NOME	FUNCAO	ENCAR	DATA_ENTRADA	SAL	PREMIOS	NDEP
1788	Maria Dias	Analista	1566	7/11/82	565000		20
1902	Catarina Silva	Analista	1566	13/4/93	435000		20
1900	Tome Ribeiro	Continuo	1698	5/3/94	56950		30
1876	Rita Pereira	Continuo	1788	7/2/96	65100		20
1934	Olga Costa	Continuo	1782	22/6/86	68300		10
1369	Antonio Silva	Continuo	1902	22/12/96	70800		20
1782	Silvia Teles	Encarregado	1839	3/11/86	279450		10
1698	Duarte Guedes	Encarregado	1839	25/11/91	380850		30
...							

Exercício 5

Mostre os nomes de todos os analistas que trabalham no departamento 20.
Apresente o resultado por ordem alfabética crescente.

NOME

Catarina Silva

Maria Dias

Consultas sobre várias tabelas

- Produto cartesiano
- Junção interna
- Junção externa
- Junção com a própria tabela

Produto cartesiano

- O produto cartesiano de duas tabelas R1 e R2 é uma nova tabela R3 na qual cada registo de R1 é adicionado a todos os registos de R2.

Cliente

Nº_cliente	Endereço	Telefone
1022	Dias da Silva, 159	711228
1025	Av. Brasil, 21, 1º	722119

Factura

ID_Cliente	Factura
1022	101
1025	321

```
SELECT *
FROM Cliente, Factura
```

Produto1

Nº_cliente	Endereço	Telefone	ID_Cliente	Factura
1022	Dias da Silva, 159	711228	1022	101
1022	Dias da Silva, 159	711228	1025	321
1025	Av. Brasil, 21, 1º	722119	1022	101
1025	Av. Brasil, 21, 1º	722119	1025	321

O resultado do produto cartesiano é de utilidade muito limitada uma vez que os registos são associados sem qualquer critério.

Junção de tabelas

- No modelo de dados relacional os dados estão distribuídos em várias tabelas
- A junção de tabelas permite-nos obter informação integrada dos diversos dados
- Tipicamente, para efectuar uma consulta útil, usando várias tabelas, é necessário utilizar as chaves primárias e estrangeiras numa **cláusula de junção**

Junção interna (inner join) (1)

- Apenas os registos que satisfazem a **condição de junção** aparecem no resultado. **Exemplo:**

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

Condição de junção. Só os registos que satisfazem esta condição de junção é que aparecem no resultado.

Os dados destes dois atributos vêm da tabela emp.

Os dados deste atributo vêm da tabela dep.

NOME	FUNCAO	NOME
Jorge Sampaio	Presidente	Contabilidade
Silvia Teles	Encarregado	Contabilidade
Olga Costa	Continuo	Contabilidade
Augusto Reis	Encarregado	Investigação
Rita Ferreira		

Junção interna (inner join) (2)

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

Neste exemplo pode ver-se que só os registos de ambas as tabelas que têm o mesmo valor de **ndep** é que aparecem no resultado.

NOME	NDEP	NDEP NOME
Jorge Sampaio	10	10 Contabilidade
Silvia Teles	10	10 Contabilidade
Olga Costa	10	10 Contabilidade
Augusto Reis	20	20 Investigação
Rita Pereira	20	20 Investigação
Catarina Silva	20	20 Investigação
Maria Dias	20	20 Investigação
Antonio Silva	20	20 Investigação
Duarte Guedes	30	30 Vendas
Joana Mendes	30	30 Vendas
Nelson Neves	30	30 Vendas
Ana Rodrigues	30	30 Vendas
Manuel Madeira	30	30 Vendas
Tome Ribeiro	30	30 Vendas

14 rows selected.

Distinguir colunas com o mesmo nome em tabelas diferentes

```
SELECT emp.NOME, FUNCAO, dep.NOME  
FROM emp, dep  
WHERE dep.NDEP = emp.ndep;
```

Como o atributo nome aparece nas duas tabelas, para os distinguir é necessário inserir o nome da tabela antes.

```
SELECT e.NOME, e.FUNCAO, d.NOME  
FROM emp AS e, dep AS d  
WHERE d.NDEP = e.ndep;
```

Para facilitar, podemos dar um **pseudónimo** ao nome da tabela de modo a ser mais fácil escrever o comando. Por exemplo, a tabela **emp** passa a ser designada por **e** e **dep** por **d**.

Atribuição de pseudónimos a colunas

```
SELECT nome AS funcionario, sal AS salario  
FROM emp;
```

funcionario	salario
Jorge Sampaio	890000
Augusto Reis	450975
Duarte Guedes	380850
Silvia Teles	279450
Maria Dias	565000
Catarina Silva	435000
Joana Mendes	145600
Nelson Neves	212250
Ana Rodrigues	221250
Manuel Madeira	157800
Tome Ribeiro	56950
Rita Pereira	65100
Olga Costa	68300
Antonio Silva	70800

Junção, projecção e restrição

- Vulgarmente a junção usa-se em conjunto com a projecção e a restrição, de modo a eliminar colunas e registo desnecessários para o resultado.

```
SELECT emp.nome, funcao, dep.nome  
FROM emp, dep  
WHERE emp.ndep = dep.ndep  
And sal > 220000;
```

Projecção: só mostra alguns atributos.

Junção: mostra dados das duas tabelas, emp e dep

NOME	FUNCAO	NOME
Jorge Sampaio	Presidente	Contabilidade
Silvia Teles	Encarregado	Contabilidade
Augusto Reis	Encarregado	Investigação
Catarina Silva	Analista	Investigação
Maria Dias	Analista	Investigação
Duarte Guedes	Encarregado	Vendas
Ana Rodrigues	Vendedor	Vendas

7 rows selected.

Restrição: só mostra alguns registo.

Exercício 6

- Mostre o nome, função, salário e local de trabalho de todos os empregados de Coimbra cujo salário é superior a 150000.

NOME	FUNCAO	SAL	LOCAL
Duarte Guedes	Encarregado	380850	Coimbra
Nelson Neves	Vendedor	212250	Coimbra
Ana Rodrigues	Vendedor	221250	Coimbra
Manuel Madeira	Vendedor	157800	Coimbra

Funções de grupo

- Actuam sobre grupos de registo
- Produzem um valor por cada grupo de registo
- Por omissão, todos os registo de uma tabela
são considerados um único grupo

Funções de grupo – Exemplos

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

```
SELECT max(sal), avg(sal) FROM EMP;
```

Funções de grupo

- Como saber o número de empregados **por departamento?**

```
SELECT count(*)
```

```
FROM emp
```

```
GROUP BY ndep
```

```
SELECT count(*), dep.nome
```

```
FROM emp, dep
```

```
WHERE emp.ndep = dep.ndep
```

```
GROUP BY emp.ndep, dep.nome
```

SQL – funções de grupo

- AVG() - Returns the average value
- COUNT() - Returns the number of rows
- FIRST() - Returns the first value
- LAST() - Returns the last value
- MAX() - Returns the largest value
- MIN() - Returns the smallest value
- SUM() - Returns the sum

Exercício 7

- Mostre o maior salário por cada função diferente presente na base de dados

Restrições sobre os grupos

- Usamos a cláusula HAVING para aplicar restrições sobre os grupos gerados por uma expressão GROUP BY

```
SELECT count(*)  
FROM emp  
GROUP BY ndep  
HAVING count(*) > 5
```

Funcionamento do comando SELECT

- As linhas e as colunas são pré-selecionadas com informação das claúsulas SELECT e WHERE.
- Esse conjunto de linhas é distribuido por grupos em função da informação existente na cláusula GROUP BY. Para cada um dos grupos são calculadas as respectivas funções de grupo que aparecem na cláusula SELECT.
- Só são mostrados os grupos que passarem nas condições de HAVING.
- Há casos especiais que se podem resolver tanto com restrições em WHERE como em HAVING (quando uma coluna é simultaneamente uma característica do registo individual e do grupo)

Subconsultas

- Como consultar todos os funcionários cujo salário é superior à média?
- Obter o salário médio:

```
SELECT avg(sal) FROM emp
```

- Fazer uma consulta aos resultados da (sub)consulta:

```
SELECT * FROM emp
```

```
WHERE sal > (SELECT avg(sal) FROM emp)
```

Alguns comandos DDL da linguagem SQL

- **CREATE** ← Cria tabelas
- **DROP** ← Remove tabelas

*DDL – Data Definition Language

Criação de uma tabela

```
CREATE TABLE nome_da_tabela
(
    Nome_coluna    tipo(tamanho)      restrições_de_integridade,
    Nome_coluna    tipo(tamanho)      restrições_de_integridade,
    ...
);
```

restrições_de_integridade,
restrições_de_integridade,

Restrições de integridade:
Condições a que os valores a guardar nas colunas/tabelas devem obedecer.

Exemplo:

```
CREATE TABLE ELEITOR
(
    BI           NUMERIC(8)        UNIQUE,
    NOME         CHAR(60)          NOT NULL,
    ENDERECO     CHAR(120),
    N_ELEITOR    NUMERIC(5),
    COD_FREG    NUMERIC(4),
    PRIMARY KEY (COD_FREG, N_ELEITOR));
```

Tipos de dados SQL

- A especificação SQL define um conjunto de tipos de dados: boolean, integer, varchar, etc.
- Cada SGBD oferece as suas opções para definição de tipos de dados:

Data type	SQLServer	Oracle	MySQL	PostgreSQL	...
Boolean	Bit	Byte	Tinyint	Boolean	...
...

http://www.w3schools.com/sql/sql_datatypes_general.asp

Eliminar uma tabela

```
DROP TABLE nome_da_tabela;
```

Exemplo:

```
DROP TABLE ELEITOR;
```

- A eliminação de uma tabela causa a perda de todos os dados nelas existentes, assim como todos os índices associados.
- Uma tabela só pode ser apagada por quem a criou ou por quem tenha privilégios específicos para isso.

Geração de SQL (DROP e CREATE)

```
DROP TABLE emp;  
DROP TABLE dep;  
  
CREATE TABLE dep  
(  
    ndep ... PRIMARY KEY  
    ...  
);  
  
CREATE TABLE emp  
(  
    nemp ... PRIMARY KEY  
    ndep ... REFERENCES dep(ndep)  
    ...  
);
```

Assumindo que não existem erros de sintaxe, o que acontece ao executarmos este *script* sobre uma base de dados **vazia**?

Erro na primeira linha! A tabela emp não existe ainda, não pode ser apagada! A execução do script é interrompida.

Note ainda que a ordem das duas instruções CREATE é importante. Porquê?

A coluna **ndep** na tabela **emp** referencia a coluna **ndep** na tabela **dep**.

Logo, a tabela **dep** tem de ser criada primeiro. Caso as instruções CREATE estivessem colocadas por outra ordem, a execução do script seria interrompida por altura do primeiro CREATE

Comandos DML da linguagem SQL

- **INSERT** ← Insere dados numa tabela
- **UPDATE** ← Actualiza dados numa tabela
- **DELETE** ← Remove dados de uma tabela
(não apaga a tabela!)

*DML – Data Manipulation Language. O comando SELECT também é um comando DML

Inserir novos registos numa tabela

```
INSERT INTO nome_da_tabela (coluna1, coluna2,...)  
VALUES (valor1, valor2, ...);
```

Exemplo:

```
INSERT INTO ELEITOR (BI, NOME, ENDERECO, N_ELEITOR)  
VALUES (4537687, 'António Silva', 'R. Carlos seixas, 29, 1º, Esq', 2075);
```

- Apenas se pode inserir um registo de cada vez (quando se indica os valores explicitamente no comando).
- Pode-se omitir a lista de nomes de colunas desde que se respeite a ordem das colunas definida na tabela e não se omita nenhum valor.
- Pode-se inserir apenas alguns campos de um registo, ficando, neste caso, os campos omitidos com valores nulos.
 - Alternativa: Colocar a palavra **NULL** como valor do campo que não deve ser preenchido

Actualizar registos

```
UPDATE nome_da_tabela [pseudónimo_opcional]  
SET coluna = valor,  
    coluna = valor,  
    ....  
WHERE condição;
```

Exemplo:

```
UPDATE ELEITOR  
SET NOME = 'António Dias da Silva',  
    BI = 4537687  
WHERE N_ELEITOR = 2075;
```

- Se a cláusula WHERE for omitida todos os registos da tabela são actualizados.
- **[SQL avançado]** Os **valores** a actualizar podem ser o resultado de **subconsultas** ou **expressões**.

Apagar registos

```
DELETE FROM nome_da_tabela  
WHERE condição;
```

Exemplo:

```
DELETE FROM ELEITOR  
WHERE N_ELEITOR = 2075;
```

- Se a cláusula WHERE for omitida todos os registos da tabela são apagados.

Preservação da integridade dos dados

- Restrições de integridade **definidas**:
 - na criação de tabelas
 - na alteração de tabelas
- Restrições de integridade **verificadas** na execução de:
 - INSERT
 - UPDATE
 - DELETE

Exemplos de restrições

```
CREATE TABLE ALUNO
(
    A_NUM      NUMERIC(4)  PRIMARY KEY,
    NOME       CHAR(60)      NOT NULL,
    ENDER     CHAR(120),
    TEL        NUMERIC(8),
    BI         NUMERIC(4)  UNIQUE
);

CREATE TABLE DISCIPLINA
(
    NOME       CHAR(60)      UNIQUE,
    D_CODIG    CHAR(6)        PRIMARY KEY,
    ANO        NUMERIC(1)    CHECK ((ANO >= 1) AND (ANO <= 5)),
    SEM        NUMERIC(1)    CHECK (SEM IN (1,2))
);

CREATE TABLE A_D
(
    A_NUM          NUMERIC(4),
    D_CODIG        CHAR(6),
    EPOCA          CHAR(6),
    NOTA          NUMERIC(2) CHECK ((NOTA >= 0) AND (NOTA <= 20)),
    CONSTRAINT A_D_CE1 FOREIGN KEY      (A_NUM)      REFERENCES ALUNO(A_NUM),
    CONSTRAINT A_D_CE2 FOREIGN KEY      (D_CODIG)    REFERENCES DISCIPLINA(D_CODIG),
    CONSTRAINT A_D_CP  PRIMARY KEY      (A_NUM, D_CODIG, EPOCA)
);
```

Violação de restrições de integridade: Exemplos (1)

Registros presentes nas tabelas

Aluno

(P) A_NUM	NOME	ENDER	TEL	BI
1002	A. Mendes	R. Sota, 23	1234	987654
1008	M. Melo	Av. Sá Band.	5342	457541

Disciplina

NOME	(P) D_CODIG	ANO	SEM
Álgebra	ALGB	1	1
Electrónica I	ELEC	2	2

A_D (P = A_NUM, D_CODIG, EPOCA)

(F) A_NUM	(F) D_CODIG	EPOCA	NOTA
1008	ALGB	Set.	15

```
INSERT INTO ALUNO
```

```
VALUES (1002, 'A. Cavaco', 'R. de Baixo, 23', 96455345310, 228620 )
```

Erro: viola integridade de entidade
pois o aluno nº 1002 já existe na tabela

Erro: O tamanho máximo do campo TEL é 8

```
INSERT INTO A_D
```

```
VALUES (1004, 'ELEC', 'Julho', 12);
```

Erro: viola a integridade referencial pois 1004 não
tem correspondência na tabela de referência 'Aluno'

```
INSERT INTO DISCIPLINA
```

```
VALUES ('Física', 'FISC', 0, 1);
```

Erro: viola teste de integridade na coluna Ano

Violação de restrições de integridade: Exemplos (2)

Registros presentes nas tabelas

Aluno

(P) A_NUM	NOME	ENDER	TEL	BI
1002	A. Mendes	R. Sota, 23	1234	987654
1008	M. Melo	Av. Sá Band.	5342	457541

Disciplina

NOME	(P) D_CODIG	ANO	SEM
Álgebra	ALGB	1	1
Electrónica I	ELEC	2	2

A_D (P = A_NUM, D_CODIG, EPOCA)

(F) A_NUM	(F) D_CODIG	EPOCA	NOTA
1008	ALGB	Set.	15

UPDATE ALUNO

```
SET NOME = 'D. Mendes', A_NUM = 1006  
WHERE A_NUM = 1008;
```

Erro: viola integridade referencial pois altera dado de referência correspondente à chave estrangeira A_NUM da tabela A_D

DELETE FROM ALUNO

```
WHERE A_NUM = 1008;
```

Erro: viola a integridade referencial pois há dados dependentes do registo a apagar na tabela A_D

Outros Exemplos (correctos!)

Registros presentes nas tabelas

Aluno

(P)	A_NUM	NOME	ENDER	TEL	BI
	1002	A. Mendes	R. Sota, 23	1234	987654
	1008	M. Melo	Av. Sá Band.	5342	457541

Disciplina

NOME	(P)	D_CODIG	ANO	SEM
Álgebra		ALGB	1	1
Electrónica I		ELEC	2	2

A_D (P = A_NUM, D_CODIG, EPOCA)

(F)	(F)	EPOCA	NOTA
1008	ALGB	Set.	15

```
INSERT INTO A_D (A_NUM, D_CODIGO, EPOCA)
VALUES (1002, 'ALGB', 'Set');
```

Correto, apesar de NOTA ficar a Null

```
UPDATE A_D
SET D_CODIG = 'ELEC', NOTA = 14;
```

Correto, pois a chave externa D_CODG é alterada mas mantém um valor que existe na tabela Disciplina

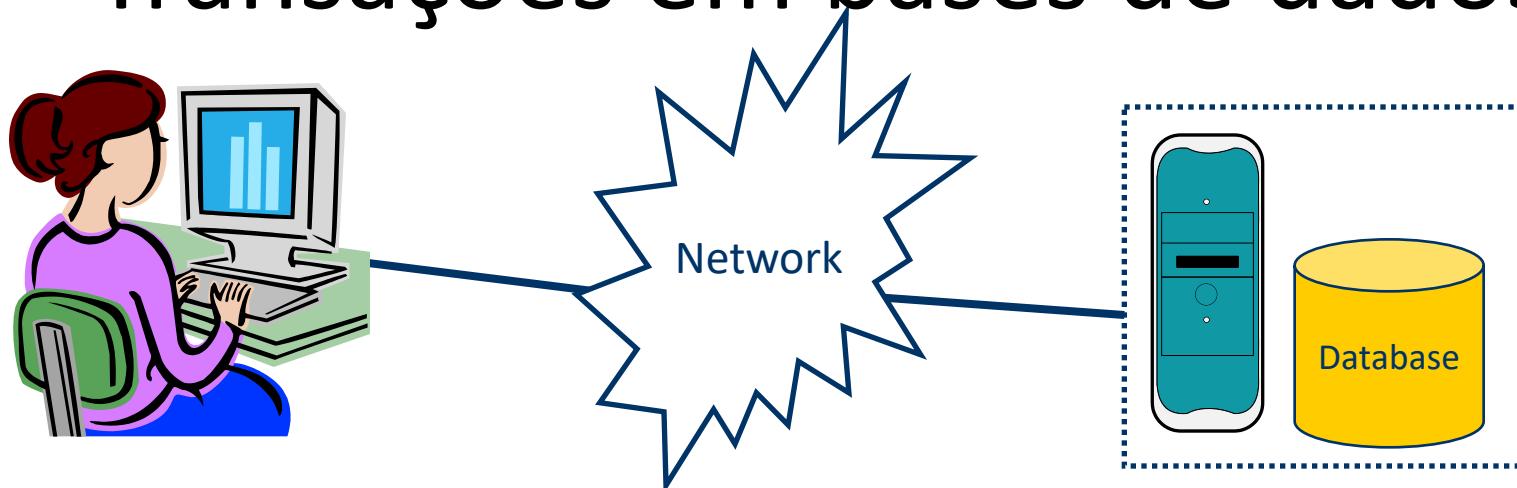
```
DELETE FROM A_D
WHERE A_NUM = 1008
```

Correto, eliminamos a relação entre o aluno 1008 e a disciplina ALGB

Integridade através de abordagem declarativa

- Evita a necessidade de programas para garantir a integridade dos dados
- Definição simples e rápida
- Testes de integridade optimizados pelo SQL
- Regras centralizadas e controladas pelo SGBD
- É fácil alterar restrições sem mexer nas aplicações
- Flexibilidade em activar/ desactivar restrições
- Note que apesar disto, **também as aplicações devem estar preparadas para antecipar (impedir), ou tratar, os casos de erro**
 - Exemplo: Inserção de valores duplicados (que deveriam ser únicos), remoção de registo referenciados por outras tabelas, ...
 - Mostrar mensagens que o utilizador (que não tem conhecimentos técnicos) possa compreender

Transações em bases de dados



“Transfere €1.000.000 da conta
43248932 para a conta 43298743”

```
1: boolean ok = account1.remove(1000000);
2: if (ok)
3: {
4:     account2.deposit(1000000);
5: }
```

- E se a aplicação tem um crash na linha 2??
- E se, na linha 4, a ligação de rede se perde??
- E se o servidor tem um crash entre a linha 1 e 2??

Porque as bases de dados suportam transações?

- Porque as pessoas usam bases de dados?
 - Separação entre lógica de negócio e gestão de dados
 - Desempenho no acesso e armazenamento de dados
 - **Garantia de integridade dos dados**
- Transação:
 - Um conjunto de operações que podem ser executadas como um todo, atomicamente
- Todas as bases de dados têm suporte direto para transações
- Dois tipos de implementação:
 - Redo and Undo logs
 - Shadowing (menos comum)

Transações

- Início bem definido
- Ou termina com um **COMMIT** ou existe um **ROLLBACK**
 - **COMMIT** → Todas as alterações se tornam permanentes
 - **ROLLBACK** → Todas as alterações são desfeitas
- Durante uma transação todas as alterações feitas são visíveis apenas para quem as está a fazer.
- Os restantes utilizadores/transações vêem os dados como se nada estivesse a acontecer

```
(...)  
trans.beginTransaction();  
  
boolean ok = account.remove(1000000);  
  
if (ok)  
{  
    account_2.deposit(1000000);  
    trans.commitTransaction();  
}  
else  
{  
    trans.rollbackTransaction();  
}
```

Transações e SQL

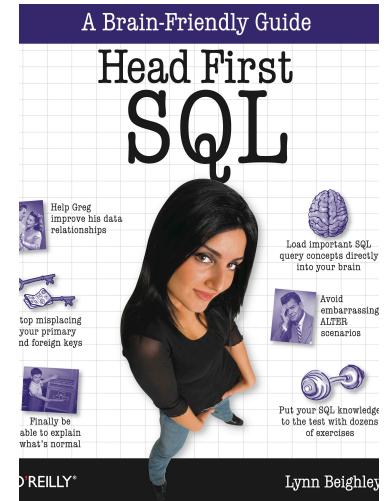
- BEGIN
 - Usado para marcar o inicio de uma transação
- COMMIT
 - Conclui a transação
- ROLLBACK
 - Desfaz a transação

ACID

- As transações são normalmente **ACID**
- **A – Atomic**: Todas as operações são efetuadas ou nenhuma é. Não existe meio termo.
- **C – Consistent**: O sistema transita de um ‘estado correto’ para um ‘estado correto’. I.e., as regras e restrições da aplicação (invariantes) não são violadas.
 - O sistema tem de ser implementado corretamente pelo programador.
- **I – Isolated**: Cada transação executa como se estivesse sozinha (isolada das restantes). As suas ações não são visíveis para além da sua fronteira.
 - Isto significa que, se várias transações executam concorrentemente o sistema comporta-se como se as tivesse executado uma após a outra (semântica serializável).
- **D – Durable**: Quando uma transação termina, não se pode reverter (é irreversível).
 - O sistema tem mecanismos para tolerar algumas falhas. Uma transação só conclui quando os dados estão escritos em armazenamento estável.

Para saber mais...

- **Head first SQL**
 - Chapter 1, 2, 3, 4
- **W3Schools SQL Tutorial**
 - <http://www.w3schools.com/sql>



I THINK WE SHOULD
BUILD AN SQL
DATABASE.

UH-OH

DOES HE UNDERSTAND
WHAT HE SAID OR
IS IT SOMETHING
HE SAW IN A TRADE
MAGAZINE AD?

S. Adams E-mail: SCOTTADAMS@AOL.COM

WHAT COLOR DO YOU
WANT THAT DATABASE?

I THINK
MAUVE HAS
THE MOST
RAM.

© 1995 United Feature Syndicate, Inc.(NYC)