Bases de Dados

SQL: STRUCTURED QUERY LANGUAGE

NORTE2020 PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Objetivos do SQL

Uma linguagem de interação com bases de dados deve permitir:

- Criar base de dados e estrutura das relações
- Realizar tarefas sobre os dados (ex: inserir, modificar e eliminar)
- Realizar queries simples e complexas
- Deve permitir que estas tarefas sejam realizadas com o menor esforço para o utilizador
- A estrutura dos comandos deve ser de fácil aprendizagem
- Deve ser portável: deve ser possível utilizar o mesmo comando em diferentes SGBD

SQL é uma destas linguagens

Notação utilizada

- Letras maiúsculas são usadas para palavras reservadas. (Ex: **SELECT**, **FROM**, **WHERE**, **CREATE**, …)
- Letras minúsculas são usadas para as restantes palavras
- Barra vertical indica que se pode escolher entre vários componentes. (Ex: DISTINCT**|**ALL)
- Chavetas indicam elemento obrigatório. (Ex: **{***|nomeColuna**})**
- Parêntesis retos indicam elemento opcional. (Ex: **[**WHERE condicao**])**
- Reticências indicam repetição opcional de um item zero ou mais vezes. (Ex: FROM tabela [,**…**])

Comandos SQL

Um comando SQL contém:

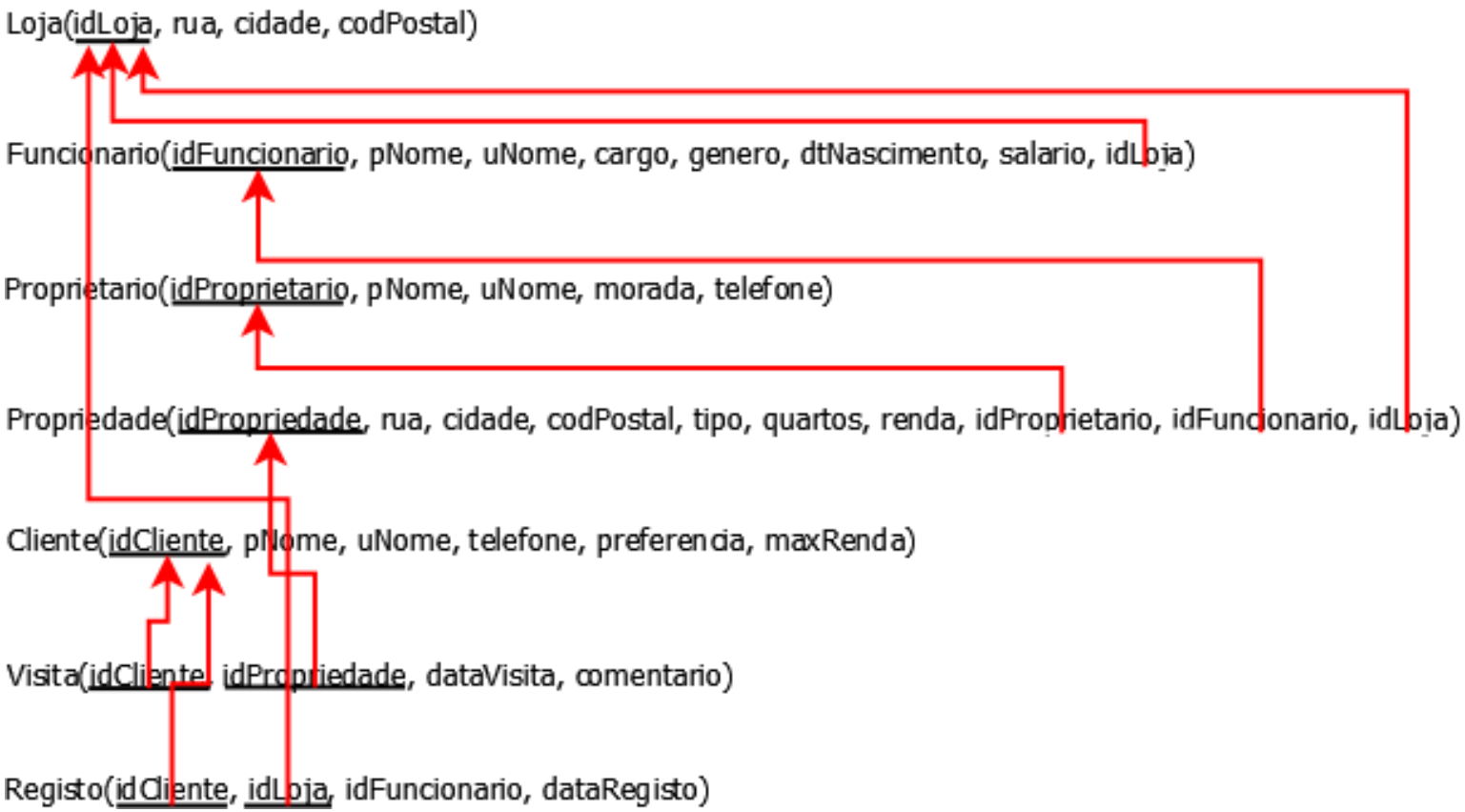
- Palavras reservadas
  - Parte fixa da linguagem
  - Devem ser escritas exatamente da forma prevista
  - Não são case sensitive.
    - SELECT, Select e select são todos válidos
- Palavras definidas pelo utilizador
  - Para representar objetos da BD (tabelas, colunas, vistas,…)
  - Podem conter letras maiúsculas, minúsculas, números e underscore
  - Não podem ter mais do que 128 caracteres
    - em alguns casos o limite é menor
  - Têm de começar por uma letra
  - Não podem conter espaços

Em alguns casos devem terminar com ‘;’

Dados nas tabelas são *case sensitive*.

- ‘Maria’ não é o mesmo que ‘MARIA’.

Estrutura da BD dos exemplos



Definição de dados

Criar uma base de dados:

```
CREATE SCHEMA [nome | AUTHORIZATION idCriador]
```

```
CREATE SCHEMA imobicasa;
```

- Cria a base de dados chamada imobicasa

Eliminar uma base de dados:

```
DROP SCHEMA [RESTRICT | CASCADE]
```

```
DROP SCHEMA imobicasa;
```

- Elimina a base de dados chamada imobicasa

Tabelas

```
CREATE TABLE tabela
{(coluna tipoDados [NOT NULL] [UNIQUE]
[DEFAULT opcao] [CHECK (condicao)] [, … ]}
[PRIMARY KEY (listOfColumns),]
{[UNIQUE (listaColunas)] [, … ]}
{[FOREIGN KEY (listaForeigKeys)
REFERENCES tabelaForeignKey [(listaChaves)]
[MATCH {PARTIAL | FULL}
[ON UPDATE acao]
[ON DELETE acao]] [, … ]}
{[CHECK (condicao)] [, …]})
```

```
DROP TABLE tabela [RESTRICT | CASCADE]
```

Exemplo:

```
CREATE TABLE Loja(
     idLoja CHAR(4) PRIMARY KEY,
     rua VARCHAR(30),
     cidade VARCHAR(30),
     codPostal VARCHAR(10)
);
```

- Cria a tabela chamada Loja, com colunas:
  - **idLoja:** Tipo CHAR(4): tem de ter exatamente 4 caracteres; chave primária
  - **rua:** Tipo VARCHAR(30): pode ter até 30 caracteres
  - **cidade:** Tipo VARCHAR(30)
  - **codPostal:** Tipo VARCHAR(10)

```
DROP TABLE Loja;
```

- Elimina a tabela Loja

Tipos de dados

Tipo de dados	Declaração
Booleanos	BOOLEAN
Carateres	CHAR – quando o tamanho é fixo VARCHAR – quando o tamanho é variável
Bit	BIT BIT VARYING
Numéricos exatos	NUMERIC DECIMALL INTEGER SMALLINT
Numéricos aproximados	FLOAT REAL DOUBLE
Data / Hora	DATE TIME TIMESTAMP
Intervalos	INTERVAL
Objetos grandes	CHARACTER LALRGE OBJECT BINARY LARGE OBJECT

Modificar uma tabela

```
ALTER TABLE tabela
[ADD [COLUMN] coluna tipoDados [NOT NULL] [UNIQUE]
[DEFAULT default] [CHECK (condicao)]]
[DROP [COLUMN] coluna [RESTRICT | CASCADE]]
[ADD [CONSTRAINT [restricao]] defRestricao]
[DROP CONSTRAINT restricao [RESTRICT | CASCADE]]
[ALTER [COLUMN] SET DEFAULT default]
[ALTER [COLUMN] DROP DEFAULT]
```

Adicionar / eliminar coluna

Adicionar / eliminar restrição

Definir / eliminar valor default para coluna

Exemplos:

- Modificar a tabela Funcionario removendo o default 'Assistente' do cargo e fazendo com que o default para a coluna género seja 'F'

```
ALTER TABLE Funcionario
     ALTER cargo DROP DEFAULT;
ALTER TABLE Funcionario
     ALTER género SET DEFAULT 'F';
```

- Modificar a tabela Propriedade removendo a restrição de que os funcionários não podem gerir mais do que 100 casas. Modificar a tabela Cliente adicionando uma nova coluna que representa a preferência quanto ao número de quartos

```
ALTER TABLE Propriedade
     DROP CONSTRAINT menosDe100;
ALTER TABLE Cliente
     ADD prefNQuartos dQuartos;
```

Índices

Criar um índice

```
CREATE [UNIQUE] INDEX índice
ON tabela(coluna [ASC|DEESC][,…])
```

Exemplo:

```
CREATE INDEX idx_cidade
ON Loja(cidade(10));
```

Remover um índice

```
DROP INDEX indice
```

```
DROP INDEX idx_cidade;
```

Seleção (SELECT)

```
SELECT     [DISTINCT | ALL] {* | [coluna [AS novoNome]] [, . . . ]}
FROM       tabela [alias] [, . . . ]
[WHERE     condicao]
[GROUP BY  listaColunas] [HAVING condicao]
[ORDER BY  listaColunas]
```

SELECT: especifica as colunas que devem aparecer no resultado

FROM: especifica a(s) tabela(s) a ser(em) utilizada(s)

WHERE: filtra as linhas de acordo com uma condição

GROUP BY: forma grupos de linhas com o mesmo valor numa coluna

HAVING: filtra os grupos de acordo com uma condição

ORDER BY: especifica a ordem do resultado

Selecionar todas as linhas e todas as colunas.

Listar todos os detalhes de Funcionario

```
SELECT idFuncionario, pNome, uNome, cargo, genero, dtNascimento, salario, idLoja
FROM Funcionario;
```

ou

```
SELECT *
FROM Funcionario;
```

idFuncionario	pNome	uNome	cargo	genero	dtNascimento	salario	idLoja
SA9	Maria	Marques	Assistente	F	1970-02-19	9000	B007
SG14	David	Ferreira	Supervisor	M	1958-03-24	18000	B003
SG37	Ana	Santos	Assistente	F	1960-11-10	12000	B003
SG5	Susana	Silva	Gerente	F	1940-06-03	24000	B003
SL21	Joao	Alves	Gerente	M	1945-10-01	30000	B005
SL41	Julia	Borges	Assistente	F	1965-05-13	9000	B005

Selecionar colunas especificas, todas as linhas

Produzir a lista de salários dos funcionários, mostrando apenas o número de funcionário, primeiro e último nomes e salário.

```
SELECT idFuncionario, pNome, uNome, salario
FROM Funcionario;
```

idFuncionario	pNome	uNome	salario
SA9	Maria	Marques	9000
SG14	David	Ferreira	18000
SG37	Ana	Santos	12000
SG5	Susana	Silva	24000
SL21	Joao	Alves	30000
SL41	Julia	Borges	9000

DISTINCT

Listar os números de todas as propriedades que foram visitadas

```
SELECT idPropriedade
FROM Visita;
```

idPropriedade
PG36
PG4
PG4
PA14
PA14

```
SELECT DISTINCT idPropriedade
FROM Visita;
```

idPropriedade
PG36
PG4
PA14

Campos Calculados

Produzir a lista de salários mensais dos funcionários, mostrando o número de funcionário, primeiro e último nomes e salário.

```
SELECT idFuncionario, pNome, uNome, salario/12
FROM Funcionario;
```

idFuncionario	pNome	uNome	salario/12
SA9	Maria	Marques	750.0000
SG14	David	Ferreira	1500.0000
SG37	Ana	Santos	1000.0000
SG5	Susana	Silva	2000.0000
SL21	Joao	Alves	2500.0000
SL41	Julia	Borges	750.0000

```
SELECT idFuncionario, pNome, uNome, salario/12 AS salarioMensal
FROM Funcionario;
```

idFuncionario	pNome	uNome	salarioMensal
SA9	Maria	Marques	750.0000
SG14	David	Ferreira	1500.0000
SG37	Ana	Santos	1000.0000
SG5	Susana	Silva	2000.0000
SL21	Joao	Alves	2500.0000
SL41	Julia	Borges	750.0000

Seleção de linhas (WHERE)

Condições básicas:

- Comparação: comparar o valor de uma expressão com o de outra expressão
  - =, < >, !=, <, <=, >, >=
  - AND e OR para expressões mais complexas
    - Expressão avaliada da esquerda para a direita
    - Sub-expressões entre parêntesis avaliadas primeiro
    - NOT avaliado antes de AND e OR
    - AND avaliado antes de OR
    - Recomenda-se utilização de parêntesis para evitar ambiguidades
- Intervalo: verificar se o valor de uma expressão está dentro de um determinado intervalo
- Membro: verificar se o valor de uma expressão pertence a um conjunto de valores
- Padrão: verificar se o valor de uma expressão corresponde a um determinado padrão
- Null: verificar se o valor de uma expressão é NULL

Seleção por comparação

Listar as moradas de todas as lojas no Porto ou em Braga

```
SELECT *
FROM Loja
WHERE cidade='Porto' OR cidade = 'Braga';
```

idLoja	rua	cidade	codPostal
B003	R. Central 34	Braga	1122
B007	Av. Aliados 2345	Porto	4321

Seleção por intervalo (BETWEEN / NOT BETWEEN)

Listar todos os funcionários com salario entre 20000 e 30000

```
SELECT *
FROM Funcionario
WHERE salario BETWEEN 20000 AND 30000;
```

ou

```
SELECT *
FROM Funcionario
WHERE salario >= 20000 AND salario <= 30000;
```

idFuncionario	pNome	uNome	cargo	genero	dtNascimento	salario	idLoja
SG5	Susana	Silva	Gerente	F	1940-06-03	24000	B003
SL21	Joao	Alves	Gerente	M	1945-10-01	30000	B005

Seleção por membro (IN / NOT IN)

Listar todos os gerentes e supervisores

```
SELECT *
FROM Funcionario
WHERE cargo IN ('Gerente','Supervisor');
```

Ou

```
SELECT *
FROM Funcionario
WHERE cargo='Gerente' OR cargo='Supervisor';
```

idFuncionario	pNome	uNome	cargo	genero	dtNascimento	salario	idLoja
SG14	David	Ferreira	Supervisor	M	1958-03-24	18000	B003
SG5	Susana	Silva	Gerente	F	1940-06-03	24000	B003
SL21	Joao	Alves	Gerente	M	1945-10-01	30000	B005

Padrões

% e **_**

% representa qualquer sequência de caracteres

_ representa um caracter

LIKE 'H%': o primeiro carater tem de ser um H, mas o resto pode ser qualquer coisa

LIKE 'H_ _ _': a string deve ter exatamente 4 carateres e o 1º tem de ser um H

LIKE '%e': qualquer sequência de carateres que termine num e

LIKE '%Braga%': sequência de carateres de qualquer tamanho que contenha 'Braga'

NOT LIKE 'H%': o primeiro carater não pode ser um H

Se a string puder conter o carater % ou _ devemos utilizar um carater de escape.

Exemplo: **LIKE '15#%' ESCAPE '#'** para encontrar a string '15%'

Seleção por padrões

Listar todos os proprietários que vivam em Braga

```
SELECT *
FROM Proprietario
WHERE morada LIKE '%Braga%';
```

idProprietario	pNome	uNome	morada	telefone
CO40	Telma	Silva	Rua Direita 123 Braga	253986758
CO87	Carolina	Marques	Rua do Meio 54 Braga	253985746
CO93	Antonio	Magalhães	Rua Principal 987 Braga	255098345

Seleção por NULL

Listar os detalhes das visitas à propriedade PG4 em que o cliente não tenha feito comentário

```
SELECT *
FROM Visita
WHERE idPropriedade='PG4' AND comentario IS NULL;
```

idCliente	idPropriedade	dataVisita	comentario
CR56	PG4	2005-05-26	NULL

Ordenação por uma coluna

Produzir uma lista de funcionários por ordem decrescente de salario

```
SELECT *
FROM Funcionario
ORDER BY salario DESC;
```

idFuncionario	pNome	uNome	cargo	genero	dtNascimento	salario	idLoja
SL21	Joao	Alves	Gerente	M	1945-10-01	30000	B005
SG5	Susana	Silva	Gerente	F	1940-06-03	24000	B003
SG14	David	Ferreira	Supervisor	M	1958-03-24	18000	B003
SG37	Ana	Santos	Assistente	F	1960-11-10	12000	B003
SA9	Maria	Marques	Assistente	F	1970-02-19	9000	B007
SL41	Julia	Borges	Assistente	F	1965-05-13	9000	B005

Ordenação por múltiplas colunas

Produzir uma lista de propriedades ordenada por tipo e renda de forma crescente

```
SELECT *
FROM Propriedade
ORDER BY tipo, renda ASC;
```

idPropriedade	rua	cidade	codPostal	tipo	quartos	renda	idProprietario
PG4	R. Direita 54	Braga	1122	Apartamento	3	350	CO40
PG36	Av. Central 32	Braga	1122	Apartamento	3	375	CO93
PL94	R. Principal 42	Felgueiras	123	Apartamento	4	400	CO87
PG16	R. Sameiro 87	Braga	1122	Apartamento	4	450	CO93
PG21	R. Bom Jusus 32	Braga	1122	Moradia	5	600	CO87
PA14	Av Boavista 12	Porto	4321	Moradia	6	650	CO46

Funções de Agregação

COUNT, SUM, AVG, MIN, MAX

- Operam apenas sobre uma coluna
- COUNT, MIN e MAX podem ser utilizados com campos numéricos e não numéricos
- SUM e AVG apenas podem ser utilizados com campos numéricos
- Cada função elimina os NULL e opera apenas sobre os valores não nulos
  - COUNT(*) conta todas as linhas da tabela, incluindo nulos
- DISTINCT não tem efeito sobre MIN e MAX, mas pode ter efeito sobre SUM e AVG
  - Uma query só pode ter um DISTINCT
- Funções de agregação só podem ser usadas na lista do SELECT ou do HAVING

COUNT(*)

Quantas propriedades têm renda superior a 350?

```
SELECT COUNT(*) AS rendaM350
FROM Propriedade
WHERE renda>350;
```

rendaM350
5

COUNT(DISTINCT)

Quantas propriedades diferentes foram visitadas em Maio de 2004?

```
SELECT COUNT(DISTINCT idPropriedade) AS visitaMaio
FROM Visita
WHERE dataVisita BETWEEN '2004-5-1' AND '2004-5-31';
```

visitaMaio
2

COUNT e SUM

Qual é o número de gerentes e a soma dos seus salários?

```
SELECT COUNT(idFuncionario) AS nGerentes, SUM(salario) AS sumSalarios
FROM Funcionario
WHERE cargo='Gerente';
```

nGerentes	sumSalarios
2	54000

MIN, MAX, AVG

Qual é o mínimo, máximo e média dos salários dos funcionários?

```
SELECT MIN(salario) AS minimo, MAX(salario) AS maximo, AVG(salario) AS media
FROM Funcionario;
```

minimo	maximo	media
9000	30000	17000.0000

Agrupamentos (GROUP BY)

Qual é número de funcionários que trabalha em cada loja e a soma dos seus salários

```
SELECT idLoja, COUNT(idFuncionario) AS nrFuncionarios, SUM(salario) AS somaSalarios
FROM Funcionario
GROUP BY idLoja;
```

idLoja	nrFuncionarios	somaSalarios
B003	3	54000
B005	2	39000
B007	1	9000

Restrição em grupos (HAVING)

Utilizado com o GROUP BY para aplicar restrições aos grupos que aparecem nos resultados

Sintaxe semelhante à do WHERE, mas o WHERE filtra linhas, enquanto que o HAVING filtra grupos

Colunas que apareçam no HAVING devem também aparecer no GROUP BY

HAVING

Para cada loja com mais do que um funcionário, encontrar o número de funcionários que trabalham em cada loja e a soma dos seus salários.

```
SELECT idLoja, COUNT(idFuncionario) AS nrFuncionarios, SUM(salario) AS somaSalarios
FROM Funcionario
GROUP BY idLoja
HAVING COUNT(idFuncionario)>1;
```

idLoja	nrFuncionarios	somaSalarios
B003	3	54000
B005	2	39000

Updates à base de dados

INSERT: Inserir novos registos

UPDATE: Editar registos existentes

DELETE: Eliminar registos existentes

INSERT

```
INSERT INTO tabela [(listaDeColunas)]
VALUES (listadeValores)
```

Exemplo:

- Inserir uma nova linha na tabela dos funcionários fornecendo dados para todas as colunas

```
INSERT INTO Funcionario
VALUES('SG16','Andre','Brito','Assistente',
'M','1957-5-25',8300,'B003');
```

- Inserir uma nova linha na tabela dos funcionários fornecendo dados para idFuncionario, pNome, uNome, cargo, salario e idLoja

```
INSERT INTO Funcionario
(idFuncionario,pNome,uNome,cargo,salario,idLoja)
VALUES('SG16','Andre','Brito','Assitente',8300,'B003')
```

=

```
INSERT INTO Funcionario
VALUES('SG16','Andre','Brito','Assitente',
NULL,NULL,8300,'B003')
```

```
INSERT INTO tabela [(listaDeColunas)]
SELECT …
```

Exemplo:

- Se tivermos uma tabela FuncionáriosGeral e quisermos inserir lá apenas o nome de cada um e a loja em que trabalha

```
INSERT INTO FuncionariosGeral
SELECT pNome, uNome, idLoja
FROM Funcionario;
```

- A query dentro do SELECT pode ser qualquer uma, desde que tenha as mesmas colunas da tabela que queremos preencher

UPDATE

```
UPDATE tabela
SET coluna1 = valor1 [, coluna2 = valor2, …]
[WHERE condicao]
```

- Atualizar todas as linhas
  - *Dar a todos os funcionários um aumento de 3%*

```
UPDATE Funcionario
SET salario = salario*1.03;
```

- Atualizar linhas específicas
  - *Dar a todos os gerentes um aumento de 5%*

```
UPDATE Funcionario
SET salario = salario*1.05
WHERE cargo = ‘Gerente’;
```

- Atualizar múltiplas colunas
  - *Promover o funcionário ‘SG14’ ao cargo de gerente e subir-lhe o salario para 18000*

```
UPDATE Funcionario
SET cargo = ‘Gerente’, salario=18000
WHERE idFuncionario = ‘SG14’;
```

DELETE

```
DELETE FROM tabela
[WHERE  condicao]
```

- Eliminar todas as linhas
  - *Eliminar todas as linhas da tabela Visitas*

```
DELETE FROM Visitas
```

- Eliminar linhas específicas
  - *Eliminar todas as visitas que se relacionem com a propriedade 'PG4'*

```
DELETE FROM Visita
WHERE idPropriedade = 'PG4';
```

Tipos de DELETE de FOREIGN KEYS

CASCADE: quando se elimina o valor na tabela principal, elimina-se todas as linhas das outras tabelas que tenham esse valor na foreign key

SET NULL: quando se elimina o valor na tabela principal, é substituído por NULL nas tabelas dependentes

SET DEFAULT: quando se elimina o valor na tabela principal, é substituído pelo valor default nas tabelas dependentes, se existir

NO ACTION: a operação de delete é rejeitada

Exemplo:

```
FOREIGN KEY (idFuncionario) REFERENCES
Funcionario(idFuncionario)
ON DELETE SET NULL
```

Transações

COMMIT: comando termina a transação com sucesso, fazendo com que as alterações à base de dados sejam permanentes

ROLLBACK: comando aborta a transação e as alterações não são gravadas na base de dados

```
SET TRANSACTION
[READ ONLY | READ WRITE] |
[ISOLATION LEVEL READ UNCOMMITTED | READ COMMITTED |
REPEATABLE READ | SERIALIZABLE]
```

Privilégios

SELECT: privilégio para fazer selects

INSERT: privilégio para fazer inserts

UPDATE: privilégio para fazer updates

DELETE: privilégio para fazer deletes

REFERENCES: privilégio para criar foreign keys a partir das tabelas existentes

USAGE: privilégio para usar domains, collations, character sets e translations

GRANT PREVILEGES

```
GRANT {ListaPrevilegios | ALL PRIVILEGES}
ON nomeObjeto
TO {utilizador | PUBLIC}
[WITH GRANT OPTION]
```

WITH GRANT OPTION:
determina que esse utilizador possa fazer GRANT a outros utilizadores.

ListaPrivilégios:

```
SELECT
DELETE
INSERT [(coluna [, . . . ])]
UPDATE [(coluna [, . . . ])]
REFERENCES [(coluna [, . . . ])]
USAGE
```

Exemplo:

- GRANT all privileges
  - *Dar ao utilizador gerente privilegio para tudo na tabela Funcionario*

```
GRANT ALL PRIVILEGES
ON Funcionario
TO Gerente WITH GRANT OPTION;
```

- GRANT privilégios específicos
  - *Dar aos utilizadores Pessoal e Diretor os privilégios de SELECT e UPDATE na coluna salario da tabela Funcionario*

```
GRANT SELECT, UPDATE (salario)
ON Funcionario
TO Pessoal, Diretor;
```

- GRANT privilégios específicos ao público
  - *Dar a todos os utilizadores permissão de SELECT na tabela Loja*

```
GRANT SELECT
ON Loja
TO PUBLIC;
```

Retirar privilégios (REVOKE)

```
REVOKE [GRANT OPTION FOR]
    {listaPrivilegios | ALL PRIVILEGES}
ON objeto
FROM {listaUtilizadores | PUBLIC}
    [RESTRICT | CASCADE]
```

Exemplo:

- Retirar privilégios específicos do público
  - *Retirar o privilégio SELECT da tabela Loja de todos os utilizadores*

```
REVOKE SELECT
ON Loja
FROM PUBLIC;
```

- Retirar privilégios específicos de um utilizador
  - *Retirar os privilégios dados ao Diretor na tabela Funcionario*

```
REVOKE ALL PRIVILEGES
ON Funcionario
FROM Director;
```

Operadores Escalares (1)

Operador	Significado
BIT_LENGTH	Retorna o tamanho da string em bits
OCTET_LENGTH	Retorna o tamanho da string em octetos (bits a dividir por 8)
CHAR_LENGTH	Retorna o tamanho da string em carateres
CAST	Converte um valor de um tipo para outro
||	Concatena duas strings de carateres ou de bits
CURRENT_USER ou USER	Retorna o username do utilizador que está com sessão iniciada
SESSION_USER	Retorna o identificador da sessão SQL
SYSTEM_USER	Retorna o a identificação do utilizador que invocou
LOWER	Converte letras maiúsculas em minúsculas

Operadores Escalares (2)

Operador	Significado
UPPER	Converte letras minúsculas em maiúsculas
TRIM	Remove as ocorrências de um carater do início (LEADING), fim (TRAILING) ou ambos (BOTH). Exemplo: TRIM(BOTH '*' FROM '**Ola**' tem resultado 'Ola'
POSITION	Retorna a posição de uma string dentro de outra Exemplo: POSITION('ai' IN 'Praia') retorna 2
SUBSTRING	Retorna uma substring de dentro de outra Ex: SUBSTRING('Praia' FROM 1 TO 3) retorna 'Pra'
CASE	Retorna um valor de uma lista de valores especificados. Exemplo: CASE tipo WHEN 'Moradia' THEN 1 WHEN 'Apartamento' THEN 2 ELSE 0 END
CURRENT_DATE	Retorna a data atual
CURRENT_TIME	Retorna a hora atual
CURRENT_TIMESTAM	Retorna data e hora atuais
EXTRACT	Retorna o valor de um campo especifico Ex: EXTRACT(YEAR FROM Registo.dataRegisto)

Restrições de Integridade

Dados obrigatórios
- Ex: `cargo VARCHAR(10) NOT NULL`

Restrições a nível de domínio
- Ex: `género CHAR CHECK(género IN ('M','F'))`
- `CREATE DOMAIN tipoGenero AS CHAR DEFAULT 'M' CHECK (VALUE IN ('M','F'))`
  `Genero tipoGenero`
- `DROP DOMAIN tipoGenero`

Integridade de entidades
- Adicionar PRIMARY KEY

Integridade referencial
- Se a tabela tem uma foreign key, o valor inserido deve referir-se a um valor existente na tabela referenciada

CREATE ASSERTION

CREATE ASSERTION nome
CHECK (condições)

Restrições definidas pelo processo de negócio

A imobiliária pode definir que um funcionário não pode gerir mais do que 100 propriedades ao mesmo tempo

```
CREATE ASSERTION menosDe100
      CHECK (NOT EXISTS (SELECT idFuncionario
                         FROM Propriedade
                         GROUP BY idFuncionario
                         HAVING COUNT(*)>100));
```