

# SQL - Parte 1

- Última atualização: 10/03/2023



# Linguagem SQL

- SQL (Structured Query Language): Pode ser dividida em 5 conjuntos:
  - Recuperação de dados: comando Select
  - Linguagem de manipulação de dados: (DML – Data Manipulation Language): comandos para inserções (Insert), atualizações (Update) e exclusões (Delete).
  - Linguagem de definição de dados: (DDL – Data Definition Language): comandos para criação e manutenção dos objetos do banco de dados, Create, Alter, Drop, Rename, Truncate



# Linguagem SQL

- SQL (continuação):
  - Linguagem para controle de transações: Commit, Rollback e Savepoint.
  - Linguagem para controle de acesso a dados: Grant e Revoke.



# Linguagem SQL

- Comandos SQL:
  - Usualmente coloca-se cada cláusula em uma linha separada.
  - Podem ser escritos em maiúsculas ou minúsculas.
  - Devem ser utilizados tabulações e espaços para melhorar a clareza.



# Tipos de Dados

- CHAR(tamanho): seqüência de caracteres de tamanho fixo;
- VARCHAR2(tamanho): seqüência de caracteres de tamanho variável.
- NUMBER(total, decimais), integer,int,numeric, float,small int
- DATE: data e hora
- BLOB (armazena dados não estruturados como imagem e som)

Obs: existem muitos outros tipos de dados no Oracle



# DDL – Create Table

```
CREATE TABLE nome_da_tabela  
  (nome_da_coluna tipo [NULL|NOT NULL], restrições de  
   integridade);
```

```
CREATE TABLE Departamento  
  (id_departamento NUMBER(4) PRIMARY KEY,  
   nome VARCHAR2(20);
```



# DROP TABLE

- DROP TABLE <nome\_da\_tabela> [CASCADE CONSTRAINTS]



# DESC ou DESCRIBE

Desc funcionario;

Nome	Nulo?	Tipo
-----		
ID_FUNCIONARIO	NOT NULL	NUMBER
NOME		VARCHAR2(20)
ID_DEPARTAMENTO		NUMBER





# Tabela Dual

- A tabela dual contém uma coluna chamada dummy e apenas uma linha. É uma tabela criada automaticamente pelo banco de dados.
- Ela pode ser utilizada para retornar o resultado de uma função.

```
select sysdate  
from dual;
```

```
SYSDATE  
-----  
23/07/20
```



# Comandos Básicos

- Para limpar a tela:  
`clear screen;`
- Para executar um arquivo:  
`@L:\BDII\create.sql;`
- Para alterar a senha:  
`ALTER USER <usuário> IDENTIFIED BY <nova senha>`



# ALTER TABLE

```
ALTER TABLE <nome da tabela  
    [ADD definição da coluna,  
    [MODIFY definição da coluna,  
    [DROP COLUMN nome,  
    [RENAME COLUMN nome TO novo_nome,  
    [ADD definição de constraint,  
    [MODIFY CONSTRAINT definição de constraint,  
    [DROP CONSTRAINT nome,  
    [RENAME CONSTRAINT antigo to novo,  
    [ENABLE|DISABLE constraint,  
    [RENAME TO novo nome];
```



# ALTER TABLE: Exemplos

- ALTER TABLE funcionario ADD cpf NUMBER;
- ALTER TABLE funcionario MODIFY nome VARCHAR2(30);



# INSERT

- **INSERT INTO <tabela> [(colunas)] VALUES (valores);**  
insert into funcionario (id\_funcionario,nome,salario,id\_departamento)  
values (1,'Pedro',1200,2);



# Linguagem SQL - SELECT

- Estrutura Básica

- **SELECT**: Utilizada para selecionar os atributos que serão apresentados no resultado
- **FROM**: Tabelas (relações) que serão pesquisadas na consulta
- **WHERE**: Predicado (condição) sobre os atributos das tabelas da cláusula FROM.

SELECT [ALL|DISTINCT] colunas

FROM tabelas|visões|subconsultas

[WHERE cláusula]

[GROUP BY cláusula]

[HAVING cláusula]

[ORDER BY cláusula]

[FOR UPDATE cláusula [NOWAIT]]



# Linguagem SQL - SELECT

- Cláusula SELECT: utilizada para selecionar os atributos que serão apresentados no resultado.

```
select id_funcionario, nome  
from funcionario;
```

ID_FUNCIONARIO	NOME
7369	SMITH
7788	SCOTT



# Views do Dicionário de Dados do Oracle

- Select table\_name from user\_tables;
- Select index\_name from user\_indexes;
- Select constraint\_name from user\_constraints;





# Restrições de Integridade

- Podem ser definidas no momento da criação da tabela ou depois:
  - NOT NULL: a coluna não pode ser nula
  - UNIQUE: a coluna deve ter valores únicos
  - CHECK: condição para valores da coluna



# Restrições de Integridade: Exemplos

```
create table funcionario  
(id_funcionario NUMBER,  
nome_funcionario VARCHAR2(20) constraint nn_nome not null,  
salario NUMBER,  
id_departamento NUMBER,  
id_cargo NUMBER,  
cpf char(11),  
dependente char(1),  
constraint pk_funcionario PRIMARY KEY(id_funcionario),  
constraint u_cpf unique(cpf),  
constraint ck_dependente CHECK (dependente IN ('S','N')));
```

```
ALTER TABLE funcionario ADD  
CONSTRAINT ck_salario CHECK (salario IS NOT NULL);
```



# Restrições de Integridade: Exemplos

```
CONSTRAINT nome_da_restrição FOREIGN KEY (colunas) REFERENCES tabela_pai  
[ON DELETE CASCADE];
```

```
ALTER TABLE funcionario add constraint FK_funcionario foreign key  
(id_departamento) references departamento;
```



# Restrições de Integridade: Exemplos

Create table funcionario

```
(id_funcionario NUMBER,  
nome_funcionario VARCHAR2(20) constraint nn_nome not null,  
salario NUMBER,  
id_departamento NUMBER,  
id_cargo NUMBER,  
cpf char(11),  
dependente char(1));
```

```
alter table funcionario add(  
constraint pk_funcionario PRIMARY KEY(id_funcionario),  
constraint u_cpf unique(cpf),  
constraint ck_dependente CHECK (dependente IN ('S','N')));
```



# Restrições de Integridade

- ALTER TABLE nome\_da\_tabela DROP CONSTRAINT nome\_da\_restrição;
- ALTER TABLE nome\_da\_tabela DISABLE CONSTRAINT nome\_da\_restrição;
- ALTER TABLE nome\_da\_tabela ENABLE CONSTRAINT nome\_da\_restrição;

Ex: alter table funcionario disable constraint ck\_dependente;



# Sequence

- Uma sequence é um objeto do banco de dados que fornece valores sequenciais:

```
CREATE SEQUENCE nome  
    [START WITH valor]  
    [INCREMENT BY valor]  
    [MAXVALUE valor]  
    [MINVALUE valor]  
    [CYCLYE|NOCYCLE]  
    [CACHE valor| NOCACHE]  
    [ORDER|NOORDER];
```

```
DROP SEQUENCE nome_da_sequence;
```



# Sequence

- Exemplo:

```
CREATE SEQUENCE s_funcionario;
```

```
CREATE SEQUENCE s_departamento  
  start with 10  
  increment by 5;
```

- Para buscar o próximo valor:  
 Select s\_funcionario.nextval  
 From dual;
- Para buscar o valor atual:  
 Select s\_funcionario.currval  
 From dual;



# Coluna Auto Incremento

- A partir do Oracle 12c

```
create table tb1 (  
    id number GENERATED ALWAYS AS IDENTITY,  
    nome varchar2(20)  
);
```

Ou

```
create table tb1 (  
    id number GENERATED AS IDENTITY (START WITH 100),  
    nome varchar2(20)  
);  
insert into tb1(nome) values('Maria');
```

Obs: Não pode ser informado um valor para a coluna id no insert. O Oracle cria uma sequence. A sequence vai ficar associada a tabela o que não acontece quando criamos a sequence. insert into tb1(id, nome) values(2,'Ana'); --Vai dar erro





# Coluna Auto Incremento

-- Com a cláusula by default pode ser informado um valor para a coluna id no insert

```
create table tb3 (  
    id number GENERATED by DEFAULT AS IDENTITY,  
    nome varchar2(20)  
);
```

```
insert into tb3(nome) values('Ana');
```

```
insert into tb3(id,nome) values(2,'Paulo');
```

```
insert into tb3(id,nome) values(null,'Ricardo'); -- Um valor será gerado automaticamente
```



# INSERT

- INSERT INTO <tabela> [(colunas)] VALUES (valores);

```
Insert into funcionario (id_funcionario,nome,salario,id_departamento) values  
  (s_funcionario.nextval,'Pedro',1200,1);
```

```
alter table funcioanario add data_admissao date;
```

```
Insert into funcionario (id_funcionario,nome,salario,id_departamento,data_admissao)  
  values (s_funcionario.nextval,'Ana',4000,1,sysdate);
```

Ou

```
Insert into funcionario (id_funcionario,nome,salario,id_departamento,data_admissao)  
  values (s_funcionario.nextval,'Ana',4000,1,to_date('01/07/2021','dd/mm/yyyy'));
```



# SELECT

- Estrutura Básica

- **SELECT:** Utilizada para selecionar os atributos que serão apresentados no resultado
- **FROM:** Tabelas (relações) que serão pesquisadas na consulta
- **WHERE:** Predicado (condição) sobre os atributos das tabelas da cláusula FROM.

SELECT [ALL|DISTINCT] colunas

FROM tabelas|visões|subconsultas

[WHERE cláusula]

[GROUP BY cláusula]

[HAVING cláusula]

[ORDER BY cláusula]

[FOR UPDATE cláusula [NOWAIT]]



# SELECT

- Cláusula SELECT
    - \*: mostra todas as colunas da tabela
- Ex: Apresentar todos os campos da tabela funcionário para os funcionários do departamento 1;

```
Select *  
From funcionario  
Where id_departamento =1;
```



# SELECT

- Alias: altera o nome da coluna no resultado da consulta

Ex: Apresentar o nome, salario atual e o salário com um aumento de 10% .

```
Select nome,salario salario_atual,  
        salario*1.1 "salario-com-aumento"  
From funcionario;
```



# SELECT

- Criando uma tabela a partir de uma consulta:

```
Create table funcionario_2
```

```
as
```

```
Select id_funcionario,nome,salario
```

```
From funcionario
```

```
Where id_departamento=2;
```



# SELECT

- Cláusula SELECT
  - || - concatenação

Ex:

```
select nome,'Departamento =' || id_departamento departamento  
From funcionario  
Where id_departamento is not null;
```



# SELECT - DISTINCT

- Distinct (elimina duplicatas)
  - Apresentar o id\_departamento para os departamentos que tem funcionários;

```
select distinct id_departamento  
From funcionario;
```





# ORDER BY

- Ordena o resultado de uma consulta baseado em uma ou mais colunas.
- Pode ser ascendente (ASC), descendente (DESC). Se não for especificado será ascendente.
- As colunas da cláusula order by não precisam aparecer obrigatoriamente no select.

Ex: Apresentar o nome salario em ordem decrescente, e o nome do funcionario em ordem crescente.

Select salario, nome

From funcionario

Order by salario desc, nome asc;



# WHERE

- Cláusula WHERE: Especifica condições para as linhas que serão apresentadas no resultado.
  - Operadores de comparação: =, <, >, <=, >=, <>
  - Expressões de intervalo: between ... AND ...
  - Expressões sobre conjuntos: in, any, some, all, exists
  - Conectores lógicos: and, or, not



# WHERE

– Exemplo: Apresentar o nome e o salário dos funcionarios do departamento 1 ou 2 e salário > 1000.

```
select nome,salario  
from funcionario  
where id_departamento in (1,2) and salario > 1000;
```



# WHERE - NULL

- Valores Nulo
  - Ex: Apresentar o nome dos funcionarios que não estão vinculados a nenhum departamento.

```
select nome  
from funcionario  
where id_departamento is null;
```



# LIKE

- Comparações com Like ou Not like
  - %: representa qualquer seqüência de caracteres.
  - \_: representa um único caracter
  - Ex: Apresentar o nome dos Funcionário que começam por M.

```
select nome  
from funcionario  
where nome like 'M%';
```



# UPDATE

- UPDATE tabela  
SET coluna1=valor[,  
                coluna 2=valor,...]  
[WHERE condição];  
  
update funcionario  
set salario = salario\*1.1  
where id\_departamento= 1;



# DELETE

- DELETE FROM tabela[,tabelas]  
[WHERE condição];

delete from funcionario where id\_funcionario=3;



# INNER JOIN, JOIN

Ex: Apresentar o nome dos funcionários e o seu respectivo departamento  
(apenas para aqueles funcionários que estão alocados em um dos departamentos da empresa).

```
select funcionario.nome, departamento.nome  
from funcionario inner join departamento  
on funcionario.id_departamento=departamento.id_departamento;
```

```
select f.nome, d.nome  
from funcionario f inner join departamento d  
on f.id_departamento=d.id_departamento;
```

id_funcionario	nome	salario	id_departamento
1	Ana	3000	1
2	Pedro	6000	1
3	Maria	5000	2

Id_departamento	nome
1	Rh
2	Vendas

Resultado da consulta:

nome	nome
Ana	Rh
Pedro	Rh
Maria	Vendas