Lisbon School of Economics & Management
Universidade de Lisboa

U LISBOA
UNIVERSIDADE DE LISBOA

SQL
STRUCTURED QUERY LANGUAGE

Carlos J. Costa (2021)

Introdução

- Junho 1970: publicação por E.F. Codd, no ACM Journal, do artigo “A Relational Model of Data for Large Shared Data Banks”.
- O modelo proposto por Codd é hoje a base de trabalho para qualquer SGBD Relacional.
- A 1ª implementação comercial da linguagem **SEQUEL** foi realizada pela IBM (System R) e tinha por objectivo a implementação do modelo de Codd. A evolução desta linguagem deu origem ao **SQ**L.
- A 1ª implementação comercial de SQL foi realizada pela Relational Software Inc., hoje denominada Oracle Corporation.
- O SQL é uma **linguagem normalizad**a. Normas SQL: SQL-86, SQL-89, SQL2.
- SQL3 inclui mecanismos object-oriented (1999).

Características do SQL

- Implementa os conceitos definidos no Modelo Relacional.
- Com a linguagem SQL é possível:
  - Criar, Alterar e Remover todas as componentes de uma Base de Dados (ex: tabelas);
  - Inserir, Alterar e Apagar dados;
  - Interrogar a Base de Dados;
  - Controlar o acesso dos utilizadores à Base de Dados e as operações a que cada um deles pode ter acesso.
  - Obter a garantia da consistência e integridade dos dados.

Características do SQL

A linguagem SQL tem duas vertentes:

- LDD- Linguagem de Definição de Dados (*Data Definition Language*)
- LMD - Linguagem de Manipulação de Dados (*Data Manipulation Language*)

SQL - LDD

- Quais as suas caracteristicas.
- Domínios
- Criação de Tabelas
- Restrições de Integridade
- Alteração de Tabelas
- Eliminação de Tebelas

SQL como LDD

- O SQL como Linguagem de Definição de Dados (LDD) permite:
  - Criar (CREATE), remover (DROP) e alterar (ALTER) tabelas
  - Descrever restrições de integridade
  - Registar e remover utilizadores
  - Atribuir e retirar privilégios aos utilizadores.

Domínios

- Através da instrução CREATE DOMAIN é possível definir um domínio (tipo de dados) genérico ao qual podem ser atribuídos vários atributos.

- CREATE DOMAIN dm_morada VARCHAR(30)

Criação de Tabelas

- Através da instrução CREATE TABLE é Criação de Tabelas

CREATE TABLE <nome-tabela> (<definição de colunas e restrições de integridade)

Exemplo:

```
CREATE TABLE Empregado (
   ID INTEGER, Nome CHAR(50),
   Data_Nasc DATE,
   Salario FLOAT)
```

Restrições de Integridade

1) Restrição **NOT NULL** especifica que uma coluna não admite valores nulos
2) Restrição **PRIMARY KEY** Identifica a(s) coluna(s) que constituem a chave primária
3) Restrição **UNIQUE** Identifica uma coluna , ou um grupo de colunas, como chave candidata, i.e., o seu valor não se repete na tabela .
4) Restrição **CHECK** restrições sobre valores de colunas
5) Restrições de integridade referencial: especificar na tabela referenciadora quais os atributos importados (chaves estrangeiras) e de que tabelas são importados (tabelas referenciadas).

Restrições de Integridade

- Exemplo: definição da Tabela Funcionário:

```
CREATE TABLE Funcionario (
            cod-func INTEGER,
            Nome CHAR(50), ...
            cod-dep CHAR(3),
FOREIGN KEY (cod-dep)
            REFERENCES Departamento (cod-dep),
  PRIMARY KEY (cod-func),
ON UPDATE CASCADE,
ON DELETE SET NULL)
```

Exemplo Access

```
Create Table Cliente (
clientID int NOT NULL PRIMARY KEY,
nome string(20)
);

Create Table Vendas(
        vendasID int NOT NULL,
        data date,
        clientID int,
PRIMARY KEY (vendasID),
CONSTRAINT FK_clientID FOREIGN KEY (clientID)
        REFERENCES Cliente(clientID)
);
```

Alteração de Tabelas

- Através da instrução ALTER TABLE é possível alterar as tabelas existentes:
  - adição de novas colunas ou restrições de integridade
  - modificação das características de uma coluna
  - eliminação de colunas ou restrições de integridade existentes.

```
ALTER TABLE Funcionario
ADD COLUMN nacionalidade VARCHAR(15)
DEFAULT 'portuguesa'

ALTER TABLE Cliente
DROP COLUMN nacionalidade
```

Eliminação de Tabelas

A eliminação de uma tabela do esquema de uma Base de Dados é feita através do comando DROP TABLE.

Exemplo:

```
DROP TABLE Funcionario
```

SQL - LMD

- Caracteristicas de SQL – LMD
- Select
  - Select Where

    <condições>
  - Funções de Agregação
  - Agrupemento (*group by*)
  - Restrições sobre grupos (*having*)
  - Ordenação (*order by*)
- Insert, Delete, Update

SQL como LMD

- O SQL como Linguagem de Manipulação de Dados (LMD) possui instruções de:

- Interrogação da Base de Dados:
  - instrução SELECT

- Actualização da Base de Dados:
  - instruções INSERT, DELETE e UPDATE

SELECT

Tabela **Cliente**

Num	NIF	BI	Nome	Telefone	Morada	Cidade	Vendas
906	123644569	7	Paula	21998	Av. AAA	Lisboa	20 000
9000	100361235	6	Joana	21333	R. XPTO	Coimbra	30 000
9001	100392687	5	Ana	22555	R. XXX	Aveiro	40 000
9002	112531259	3	Pedro	23387	R. VVV	Aveiro	20 000
9003	112362268	4	José	24463	R. SSS	Porto	50 000
9004	136545568	1	Isabel	23959	R. BBB	Porto	30 000
9005	123642642	6	Leonor	21456	R. SSS	Lisboa	15 000

- Qual o nome, morada e telefone dos Clientes ?
- Como seleccionava se a tabela estivesse impressa?

SELECT

Selecção de colunas

SELECT <colunas> FROM <tabelas>

Onde:

<colunas> especifica a lista de atributos cujos valores interessa conhecer

<tabelas> especifica quais as tabelas envolvidas no processamento em questão

SELECT

Solução em SQL:

```
SELECT Nome, Morada, Telefone  FROM Cliente
```

Resultado Final:

Nome	Morada	Telefone
Paula	Av. AAA	21998
Joana	R. XPTO	21333
Ana	R. XXX	22555
Pedro	R. VVV	23387
José	R. SSS	24463
Isabel	R. BBB	23959
Leonor	R. SSS	21456

SELECT

Num	NIF	BI	Nome	Telefone	Morada	Cidade
906	12364	45697	Paula	21998	Av. AAA	Lisboa
9000	10036	12356	Joana	21333	R. XPTO	Coimbra
9001	10039	26875	Ana	22555	R. XXX	Aveiro
9002	11253	12593	Pedro	23387	R. VVV	Aveiro
9003	11236	22684	José	24463	R. SSS	Porto
9004	13654	55681	Isabel	23959	R. BBB	Porto
9005	12364	26426	Leonor	21456	R. SSS	Lisboa

Qual o nome, telefone e morada dos clientes de Lisboa ?

Como seleccionava se a tabela estivesse impressa?

SELECT /Where

SELECT ...

FROM …

WHERE Condição

- Uma condição é uma expressão que produz sempre um resultado do tipo Boolean, isto é, Verdade (TRUE) ou Falso (FALSE).
- A condição tem como finalidade filtrar os registos que se pretendem calcular, i.e., para cada linha da tabela a condição é avaliada e caso seja verdadeira o registo é seleccionado.

SELECT /Where

Selecção de registos

SELECT <colunas> FROM <tabelas>

[WHERE <condicao>]

<condicao> traduz a expressão lógica que define a condição a verificar-se para selecção dos registos

SELECT /Where

Solução em SQL

```
SELECT Nome, Telefone, Morada
FROM Cliente
WHERE Cidade= "Lisboa"
```

Resultado Final:

Nome	Telefone	Morada
Paula	21998	Av. AAA
Leonor	21456	R. SSS

Construção de Condições

- Utilizam-se operadores relacionais e/ou lógicos para impôr e combinar restrições de selecção
- O resultado da aplicação destes operadores é sempre o valor lógico TRUE ou FALSE.
- Operadores Relacionais
  - Permitem impôr restrições <Campo> vs <Valor> (=, >,<,>=,<=,<>)
- Operadores Lógicos
  - Permitem combinar restrições (AND, OR, NOT)
- Precedência dos Operadores
  - parêntesis, multiplicação e divisão, adição e subtracção, NOT, AND, OR.

Construção de Condições

Outros Operadores (BETWEEN, IN, IS)

Operador BETWEEN permite especificar intervalos de valores

SELECT ...

FROM ...

WHERE campo [NOT] BETWEEN valor1 AND valor2

Exemplo:Qual o nome e número respectivo, dos Clientes cujo Vendas estão compreendidas entre 20000 e 30000?

- Solução Sem Operador Between:

```
SELECT Num, Nome
FROM Cliente
WHERE Vendas >= 20 000 AND Vendas <= 30 000
```

- Solução Com Operador Between:

```
SELECT Num, Nome
FROM Cliente
WHERE Vendas BETWEEN 20 000 AND 30 000
```

Construção de Condições

Outros Operadores (BETWEEN, IN, IS)

Operador IN permite especificar conjuntos de valores

SELECT ...

FROM ...

WHERE campo [NOT] IN (valor1 , valor2 , valorn)

Exemplo: **Qual o nome, morada e contacto dos clientes da cidade de Lisboa e Coimbra ?**

- Solução Sem Operador IN:

```
SELECT Nome, Morada, Telefone
FROM Cliente
WHERE Cidade = "Lisboa" OR Cidade = "Coimbra"
```

- Solução Com Operador IN

```
SELECT Nome, Morada, Telefone
FROM Cliente
WHERE Cidade in ( "Lisboa","Coimbra")
```

Construção de Condições

Outros Operadores (BETWEEN, IN, IS)

O operador IS permite fazer comparações com o valor NULL (NULL – valor que indica o não preenchimento)

SELECT ...
FROM ...
WHERE Campo IS [NOT] NULL

Exemplo: Qual o nome, morada e contacto dos clientes que possuam morada definida?

```
SELECT Nome, Morada, Telefone
FROM Cliente
WHERE Morada IS NOT NULL
```

Exemplo

Qual o nome, morada e contacto dos clientes cujas Vendas estão compreendidas entre 20000 e 30000, que são da cidade de Lisboa e Porto e que possuem morada definida?

```
SELECT Nome, Morada, Telefone
FROM Cliente
WHERE ( Vendas BETWEEN 20000 AND 30000)
AND ( Cidade IN ( "Lisboa","Porto"))
AND (Morada IS NOT NULL);
```

Funções de Agregação

- Têm por objectivo executar cálculos sobre o resultado de um comando SELECT
- Funções de Agregação:
  - **COUNT** Devolve o número de linhas
  - **MAX**(coluna) Devolve o maior valor da coluna
  - **MIN**(coluna) Devolve o menor valor da coluna
  - **SUM**(coluna) Devolve a soma de todos os valores da coluna
  - **AVG**(coluna) Devolve a média **(Average**) de todos os valores da coluna
- As funções Min, Max, Count(…) e Count(*) podem ser utilizadas com qualquer tipo de dados (numéricos, alfanuméricos)
- As funções SUM e AVG apenas podem ser aplicadas a campos numéricos.

Funções de Agregação - COUNT

- COUNT (*)
  - devolve o número de linhas que resulta de um SELECT
- COUNT(Coluna)
  - devolve nº de ocorrências na coluna diferentes de NULL
- COUNT(DISTINCT Coluna)
  - devolve o nº de ocorrências (sem repetições) na coluna

Funções de Agregação - COUNT

- Para SELECCIONAR e CALCULAR

  SELECT <f_agregacao>
  FROM <tabelas>
  [WHERE <condicao>]

- **<f_agregacao>-**função a aplicar ao resultado da selecção
- **<condicao> -** condição de selecção dos elementos sobre os quais será aplicada a função de cálculo

ATENÇÃO:

- Quando utilizar funções de agregação não poderá seleccionar campos dos registos (excepto com o GROUPBY)

Funções de Agregação - COUNT

- Exemplos 1: Qual o número de clientes da empresa?

```
SELECT Count(*)
FROM Cliente
```

- Exemplo 2: Qual menor venda a um cliente de Lisboa?

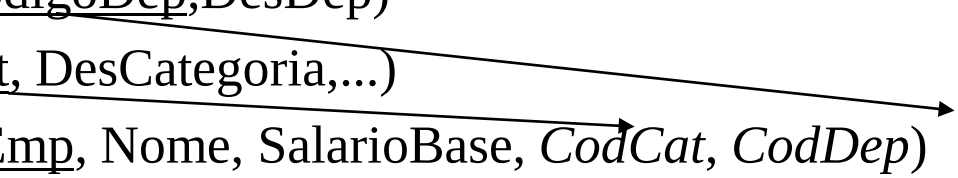
```
SELECT Min(Vendas)
FROM Cliente
WHERE Cidade="Lisboa"
```

- Exemplo 3: Qual o número de cidades?

```
SELECT Count(DISTINCT Cidade)
FROM Cliente
```

Agrupamento

- Departamento(CodigoDep,DesDep)
- Categoria(CodCat, DesCategoria,...)
- Empregado(CodEmp, Nome, SalarioBase, *CodCat*, *CodDep*)



- Qual o número de funcionários por departamento ?
- Qual o montante de salários por departamento ?
- Qual a média salarial por categoria profissional ?
- Qual o salário mínimo praticado em cada Dept para cada Categoria?

- AGRUPAR e CALCULAR

Agrupamento

- Para Agrupar - GROUP BY

  SELECT <colunas>

  FROM <tabelas>

  [WHERE <condicao>]

  [GROUP BY <Campo1, Campo2,…>]

- A filosofia é deixar de ter registos isolados para passar a ter agrupamentos.Perde-se a identidade individual em favor da categorização.

Portanto há que ter em ATENÇÃO:

- Os elementos do SELECT não poderão referir características de elementos individuais mas de grupo, que poderão ser: cálculos referentes aos grupos, ou propriedades identificadoras de cada grupo
- A condição da cláusula WHERE é aplicada sobre cada elemento do grupo
- O agrupamento é realizado com base nos atributos escritos na cláusula GROUP BY.

Agrupamento

- **Para Calcular** – Funções de Agregação

- Têm por objectivo executar cálculos autónomos dentro de cada grupo especificado pelo GROUP BY

Agrupamento

- Qual o número de funcionários por departamento ?
  - Pretende-se: Departamento, Número de Empregados
  - Departamento: Atributo de Agrupamento (DesDep)
  - Número Empregados: Função Count

```
SELECT Departamento.DesDep, Count(*)
FROM Empregado, Departamento
WHERE Empregado.CodDep = Departamento.CodigoDep
GROUP BY Departamento.DesDep
```

Uma alternativa (qual a diferença?)

```
SELECT CodDep, Count(*)
FROM Empregado
GROUP BY CodDep
```

Agrupamento

Qualquer coluna que não seja uma função de agregação só pode estar na cláusula SELECT se estiver na cláusula GROUP BY.

SELECT [DISTINCT] coluna, …

FROM tabela, …

WHERE condição

GROUP BY coluna, …

```
SELECT Departamento.DesDep, Min(Empregado.SalarioBase) AS
   Minimo
FROM Empregado, Departamento
WHERE (Empregado.CodDep)=[Departamento].[CodigoDep]
GROUP BY Departamento.DesDep
```

Agrupamento

- Qual o montante de salários por Departamento?
  - Pretende-se: Departamento, Montante Salarial
  - Departamento: Atributo de Agrupamento (DesDep)
  - Montante Salarial: Função SUM

```
SELECT Departamento.DesDep, Sum(Empregado.SalarioBase) AS Total
FROM Empregado, Departamento
WHERE (Empregado.CodDep)=[Departamento].[CodigoDep]
GROUP BY Departamento.DesDep
```

Uma alternativa (qual a diferença?)

```
SELECT CodDep, Sum(SalarioBase) AS Total
FROM Empregado
GROUP BY CodDep
```

Funções de Agregação

```
SELECT Departamento.DesDep, Sum(Empregado.SalarioBase) AS
Total, Empregado.CodCat
FROM Empregado, Departamento
WHERE (((Empregado.CodDep)=[Departamento].[CodigoDep]))
GROUP BY Departamento.DesDep
```

- ERRADO - **CodCat** não é Característica do Agrupamento

Funções de Agregação

- Qual a média salarial por Categoria Profissional?
  - Pretende-se: Categoria, Média Salarial
  - Categoria: Atributo de Agrupamento (**CodCat**)
  - Média Salarial: Função AVG

```
SELECT Categoria.CodCat, Avg(Empregado.SalarioBase) AS
MediaSalario
FROM Empregado, Categoria
WHERE Empregado.CodCat=[Categoria].[codcat]
GROUP BY Categoria.CodCat
```

Funções de Agregação

- Qual o salário mínimo praticado em cada Departamento para cada Categoria?
  - Pretende-se: Departamento, Categoria,Salário Min.
  - Departamento e Categoria: Atributos de Agrupamento (DesDep e CodCat)
  - Salário Mínimo: Função MIN

```
SELECT [Categoria].[CodCat],
  Min([Empregado].[SalarioBase]) AS SalarioMinimo,
  [Departamento].[DesDep]
FROM Empregado, Categoria, Departamento
WHERE ([Empregado].[CodCat]=[Categoria].[codcat]) and
  (Empregado.CodDep=[Departamento].[CodigoDep])
GROUP BY [Categoria].[CodCat], [Departamento].[DesDep];
```

Funções de Agregação

- Qual o salário mínimo praticado em cada Departamento para cada Categoria?
  - Pretende-se: Departamento, Categoria,Salário Min.
  - Departamento e Categoria: Atributos de Agrupamento (cod_dep e cod_cat)
  - Salário Mínimo: Função MIN

```
SELECT coddep,codcat,MIN(salariobase)
FROM Empregado
GROUP BY coddep,codcat
```

Agrupamentos Múltiplos

- Qual o salário mínimo praticado em cada Departamento para cada Categoria?

```
SELECT coddep,codcat,MIN(salariobase)
FROM Empregado
GROUP BY coddep,codcat
```

Restrições sobre os Grupos

- Qual o salário mínimo praticado em cada Departamento cuja média salarial é superior a €20.000 ?
  - Pretende-se: Departamento,Salário Min
  - Departamento: Atributo de Agrupamento (CodDep)
  - Salário Mínimo: Função MIN
  - Restrição: A média salarial do GRUPO > 20.000
- ***Having...***

Restrições sobre os Grupos

- **Solução SQL**

```
SELECT Departamento.codigodep, MIN(salariobase) AS Minimo
FROM Empregado, Departamento
WHERE Empregado.coddep = Departamento.codigodep
GROUP BY Departamento.codigodep
HAVING AVG(salariobase)> 20000
```

Ou

```
SELECT CodDep, MIN(SalarioBase) AS Minimo
FROM Empregado
GROUP BY CodDep
HAVING AVG(SalarioBase)> 20000
```

Restrições sobre os Grupos

- Qual o número de trabalhadores por departamento cujo salário é superior a €10.000, onde a média salarial desses trabalhadores é superior ou igual a €20.000?
- Eliminam-se os empregados com salário inferior ou igual a €10.000
- Agrupam-se os funcionários seleccionados por departamento
- Calcula-se a Média desses funcionários por departamento
- Seleccionam-se os departamentos cuja média é superior ou igual a €20.000
- Faz-se a contagem do número de empregados seleccionados para esse grupo

Restrições sobre os Grupos

1. Eliminam-se os empregados com salário inferior ou igual a 10.000
   WHERE SalarioBase > 10000
2. Agrupam-se os funcionários seleccionados por departamento
   GROUP BY CodigoDep
3. Calcula-se a Média desses funcionários por departamento
4. Seleccionam-se os departamentos cuja média é superior ou igual a 20.000
   HAVING AVG(SalarioBase) >= 20000
5. Faz-se a contagem do número de empregados seleccionados para esse grupo
   SELECT COUNT(*), CodigoDep

Restrições sobre os Grupos

- Qual o número de trabalhadores por departamento cujo salário é superior a €10.000 onde a média salarial desses trabalhadores é superior ou igual a €20.000?

- Pretende-se: Departamento, NºTrabalhadores
- Departamento: Atributo de Agrupamento (cod_dep)
- NºTrabalhadores: Função COUNT
- Restrições:
  - Salário individual > €10.000
  - A média salarial do GRUPO >= €20.000

Restrições sobre os Grupos

```
SELECT CodigoDep,COUNT(*)
FROM Empregado
WHERE Empregado.SalarioBase > 10000
GROUP BY CodigoDep
HAVING AVG(SalarioBase)>= 20000
```

Ordenação

Seleccionar todos os Empregados, ordenando o resultado pelo código de empregado.

Utiliza-se comando SELECT com cláusula ORDER BY:

```
SELECT *
FROM Empregado
ORDER BY CodEmp
```

Ordenação

SELECT [DISTINCT] coluna, … | *
FROM tabela
[WHERE condição]
[GROUP BY ...]
[HAVING ...]
[ORDER BY coluna [ASC|DESC],...]

- A ordenação está baseada no valor do código ASCII de cada carácter.
- Os dígitos aparecem antes dos caracteres alfabéticos e as maiúsculas aparecem antes das minúsculas.
- 0 < 1<… <9 <… <A <B <Z <… <a <b <… <z
- Por omissão a ordenação é Ascendente.

Inserção

Comando INSERT permite fazer inserções de registos na BD

INSERT INTO <tabela> [(<colunas>)]

VALUES (<valores>)

Exemplo:

```
INSERT INTO Autor ( IdAutor, Nome, Apelido )
Values ("111","Antonio","Silva");
```

Também se pode inserir dados a partir de um query:

INSERT INTO <tabela> [(<colunas>)]

SELECT [(<colunas>)]

FROM <tabela>

Alteração

Comando UPDATE permite fazer alterações de registos na BD

UPADTE <tabela>

SET <coluna> = <expressao>,

… = …

[WHERE <condicao>]

Exemplo

```
UPDATE Cliente
SET Cliente.Nome = "Francisca"
WHERE Nome="Francisco"
```

Remoção

Comando DELETE permite apagar registos da BD

DELETE FROM <tabela>

[WHERE <condicao>]

Exemplo

```
DELETE FROM Cliente
WHERE Nome= "António"
```

SubQueries

- As sub queries permitem a utilização de uma expressão SELECT dentro de outra expressãoSELECT.
- Esta funcionalidade é particularmente útil:
  - quando é necessário efectuar cálculos ou
  - obter informação a partir de um conjunto de registos previamente seleccionado.

Onde se pode utilizar?

- ***Sub query no SELECT***
- ***Sub query no WHERE***
- **...**

Sub query no SELECT

- Utiliza-se como se fosse uma coluna normal no SELECT.

```
SELECT IdContrato, Valor, (SELECT sum(Valor) FROM
   Contrato)
FROM Contrato;
```

Ou então

```
SELECT IdContrato, Valor/(SELECT sum(Valor) FROM
   Contrato)
FROM Contrato;
```

Sub query no WHERE - ALL

- Permite seleccionar o maior de todos os valores dos registos.

? - Cliente com o contrato com maior valor

```
Select Nome
From Cliente, Contrato
Where Cliente.nc=contrato.nc  and Valor
  >= all (SELECT Valor FROM Contrato );
```

Sub query no WHERE - ANY

- Permite seleccionar o maior valor de qualquer um dos registos.

- Listar os nomes dos clientes que cujos contratos tenham valores superiores aos máximos de qualquer um dos tipos

```
Select Nome
From Cliente, Contrato
Where Cliente.nc=contrato.nc  and Valor
  >= any (SELECT max(Valor) FROM Contrato
  group by tipo );
```

Sub query no WHERE - EXISTS

- Este operador devolve verdade caso a sub *query* devolva pelo menos uma linha.

Sub query no WHERE – s/Operador

- Também se consegue utilizar uma *sub query* sem recorrer aos operadores.
- Porém, esta fica sempre limitada a retornar só um registo.

Bibliografia

- Pereira, J. L. *Tecnologias de Base de Dados*, 3ª Edição, FCA, 1998 (cap. 2).
- Damas, L. *SQL - Structured Query Language*, FCA, 1999.
- Costa, C. (2002) *SQL Structured Query Language*, Slides UC, ISCTE (versões 2002 a 2017)
- Costa, C. (2005) *SQL Structured Query Language*, Slides UC SDB, Univ. Aberta