SQL
Linguagem de Consulta Estruturada

- Introdução
- DML – Linguagem de Manipulação de Dados
  - SELECT
  - INSERT
  - UPDATE
  - DELETE
- DDL – Linguagem de Definição de Dados
  - CREATE
  - ALTER
  - DROP
- DCL – Linguagem de Controlo de Dados
  - Confidencialidade
  - Integridade

Introdução

- Objectivos
  - Permitir
    - Manipulação de dados, com possibilidade de utilização interactiva ou em programas de aplicação
    - Definição de dados
    - Definição de vistas
    - Definição de restrições de integridade
    - Definições de acesso (autorizações)
    - Manipulação de transacções
  - Permitir o tratamento de uma tabela (física ou lógica) como se fosse um simples operando

Introdução

- História
  - Interfaces Relacionais (1970)
  - SEQUEL (1974)
  - SQUARE (1975)
  - SEQUEL/2 (1976)
  - SQL (1980)
    - SQL-86
    - SQL-89
    - SQL-92 ou SQL2
    - … SQL3
- Características
  - Linguagem *set oriented*
  - Linguagem subdividida em
    - ***D**ata **M**anipulation **L**anguage* (DML)
    - ***D**ata **D**efinition **L**anguage* (DDL)
    - ***D**ata **C**ontrol **L**anguage* (DCL)

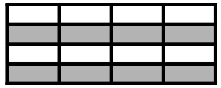
Introdução

- Características
  - Permite a sua integração noutras linguagens de programação
    - SQL embutido
    - PL/SQL (*Oracle*) e 4GL (*Informix*)
  - Comandos da linguagem
    - Cláusulas
    - Predicados
    - Funções
    - Não são sensíveis a maiúsculas/minúsculas
    - Terminam com **;**
    - Formato livre, mas sugere-se
      - Cada cláusula numa linha
      - Cláusulas alinhadas
      - Palavras reservadas em maiúsculas
    - Exemplo

```
SELECT DISTINCT nome
FROM t_empregado
WHERE localidade = 'LEIRIA';
```

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Permite efectuar consultas à base de dados
  - Aplica-se a uma ou mais tabelas ou vistas
  - Implementa todos os operadores da Álgebra Relacional
    - Operadores relacionais unários
      - Selecção ou Restrição



      - Projecção

    - Operadores relacionais binários
      - Produto cartesiano
      - Junção
      - União
      - Intersecção
      - Exclusão ou Diferença

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Operadores relacionais binários
    - Produto cartesiano

NomeAluno	Curso
João	1
Maria	1
José	2

Produto Cartesiano

CodCurso	NomeCurso
1	Eng. Informática
2	Gestão de Empresas

=

NomeAluno	Curso	CodCurso	NomeCurso
João	1	1	Eng. Informática
João	1	2	Gestão de Empresas
Maria	1	1	Eng. Informática
Maria	1	2	Gestão de Empresas
José	2	1	Eng. Informática
José	2	2	Gestão de Empresas

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Operadores relacionais binários
    - Junção

NomeAluno	Curso
João	1
Maria	1
José	2

Junção

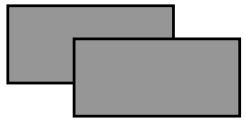
CodCurso	NomeCurso
1	Eng. Informática
2	Gestão de Empresas

=

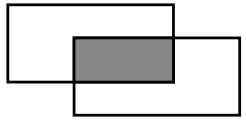
NomeAluno	Curso	CodCurso	NomeCurso
João	1	1	Eng. Informática
Maria	1	1	Eng. Informática
José	2	2	Gestão de Empresas

DML
Linguagem de Manipulação de Dados

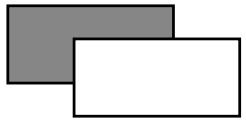
- Comando SELECT
  - Operadores relacionais binários
    - União



- Intersecção



- Exclusão ou Diferença



DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Sintaxe

```
SELECT [DISTINCT | ALL]
        {* | expressão_coluna [AS novo_nome][, ...]}
FROM nome_tabela [pseudónimo] [, ...]
[WHERE condições de restrição da tabela]
[GROUP BY <lista de colunas para agrupar>
[HAVING condições de restrição dos grupos]]
[ORDER BY <lista de colunas para ordenar o resultado>]
;
```

  - Exemplo

```
SELECT *
FROM t_aluno;
```

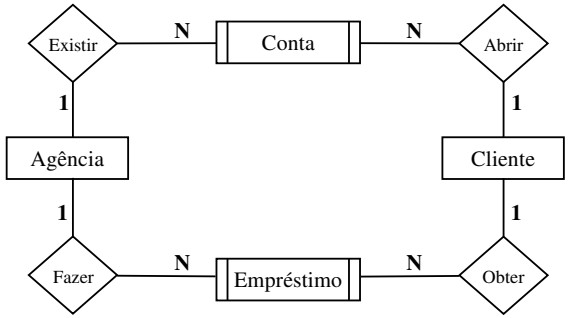
NomeAluno	Curso
João	1
Maria	1
José	2

DML
Linguagem de Manipulação de Dados

- Exemplo

Considerando só uma parte da BD de um banco

- Diagrama de Entidade-Relacionamento



- Relações Resultantes

Cliente (NCliente, nome, morada, telefone, profissao, idade)

Agencia (CodAgencia, designacao, localidade)

Conta (NConta, tipoC, saldo, CodAgencia → (Agencia), NCliente → (Cliente))

Emprestimo (NEmp, tipoE, valor, CodAgencia → (Agencia), NCliente → (Cliente))

DML
Linguagem de Manipulação de Dados

- Exemplo

Considerando só uma parte da BD de um banco

- Informação armazenada

TCliente

NCliente	Nome	Morada	Telefone	Profissao	Idade
100	João	Batalha	123456	Electricista	55
101	Manuel	Leiria		Carpinteiro	34
102	Antunes	Coimbra	345678	Arquitecto	29
103	Clara	Leiria	567890	Médica	40

TAgencia

CodAgencia	Localidade
200	Leiria
240	Batalha
250	Coimbra

TEmprestimo

NEmp	TipoE	Valor	CodAgencia	NCliente
1000	A	10.000	250	100
1001	A	20.000	200	101
1002	C	100.000	200	102
1003	B	50.000	200	103

TConta

NConta	TipoC	Saldo	CodAgencia	NCliente
10050	F	1.000	250	100
22220	A	25.000	200	102
30505	B	1.000	240	102
22555	B	4.000	200	103

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Consultas simples
    - Mostrar todas as colunas e todas as linhas
      - Exemplo

```
SELECT *
FROM TCliente;
```

NCliente	Nome	Morada	Telefone	Profissao	Idade
100	João	Batalha	123456	Electricista	55
101	Manuel	Leiria		Carpinteiro	34
102	Antunes	Coimbra	345678	Arquitecto	29
103	Clara	Leiria	567890	Médica	40

```
SELECT CodAgencia, localidade
FROM TAgencia;
```

CodAgencia	Localidade
200	Leiria
240	Batalha
250	Coimbra

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Consultas simples
    - Projecção
      - Exemplo
        - Mostrar n.º, nome e idade dos clientes

TCliente

NCliente	Nome	Morada	Telefone	Profissao	Idade
100	João	Batalha	123456	Electricista	55
101	Manuel	Leiria		Carpinteiro	34
102	Antunes	Coimbra	345678	Arquitecto	29
103	Clara	Leiria	567890	Médica	40

```
SELECT NCliente, Nome, Idade
FROM TCliente;
```

NCliente	Nome	Idade
100	João	55
101	Manuel	34
102	Antunes	29
103	Clara	40

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Consultas simples
    - Pseudónimos de colunas
      - Exemplos

```
SELECT CodAgencia  "Código da Agência"
FROM TAgencia;
```

```
SELECT CodAgencia  AS  "Código da Agência"
FROM TAgencia;
```

Código da Agência
200
240
250

```
SELECT CodAgencia  Código_da_ Agência
FROM TAgencia;
```

Código_da_Agência
200
240
250

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Consultas simples
    - Predicado DISTINCT
      - Exemplos

SELECT Morada
FROM TCliente;

Morada
Batalha
Leiria
Coimbra
Leiria

SELECT **DISTINCT** Morada
FROM TCliente;

Morada
Batalha
Leiria
Coimbra

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Consultas simples
    - Expressões em colunas
      - Concatenação (**||** ou **&**)
        - Exemplo

SELECT nome||'é o cliente número '||NCliente
AS "Identificação dos Clientes"
FROM TCliente;

Identificação dos Clientes
João é o cliente número 100
Manuel é o cliente número 101
Antunes é o cliente número 102
Clara é o cliente número 103

      - Expressões Aritméticas (**+**, **-**, ***, **/**)
        - Exemplo

SELECT NCliente, Saldo AS "Saldo Euros",
saldo*200.482 AS "Saldo Esc"
FROM TConta;

NCliente	Saldo Euros	Saldo Esc
100	1000	200482
102	25000	5012050
102	500	100241
103	4000	801928

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Consultas simples
    - Selecção ou Restrição (Cláusula **WHERE**)
      - Sintaxe

        WHERE expressão
        [AND | OR expressão
        [...]]

      - Exemplo
        - Mostrar informação dos clientes de Leiria

TCliente

NCliente	Nome	Morada	Telefone	Profissao	Idade
100	João	Batalha	123456	Electricista	55
101	Manuel	Leiria		Carpinteiro	34
102	Antunes	Coimbra	345678	Arquitecto	29
103	Clara	Leiria	567890	Médica	40

```
SELECT *
FROM TCliente
WHERE morada = 'Leiria';
```

Comparação de caracteres é sensível a maiúsculas/minúsculas

NCliente	Nome	Morada	Telefone	Profissao	Idade
101	Manuel	Leiria		Carpinteiro	34
103	Clara	Leiria	567890	Médica	40

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Consultas simples
    - Selecção ou Restrição (Cláusula **WHERE**)
      - Operadores de comparação
        - = igual a
        - > maior que
        - >= maior ou igual a
        - < menor que
        - <= menor ou igual a
        - <> , != diferente
      - Operadores lógicos
        - NOT Negação
        - AND E lógico
        - OR OU lógico
      - Exemplo
        - Mostrar todos os clientes com menos de 40 anos que morem em Leiria

```
SELECT *
FROM TCliente
WHERE morada = 'Leiria'
AND idade < 40;
```

NCliente	Nome	Morada	Telefone	Profissao	Idade
101	Manuel	Leiria		Carpinteiro	34

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Consultas simples
    - Selecção ou Restrição (Cláusula **WHERE**)
      - Outros operadores
        - **BETWEEN** x **AND** y
        - **IN**
      - Exemplo
        - Mostrar todos os clientes com idade entre 25 e 30 anos

TCliente

NCliente	Nome	Morada	Telefone	Profissao	Idade
100	João	Batalha	123456	Electricista	55
101	Manuel	Leiria		Carpinteiro	34
102	Antunes	Coimbra	345678	Arquitecto	29
103	Clara	Leiria	567890	Médica	40

```
SELECT *
FROM TCliente
WHERE idade >= 25
AND idade <= 30;

SELECT *
FROM TCliente
WHERE idade BETWEEN 25 AND 30;

SELECT *
FROM TCliente
WHERE idade IN (25, 26, 27, 28, 29, 30);
```


DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Consultas simples
    - Selecção ou Restrição (Cláusula **WHERE**)
      - Outros operadores
        - **LIKE**
          - Caracteres de substituição
            - vários caracteres **%** ou *****
            - um caracter **_** ou **?**
        - **IS NULL**
      - Exemplos
        - Mostrar todos os clientes cujo nome começa por A

```
SELECT *
FROM TCliente
WHERE nome LIKE 'A%';
```

        - Mostrar todos os clientes que não têm telefone

```
SELECT *
FROM TCliente
WHERE telefone IS NULL;
```

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Consultas simples
    - Selecção ou Restrição (Cláusula **WHERE**)
      - Predicados de negação
        - **<>**
        - **NOT**
        - **NOT** BETWEEN x AND y
        - **NOT** IN
        - **NOT** LIKE
        - IS **NOT** NULL
      - Exemplos
        - Mostrar todos os clientes que têm telefone

```
SELECT *
FROM TCliente
WHERE telefone IS NOT NULL;
```

        - Mostrar todos os clientes que não moram em Leiria

```
SELECT *
FROM TCliente
WHERE morada <> 'Leiria';

SELECT *
FROM TCliente
WHERE NOT (morada = 'Leiria');
```


DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Consultas simples
    - Selecção ou Restrição (Cláusula **WHERE**)
      - Exemplos
        - Mostrar todos os clientes que não são trintões

```
SELECT *
FROM TCliente
WHERE NOT (idade >= 30
AND idade <40);

SELECT *
FROM TCliente
WHERE idade NOT BETWEEN 30 AND 39;

SELECT *
FROM TCliente
WHERE idade NOT IN (30, 31, 32, 33, 34, 35, 36,
37, 38, 39);
```

        - Mostrar todos os clientes cujo nome não começa por A

```
SELECT *
FROM TCliente
WHERE nome NOT LIKE 'A%';
```

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Consultas simples
    - Restrição + Projecção
      - Exemplo
        - Mostrar n.º, nome, profissão e idade dos clientes que moram em Leiria

TCliente

NCliente	Nome	Morada	Telefone	Profissao	Idade
100	João	Batalha	123456	Electricista	55
101	Manuel	Leiria		Carpinteiro	34
102	Antunes	Coimbra	345678	Arquitecto	29
103	Clara	Leiria	567890	Médica	40

```
SELECT NCliente, Nome, Profissao, Idade
FROM TCliente
WHERE morada = 'Leiria';
```

NCliente	Nome	Profissao	Idade
101	Manuel	Carpinteiro	34
103	Clara	Médica	40

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Ordenação de resultados
    - Cláusula **ORDER BY**
      - Sintaxe
        - ORDER BY col1 ASC|DESC [, col2 ASC|DESC, ...]
      - Exemplo
        - Mostrar n.º, nome, profissão e idade dos clientes que moram em Leiria, ordenados por ordem alfabética do nome

```
SELECT NCliente, Nome, Profissao, Idade
FROM TCliente
WHERE morada = 'Leiria'
ORDER BY nome ASC;
```

NCliente	Nome	Profissao	Idade
103	Clara	Médica	40
101	Manuel	Carpinteiro	34

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Ordenação de resultados
    - Cláusula **ORDER BY**
      - Exemplo
        - Mostrar todas as contas ordenadas por ordem descendente do saldo e ascendente do n.º de cliente

TConta

NConta	TipoC	Saldo	CodAgencia	NCliente
10050	F	1.000	250	100
22220	A	25.000	200	102
30505	B	1.000	240	102
22555	B	4.000	200	103

```
SELECT *
FROM TConta
ORDER BY saldo DESC, NCliente;
```

NConta	TipoC	Saldo	CodAgencia	NCliente
22220	A	25.000	200	102
22555	B	4.000	200	103
30505	B	1.000	240	102
10050	F	1.000	250	100

NConta	TipoC	Saldo	CodAgencia	NCliente
22220	A	25.000	200	102
22555	B	4.000	200	103
10050	F	1.000	250	100
30505	B	1.000	240	102

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Funções de agregação
    - Operam sobre um conjunto de linhas de uma só coluna
    - Produzem como resultado um valor
    - Só é permitida a sua utilização nas cláusulas SELECT e HAVING
    - Funções que se utilizam em colunas numéricas e não numéricas
      - COUNT
      - MIN
      - MAX
    - Funções que só se utilizam em colunas numéricas
      - SUM
      - AVG
      - STDDEV
      - VARIANCE

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Funções de agregação
    - COUNT
      - Sintaxe
        - COUNT([DISTINCT | ALL] nome_coluna | *)
      - Exemplos
        - Mostrar o n.º de clientes do banco

          SELECT **COUNT**(*)
          FROM TCliente;

        - Mostrar o n.º de contas com saldo superior a 500 Euros

          SELECT **COUNT**(*)
          FROM TConta
          WHERE saldo > 500;

        - Mostrar o n.º de clientes que têm telefone

          SELECT **COUNT**(telefone)
          FROM TCliente;

        - Mostrar o n.º de clientes que contraíram um empréstimo ao banco

          SELECT **COUNT**(DISTINCT NCliente)
          FROM TEmprestimo;

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Funções de agregação
    - Sintaxe
      - MIN([DISTINCT | ALL] nome_coluna)
      - MAX([DISTINCT | ALL] nome_coluna)
      - SUM([DISTINCT | ALL] nome_coluna)
      - AVG([DISTINCT | ALL] nome_coluna)
      - STDDEV([DISTINCT | ALL] nome_coluna)
      - VARIANCE([DISTINCT | ALL] nome_coluna)
    - Exemplos
      - Mostrar o valor do saldo mais baixo do banco

        SELECT **MIN**(saldo)
        FROM TConta;

      - Mostrar o valor de empréstimo mais elevado do banco

        SELECT **MAX**(valor)
        FROM TEmprestimo;

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Funções de agregação
    - Exemplos
      - Mostrar o valor total creditado no banco

```
SELECT SUM(saldo)
FROM TConta;
```

      - Mostrar o salto total (Euros e Esc) de todas as contas do cliente 102

```
SELECT SUM(saldo) Total_Euros,
       SUM(saldo*200,482) Total_Escudos
FROM TConta
WHERE NCliente = 102;
```

      - Mostrar o valor médio dos empréstimos feitos ao banco

```
SELECT AVG(valor)
FROM TEmprestimo;
```

      - Mostrar o valor médio de todos os empréstimos efectuados na agência 200 do banco

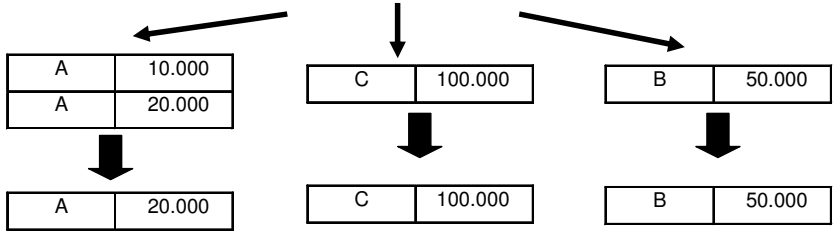
```
SELECT AVG(valor)
FROM TEmprestimo
WHERE CodAgencia = 200;
```


DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Grupos
    - Cláusula **GROUP BY**
      - Permite a divisão da tabela em grupos de linhas
      - Exemplo
        - Mostrar o valor máximo para cada tipo de empréstimos contraídos no banco

TEmprestimo

NEmp	TipoE	Valor	CodAgencia	NCliente
1000	A	10.000	250	100
1001	A	20.000	200	101
1002	C	100.000	200	102
1003	B	50.000	200	103



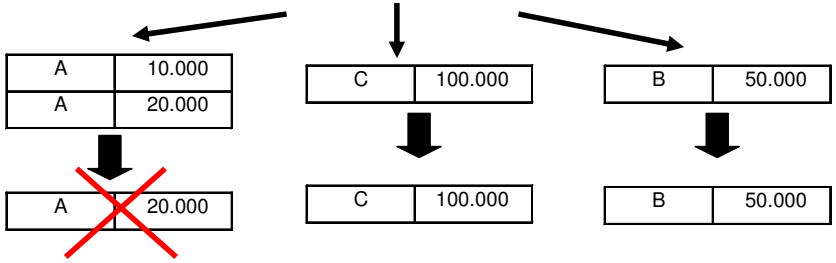

```
SELECT TipoE, MAX(valor)
FROM TEmprestimo
GROUP BY TipoE;
```

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Grupos
    - Cláusula **HAVING**
      - Permite aplicar restrição aos grupos criados
      - Exemplo
        - Mostrar o valor máximo para cada tipo de empréstimo cujo valor máximo é superior a 25.000 Euros

TEmprestimo

NEmp	TipoE	Valor	CodAgencia	NCliente
1000	A	10.000	250	100
1001	A	20.000	200	101
1002	C	100.000	200	102
1003	B	50.000	200	103




```
SELECT TipoE, MAX(valor)
FROM TEmprestimo
GROUP BY TipoE
HAVING MAX(valor) > 25000;
```


DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Grupos
    - Sintaxe

```
SELECT ... FROM ...
GROUP BY nome_coluna1 [, nome_coluna2, ...]
[HAVING expressão
[AND | OR expressão
[...]]];
```

    - Funcionamento interno
      - **1.º** Selecção das linhas validadas pela cláusula WHERE
      - **2.º** Divisão das linhas validadas nos grupos definidos na cláusula GROUP BY
      - **3.º** Aplicação das funções de agregação a cada grupo
      - **4.º** Aplicação das expressões definidas na cláusula HAVING às linhas resultantes dos passo anterior
    - Na cláusula SELECT só podem aparecer funções de agregação ou colunas que na cláusula GROUP BY definam agrupamentos

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Grupos
    - Exemplos
      - Mostrar o n.º de empréstimos concedidos pelo banco

```
SELECT COUNT(*)
FROM TEmprestimo;
```

      - Mostrar o n.º de clientes que o banco possui em cada localidade

```
SELECT morada, COUNT(*)
FROM TCliente
GROUP BY morada;
```

      - Mostrar o n.º de clientes que o banco possui em cada localidade, mas só para as localidades com mais de 1000 clientes

```
SELECT morada, COUNT(*)
FROM TCliente
GROUP BY morada
HAVING COUNT(*) > 1000;
```

      - Mostre as agências do banco que têm pelo menos 1000 contas de cada tipo. Exclua a agência com código 1.

```
SELECT CodAgencia, TipoC, COUNT(*)
FROM TConta
WHERE CodAgencia <> 1
GROUP BY CodAgencia, TipoC
HAVING COUNT(*) >= 1000;
```


DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Junção
    - Sintaxe

```
SELECT ...
FROM tabela1, tabela2 [, tabela3, ...]
WHERE condiçãoJunção1
[AND condiçãoJunção2 ...];
```

    - Junção com N tabelas necessita de pelo menos N-1 condições de junção
    - Utilização de pseudónimos de tabela

```
FROM NomeTabela1 [AS] tab1, nomeTabela2 [AS] tab2
```

      - Exemplo
        - Mostrar a informação dos clientes do banco e respectivas contas

```
SELECT TCliente.*, Tconta.*
FROM TCliente, TConta
WHERE TCliente.NCliente = TConta.NCliente;

SELECT tcli.*, tc.*
FROM TCliente AS tcli, TConta AS tc
WHERE tcli.NCliente = tc.NCliente;
```

DML
Linguagem de Manipulação de Dados

Comando SELECT

- Junção
  - Exemplo
    - Mostrar a informação dos clientes do banco e respectivas contas

```
SELECT tcli.*, tc.*
FROM TCliente tcli, TConta tc
WHERE tcli.NCliente = tc.NCliente;
```

TConta

NConta	TipoC	Saldo	CodAgencia	NCliente
10050	F	1.000	250	100
22220	A	25.000	200	102
30505	B	1.000	240	102
22555	B	4.000	200	103

TCliente

NCliente	Nome	Morada	Telefone	Profissao	Idade
100	João	Batalha	123456	Electricista	55
101	Manuel	Leiria		Carpinteiro	34
102	Antunes	Coimbra	345678	Arquitecto	29
103	Clara	Leiria	567890	Médica	40

NCliente	Nome	Morada	Telefone	Profissao	Idade	NConta	TipoC	Saldo	CodAgencia	NCliente
100	João	Batalha	123456	Electricista	55	10050	F	1000	250	100
102	Antunes	Coimbra	345678	Arquitecto	29	22220	A	25000	200	102
102	Antunes	Coimbra	345678	Arquitecto	29	30505	B	1000	240	102
103	Clara	Leiria	567890	Médica	40	22555	B	4000	200	103

SQL
Linguagem de Consulta Estruturada

DML
Linguagem de Manipulação de Dados

Comando SELECT

- Junção
  - Exemplo
    - Mostrar a informação dos clientes do banco que possuem contas em agências situadas na localidade onde residem

```
SELECT tcli.*
FROM TCliente tcli, TConta tc, TAgencia ta
WHERE tcli.NCliente = tc.NCliente
AND tc.CodAgencia = ta.CodAgencia
AND tcli.Morada = ta.Localidade;
```

NCliente	Nome	Morada	Telefone	Profissao	Idade
103	Clara	Leiria	567890	Médica	40

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Junção
    - Tipos
      - Equi-Junção
        - Condição de junção utiliza o operador de igualdade
      - Não-Equi-Junção
        - Condição de junção não utiliza o operador de igualdade
      - Utilizando a mesma tabela
        - Atribuição de pseudónimos

          FROM nome_tabela [AS] tab1, nome_tabela [AS] tab2
      - Exemplo
        - Mostrar todos os clientes que tem idade superior à do cliente n.º 101

          SELECT c1.*
          FROM TCliente c1, TCliente c2
          WHERE c2.NCliente = 101
          AND c1.idade > c2.idade;

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Junção
    - Tipos
      - Externa
        - Permite conter linhas que não satisfazem a condição de junção
        - Operadores

          LEFT JOIN
          RIGHT JOIN
          FULL JOIN
        - Sintaxe

          FROM nome_tabela1 **LEFT JOIN** | **RIGHT JOIN** | **FULL JOIN**
          nome_tabela2 **ON** condição_junção

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Junção
    - Tipos
      - Exemplo
        - Mostrar o nome de todos os clientes que existem no banco e contas que eventualmente tenham aberto

```
SELECT c.*, cli.Nome
FROM TConta c RIGHT JOIN TCliente cli
     ON c.NCliente = cli.NCliente;
```

```
SELECT c.*, cli.Nome
FROM TConta c, TCliente cli
WHERE c.NCliente = cli.NCliente (+);
```

NConta	TipoC	Saldo	CodAgencia	NCliente	Nome
10050	F	1.000	250	100	João
22220	A	25.000	200	102	Antunes
30505	B	1.000	240	102	Antunes
22555	B	4.000	200	103	Clara
NULL	NULL	NULL	NULL	101	Manuel

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Junção vertical
    - Utilização dos operadores de conjunto entre consultas
    - Regras
      - As consultas têm de ter o mesmo n.º de colunas na cláusula SELECT
      - As colunas têm de ser do mesmo tipo segundo a ordem pelas quais são definidas
    - Sintaxe da UNIÃO

```
SELECT ...
UNION [ALL]
SELECT ... ;
```

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Junção vertical
    - UNIÃO
      - Exemplo
        - Mostrar o n.º e tipo de todas as contas e o n.º e tipo de todos os empréstimos existentes no banco

```
SELECT NConta AS Num, TipoC AS Tipo
FROM  TConta
UNION ALL
SELECT NEmp, TipoE
FROM  TEmprestimo;
```

Num	Tipo
10050	F
22220	A
30505	B
22555	B
1000	A
1001	A
1002	C
1003	B

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Junção vertical
    - UNIÃO
      - Exemplo
        - Mostrar o n.º, nome e morada de todos os clientes do banco que tenham idade igual ou superior a 50 ou idade inferior a 30

```
SELECT NCliente, Nome, Morada
FROM TCliente
WHERE idade >= 50
UNION
SELECT NCliente, Nome, Morada
FROM TCliente
WHERE idade < 30;
```

O mesmo que

```
SELECT NCliente, Nome, Morada
FROM TCliente
WHERE idade >= 50
OR idade < 30;
```

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Junção vertical
    - Sintaxe da INTERSECÇÃO

```
SELECT ...
INTERSECT
SELECT ... ;
```

- Este operador não é suportado por todos os sistemas
- Exemplo
  - Mostrar todos os clientes que são simultaneamente devedores e credores da agência 250

```
SELECT TCliente.*
FROM  TCliente, TConta
WHERE TCliente.NCliente = TConta.NCliente
AND TConta.CodAgencia = 250
INTERSECT
SELECT TCliente.*
FROM TCliente, TEmprestimo
WHERE TCliente.NCliente = TEmprestimo.NCliente
AND TEmprestimo.CodAgencia = 250;
```


DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Junção vertical
    - Sintaxe da DIFERENÇA

```
SELECT ...
MINUS | EXCEPT
SELECT ... ;
```

- Este operador não é suportado por todos os sistemas
- Exemplo
  - Mostrar todos os clientes do banco que possuem empréstimos mas não possuem contas

```
SELECT TCliente.*
FROM  TCliente, TEmprestimo
WHERE TCliente.NCliente = TEmprestimo.NCliente
EXCEPT
SELECT TCliente.*
FROM TCliente, TConta
WHERE TCliente.NCliente = TConta.NCliente;
```

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Junção vertical
    - Cláusula **ORDER BY**
      - Sintaxe

        ORDER BY NumColuna1 ASC|DESC
        [, NumColuna2 ASC|DESC, ...]

      - Só se utiliza na última consulta
      - As colunas são referenciadas pela posição que ocupam
      - Exemplo
        - Mostrar todos as contas e todos os empréstimos do banco

```
SELECT NConta, TipoC, Saldo
FROM  TConta
UNION ALL
SELECT NEmp, TipoE, Valor
FROM  TEmprestimo
ORDER BY 3 DESC,  2;
```


DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Subconsultas
    - Comando SELECT incluído em cláusulas de outro comando SELECT
    - Introduzidas nas cláusulas WHERE e/ou HAVING após um operador relacional
    - N.º de colunas projectadas na subconsulta é igual ao n.º de colunas definidas na expressão de comparação
    - Cláusula ORDER BY só é utilizada na consulta principal
    - Funcionamento interno

      1.º Execução do comando SELECT interno

      2.º Utilização do resultado obtido na execução do comando SELECT externo

DML
Linguagem de Manipulação de Dados

Comando SELECT

- Subconsultas
  - Exemplo
    - Mostrar por ordem alfabeticamente do nome, todos os clientes com idade superior à do cliente n.º 101

```
SELECT *
FROM TCliente
WHERE idade > (SELECT idade
               FROM TCliente
               WHERE NCliente = 101)
ORDER BY nome;
```

RESOLUÇÃO

1.º SELECT idade
FROM TCliente
WHERE NCliente = 101

Idade
34

2.º SELECT *
FROM TCliente
WHERE idade > **34**
ORDER BY nome;

NCliente	Nome	Morada	Telefone	Profissao	Idade
103	Clara	Leiria	567890	Médica	40
100	João	Batalha	123456	Electricista	55

DML
Linguagem de Manipulação de Dados

Comando SELECT

- Subconsultas
  - Tipo
    - Escalares
      - Mostrar o(s) cliente(s) mais novo(s) do banco

```
SELECT *
FROM TCliente
WHERE idade = (SELECT MIN(idade)
               FROM TCliente);
```

    - Linha
      - Mostrar o(s) cliente(s) cujas contas têm o menor saldo em cada agência do banco

```
SELECT *
FROM TCliente
WHERE NCliente IN
        (SELECT NCliente
         FROM TConta
         WHERE (CodAgencia, Saldo) IN
                (SELECT CodAgencia, MIN(Saldo)
                 FROM TConta
                 GROUP BY CodAgencia))
ORDER BY Nome;
```

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Subconsultas
    - Tipo
      - Tabela
        - SOME | ANY e ALL
          - Utilizam-se conjuntamente com os operadores de comparação < e >

            >ALL
            >ANY
            <ALL
            <ANY
        - IN
        - NOT IN

        - Mostrar o n.º do(s) cliente(s) cujas contas têm um saldo inferior aos valores dos empréstimo mais baixos de cada agência do banco

          ```
          SELECT NCliente
          FROM TConta
          WHERE saldo <ANY (SELECT MIN(valor)
                            FROM TEmprestimo
                            GROUP BY CodAgencia);
          ```


DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Subconsultas Correlacionadas
    - Subconsultas encadeadas
    - Executadas uma vez para cada "linha candidata" da consulta principal
    - Utilizam os valores da consulta principal
    - Funcionamento interno

      1.º Obtenção de uma "linha candidata" (consulta externa)

      2.º Execução da subconsulta utilizando a "linha candidata" obtida

      3.º Utilização de valores resultantes da subconsulta para qualificar ou desqualificar a "linha candidata"

      4.º Repetição dos passos anteriores até não restarem mais "linhas candidatas"

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Subconsultas Correlacionadas
    - Exemplos
      - Mostrar a informação dos clientes do banco que possuem contas em agências situadas na localidade onde residem

```
SELECT tcli.*
FROM TCliente tcli
WHERE tcli.morada IN
                    (SELECT DISTINCT Localidade
                     FROM TConta tc, TAgencia ta
                     WHERE tc.CodAgencia = ta.CodAgencia
                     AND NCliente = tcli.NCliente);
```

      - Mostrar o(s) cliente(s) cujas contas têm o menor saldo em cada agência do banco

```
SELECT *
FROM TCliente
WHERE NCliente IN
          (SELECT NCliente
           FROM TConta tc
           WHERE Saldo =
                    (SELECT min(SALDO)
                     FROM TConta
                     WHERE  CodAgencia = tc.CodAgencia))
ORDER BY Nome;
```


DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Subconsultas
    - Operador **EXISTS**
      - Sintaxe

        WHERE [NOT] **EXISTS** (Subconsulta)

      - Exemplo
        - Mostrar todos os clientes que são simultaneamente devedores e credores da agência 250

```
SELECT tc.*
FROM  TCliente tc
WHERE EXISTS (SELECT *
                    FROM TConta
                    WHERE NCliente = tc.NCliente
                    AND CodAgencia = 250)
AND EXISTS  (SELECT 1
                    FROM TEmprestimo
                    WHERE NCliente = tc.NCliente
                    AND CodAgencia = 250);
```

```
SELECT TCliente.*
FROM  TCliente, TConta
WHERE TCliente.NCliente = TConta.NCliente
AND TConta.CodAgencia = 250
INTERSECT
SELECT TCliente.*
FROM TCliente, TEmprestimo
WHERE TCliente.NCliente = TEmprestimo.NCliente
AND TEmprestimo.CodAgencia = 250;
```

DML
Linguagem de Manipulação de Dados

- Comando SELECT
  - Subconsultas
    - Operador **EXISTS**
      - Exemplo
        - Mostrar todos os clientes do banco que possuem empréstimos mas não possuem contas

```
SELECT TCliente.*
FROM  TCliente
WHERE EXISTS
            (SELECT 1
             FROM TEmprestimo
             WHERE NCliente = TCliente.NCliente)
AND NOT EXISTS
            (SELECT *
             FROM TConta
             WHERE NCliente = TCliente.NCliente);
```

```
SELECT TCliente.*
FROM TCliente, TEmprestimo
WHERE TCliente.NCliente = TEmprestimo.Ncliente
EXCEPT
SELECT TCliente.*
FROM  TCliente, TConta
WHERE TCliente.NCliente = TConta. NCliente
```


DML
Linguagem de Manipulação de Dados

- Comando INSERT
  - Permite inserir uma nova linha de informação numa tabela
    - Sintaxe

```
INSERT INTO  nome_tabela [lista de colunas a inserir]
VALUES  (valor1, valor2, ...);
```

    - Exemplos

```
INSERT INTO TCliente
VALUES (104, 'António', 'Lisboa', NULL, 'Enfermeiro', 32);

INSERT INTO TCliente (NCliente, Nome, Morada)
VALUES (105, 'José Silva', 'Coimbra');
```

DML
Linguagem de Manipulação de Dados

- Comando INSERT
  - Permite copiar várias linhas de uma tabela para outra tabela
    - Sintaxe

```
INSERT INTO  nome_tabela   [lista de colunas a inserir]
      SELECT  ...;
```

    - Exemplos
      - Considere a estrutura da tabela TClienteMau

TClienteMau

NCliente	Nome	SaldoTotal

      - Inserir na tabela TClienteMau o n.º e nome dos clientes, bem como o valor total referente ao saldo de todas as suas contas, para clientes com saldo total negativo

```
INSERT INTO TClienteMau
   SELECT tc.NCliente, Nome, SUM(tc.saldo) SaldoTotal
   FROM TCliente tcli, TConta tc
   WHERE tcli.NCliente = tc.NCliente
   GROUP BY tc.NCliente, Nome
   HAVING SUM(tc.Saldo) < 0;
```


DML
Linguagem de Manipulação de Dados

- Comando UPDATE
  - Permite alterar todas as linhas de uma tabela
    - Sintaxe

```
UPDATE  nome_tabela
SET  colA = valor1 | expressao, ...;
```

    - Exemplos

```
UPDATE TConta
SET CodAgencia = CodAgencia + 1000;
```

```
UPDATE TEmprestimo
SET CodAgencia = CodAgencia + 1000,
     Valor = Valor * 0.95;
```

DML
Linguagem de Manipulação de Dados

- Comando UPDATE
  - Permite alterar apenas as linhas que satisfazem determinada condição
    - Sintaxe

```
UPDATE  nome_tabela
SET  colA = valor1 | expressao, ...
WHERE  expressão_condicional;
```

    - Exemplos
      - Aplique uma penalização de 2% a todas as contas com saldo negativo

```
UPDATE TConta
SET Saldo = Saldo * 0.98
WHERE Saldo < 0;
```

      - Atribua um bónus de 10% a todas as contas do cliente mais antigo do banco com saldo superior a 50000 Euros.

```
UPDATE TConta
SET Saldo = Saldo * 1.10
WHERE Ncliente = (SELECT MIN(NCliente)
                  FROM TConta
                  WHERE saldo > 50000);
```


DML
Linguagem de Manipulação de Dados

- Comando DELETE
  - Permite eliminar todas as linhas de uma tabela
    - Sintaxe

```
DELETE FROM nome_tabela;
```

    - Exemplo

```
DELETE FROM TEmprestimo;
```

  - Permite eliminar apenas as linhas de uma tabela que satisfazem determinada condição
    - Sintaxe

```
DELETE FROM nome_tabela
WHERE expressão_condicional;
```

    - Exemplo
      - Eliminar todas as contas do cliente 122

```
DELETE FROM TConta
WHERE NCliente=122;
```

      - Eliminar todas as contas dos clientes cujo saldo total das suas contas é negativo

```
DELETE FROM TConta
WHERE NCliente IN (SELECT NCliente
                   FROM TConta
                   GROUP BY NCliente
                   HAVING SUM(Saldo) <0);
```

DDL
Linguagem de Definição de Dados

- CREATE SCHEMA
- DROP SCHEMA
- CREATE DOMAIN
- ALTER DOMAIN
- DROP DOMAIN
- CREATE TABLE
- CREATE TABLE **AS**
- ALTER TABLE
- DROP TABLE
- CREATE ASSERTION
- DROP ASSERTION
- CREATE VIEW
- DROP VIEW
- CREATE INDEX
- DROP INDEX

DDL
Linguagem de Definição de Dados

- Comando CREATE SCHEMA
  - Permite criar esquemas de bases de dados onde se definem os metadados de um projecto
  - Sintaxe

    CREATE SCHEMA nome
    [AUTORIZATION nome_utilizador];
  - Exemplo

    CREATE SCHEMA GereBanco
    AUTORIZATION userABC;
  - Nalguns SGBD foi substituído pelo comando
    - CREATE DATABASE
  - Elementos de um esquema de Base de Dados
    - Tabelas
    - Domínios
    - Vistas
    - Privilégios
    - *Assertions*
    - Índices
    - Sequências

DDL
Linguagem de Definição de Dados

- Comando DROP SCHEMA
  - Permite eliminar um esquema de base de dados
  - Sintaxe

    DROP SCHEMA nome_esquema [RESTRICT | CASCADE];

  - Exemplo

    DROP SCHEMA GereBanco;

- Comando CREATE DOMAIN
  - Permite definir um tipo de dados
  - Sintaxe

    CREATE DOMAIN nome tipo_dados
    [DEFAULT valor_omissão]
    [CHECK (condição)];

  - Exemplo

    CREATE DOMAIN tipo_morada VARCHAR(30);

    CREATE DOMAIN tipo_sexo CHAR
    CHECK (VALUE IN ('M', 'F'));

DDL
Linguagem de Definição de Dados

- Comando ALTER DOMAIN
  - Permite alterar um domínio
  - Sintaxe

    ALTER DOMAIN nome_dominio novo_tipo_dados
    [DEFAULT valor_omissão]
    [CHECK (condição];

  - Exemplo

    ALTER DOMAIN tipo_morada VARCHAR(35);

- Comando DROP DOMAIN
  - Permite eliminar um domínio
  - Sintaxe

    DROP DOMAIN nome_dominio [RESTRICT | CASCADE];

  - Exemplo

    DROP DOMAIN tipo_morada CASCADE;

DDL
Linguagem de Definição de Dados

- Comando CREATE TABLE
  - Permite criar uma tabela com todas as suas colunas e algumas restrições
  - Sintaxe para a definição de colunas

```
CREATE TABLE nome_tabela(
    coluna1  tipo_dados[(tamanho)],
    ...,
    colunaN  tipo_dados[(tamanho)],
    );
```

  - Exemplo

```
CREATE TABLE TCliente (
    NCliente        INTEGER,
    Nome            VARCHAR(30),
    Morada          tipo_morada,
    Telefone        VARCHAR(9),
    Profissao       VARCHAR(15),
    Idade           SMALLINT);
```


DDL
Linguagem de Definição de Dados

- Comando CREATE TABLE
  - Restrições de Integridade
    - Domínio
      - DEFAULT
      - CHECK
    - Entidade
      - NOT NULL
      - UNIQUE
      - PRIMARY KEY
    - Referencial
      - CHECK
      - FOREIGN KEY
        - Sintaxe

          FOREIGN KEY (lista_colunas)
          REFERENCES NomeTabela(lista_colunas) [propriedades]

        - Propriedades ON UPDATE ou ON DELETE

          SET NULL
          SET DEFAULT
          CASCADE
          RESTRICT
          NO ACTION

DDL
Linguagem de Definição de Dados

- Comando CREATE TABLE
  - Exemplos

```
CREATE TABLE TCliente (
    NCliente    INTEGER PRIMARY KEY,
    Nome        VARCHAR(30) NOT NULL,
    Morada      tipo_morada DEFAULT = 'Leiria',
    Telefone    VARCHAR(9),
    Profissao   VARCHAR(15),
    Idade       SMALLINT,
    CHECK (idade BETWEEN 16 AND 110));

CREATE TABLE TCliente (
    NCliente    INTEGER,
    PRIMARY KEY(NCliente),
    Nome        VARCHAR(30) NOT NULL,
    UNIQUE(nome),
    Morada      tipo_morada,
    DEFAULT (Morada) = 'Leiria',
    Telefone    VARCHAR(9),
    Profissao   VARCHAR(15),
    Idade       SMALLINT,
    CHECK (idade BETWEEN 16 AND 110));
```


DDL
Linguagem de Definição de Dados

- Comando CREATE TABLE
  - Exemplos

```
CREATE TABLE TCliente (
    NCliente    INTEGER,
    Nome        VARCHAR(30) NOT NULL,
    Morada      tipo_morada DEFAULT = 'Leiria',
    Telefone    VARCHAR(9),
    Profissao   VARCHAR(15),
    Idade       SMALLINT,
    CONSTRAINT PK_Cliente PRIMARY KEY(NCliente),
    CONSTRAINT UN_Cliente_Nome UNIQUE(Nome),
    CONSTRAINT CK_Cliente_Idade
            CHECK (idade BETWEEN 16 AND 110));

CREATE TABLE TConta (
    NConta      INTEGER,
    ...
    CodAgencia INTEGER,
    NCliente    INTEGER,
    CONSTRAINT PK_Conta PRIMARY KEY(NConta),
    CONSTRAINT FK_Conta_Agencia FOREIGN KEY(CodAgencia)
                    REFERENCES TAgencia(CodAgencia)
                    ON UPDATE CASCADE
                    ON DELETE NO ACTION),
    CONSTRAINT FK_Conta_Cliente
                    FOREIGN KEY (NCliente)
                    REFERENCES TCliente(NCliente));
```

DDL
Linguagem de Definição de Dados

- Comando CREATE TABLE
  - Permite criar uma tabela através de dados existentes noutra(s) tabela(s)
  - Sintaxe

```
CREATE TABLE nome [lista_colunas]
        AS (Subconsulta);
```

  - Exemplos

```
CREATE TABLE TClienteXXL
AS
(SELECT *
 FROM TCliente
 WHERE NCliente IN (SELECT c.NCliente
                    FROM TConta c
                    GROUP BY c.NCliente
                    HAVING SUM(saldo) > 500000));
```

- Comando DROP TABLE
  - Permite eliminar uma tabela
  - Sintaxe

```
DROP TABLE nome_tabela [RESTRICT | CASCADE];
```

  - Exemplo

```
DROP TABLE TClienteXXL CASCADE;
```


DDL
Linguagem de Definição de Dados

- Comando ALTER TABLE
  - Permite alterar a estrutura e as restrições associadas a uma tabela
  - Sintaxe

```
ALTER TABLE nome_tabela(
    [ADD COLUMN nome_coluna tipo_dados restrições]
    [ALTER COLUMN nome_coluna novo_tipo novas_restrições]
    [DROP COLUMN nome_coluna]
    [ADD CONSTRAINT nome_restrição  restrição]
    [ALTER CONSTRAINT nome_restrição  nova_restrição]
    [DROP CONSTRAINT nome_restrição ];
```

  - Exemplo

```
ALTER TABLE TCliente
    ADD COLUMN  TipoCliente  CHAR(1)  NOT NULL;
```

DDL
Linguagem de Definição de Dados

- Comando CREATE ASSERTION
  - Permite garantir a integridade referencial
  - Sintaxe

    CREATE ASSERTION nome [CHECK condições];

  - Exemplo

```
CREATE ASSERTION Verifica_Quant_Emprestimos
   CHECK (NOT EXISTS
                (SELECT NCliente
                 FROM TEmprestimo e
                 GROUP BY e.NCliente
                 HAVING COUNT(*) >= 5));
```

- Comando DROP ASSERTION
  - Sintaxe

    DROP ASSERTION nome [RESTRICT | CASCADE];

  - Exemplo

```
DROP ASSERTION Verifica_Quant_Emprestimos;
```


DDL
Linguagem de Definição de Dados

- Comando CREATE VIEW
  - Permite criar uma tabela lógica (vista)
  - Sintaxe

    CREATE VIEW nome [lista_colunas]
    AS (Subconsulta)
    [WITH CHECK OPTION];

  - Exemplo

```
CREATE VIEW  V_contas500_1
AS (SELECT *
        FROM TConta
        WHERE saldo > 500000);
```

✓
```
INSERT INTO V_Contas500_1
VALUES (50505, 'B', 300, 240, 103);
```

✓
```
INSERT INTO V_Contas500_1
VALUES (50506, 'B', 500100, 240, 100);
```

DDL
Linguagem de Definição de Dados

- Comando CREATE VIEW
  - Exemplo

```
CREATE VIEW  V_contas500_2
AS (SELECT *
        FROM TConta
        WHERE saldo > 500000)
WITH CHECK OPTION;
```

✘
```
INSERT INTO V_Contas500_2
VALUES (50505, 'B', 300, 240, 103);
```

✓
```
INSERT INTO V_Contas500_2
VALUES (50506, 'B', 500100, 240, 100);
```

- Comando DROP VIEW
  - Sintaxe

    DROP VIEW nome [RESTRICT | CASCADE];

  - Exemplo

    DROP VIEW V_Contas500_1;

DDL
Linguagem de Definição de Dados

- Comando CREATE INDEX
  - Permite criar índices
  - Sintaxe

    CREATE [UNIQUE] INDEX nome_índices
        ON nome_tabela (coluna1 [, coluna2, ...]);

  - Exemplo

```
CREATE UNIQUE INDEX IDX_Cliente_Agencia
    ON TEmprestimo (NCliente, CodAgencia);

CREATE INDEX IDX_NomeCliente
    ON TCliente (Nome);

CREATE UNIQUE INDEX IDX_LocalAgencia
    ON TAgencia (Localidade);
```

- Comando DROP INDEX
  - Sintaxe

    DROP INDEX nome [RESTRICT | CASCADE];

  - Exemplo

    DROP INDEX IDX_LocalAgencia;

DCL
Linguagem de Controlo de Dados

- Confidencialidade
  - Privilégios
    - SELECT
    - INSERT
    - UPDATE
    - DELETE
    - *REFERENCES*
    - *USAGE*
  - GRANT
  - REVOKE
- Integridade
  - COMMIT
  - ROLLBACK

DCL
Linguagem de Controlo de Dados

- Confidencialidade
  - Comando GRANT
    - Permite atribuir privilégios aos utilizadores da BD no acesso a tabelas, vistas, ... (elementos da BD)
    - Sintaxe

```
GRANT ALL PRIVILEGES | privilégio1[(colunas)][, privilégio2, ...]
ON nome_tabela
TO PUBLIC | nome_utilizador1[, nome_utilizador2, ...]
[WITH GRANT OPTION];
```

    - Exemplos

```
GRANT SELECT, UPDATE(nome, morada, telefone)
ON TCliente
TO user_abc, user_xpto;

GRANT ALL PRIVILEGES
ON TAgencia
TO user_admin
WITH GRANT OPTION;

GRANT SELECT
ON TAgencia
TO PUBLIC;
```

DCL
Linguagem de Controlo de Dados

- Confidencialidade
  - Comando REVOKE
    - Permite retirar privilégios anteriormente concedidos aos utilizadores da BD
    - Sintaxe

      REVOKE ALL PRIVILEGES | privilégio1[(colunas)][, privilégio2, ...]
      ON nome_tabela
      FROM PUBLIC | nome_utilizador1[, nome_utilizador2, ...];

    - Exemplos

```
REVOKE SELECT, UPDATE(nome, morada, telefone)
ON TCliente
FROM user_abc, user_xpto;

REVOKE DELETE
ON TAgencia
FROM user_admin;

REVOKE ALL PRIVILEGES
ON TAgencia
FROM PUBLIC;
```


DCL
Linguagem de Controlo de Dados

- Integridade
  - Comando COMMIT
    - Permite validar as alterações efectuadas durante uma transacção
    - Sintaxe

      COMMIT;

    - Exemplo

```
UPDATE TConta
SET Saldo = Saldo - 500
WHERE NConta = 22220;

UPDATE TConta
SET Saldo = Saldo + 500
WHERE NConta = 30505;

COMMIT;
```

DCL
Linguagem de Controlo de Dados

- Integridade
  - Comando ROLLBACK
    - Permite anular todas as alterações efectuadas durante uma transacção
    - Sintaxe

      ROLLBACK;

    - Exemplos

```
UPDATE TConta
SET Saldo = Saldo - 500
WHERE NConta = 22220;

UPDATE TConta
SET Saldo = Saldo + 500
WHERE NConta = 30505;

ROLLBACK;
```