ORACLE® DATABASE

O Instrutor

Felipe dos Santos

- 13 anos atuando como administrador de Bancos de Dados (Especialista em Performance e Tunning)
- Atuou em grandes projetos em empresas nacionais e internacionais como Fiat, Gerdau, IBM, Paquetá, Sicredi, Yara.
- Carreira desenvolvida com foco em Oracle, SQLServer e Mongodb
- 9 anos de experiência como Engenheiro de Software
- Bacharel em Sistemas de Informação pela Unisinos e com especialização em Big Data and Data Science pela Universidade Federal do Rio Grande do Sul.

fdsantos@gmail.com

https://www.linkedin.com/in/fdsantosdba

AGENDA

Módulo 1

Suite Oracle; Conceitos; Tipos de dados; Tipos de Comandos; Comandos SQL básicos; Operadores Condicionais e Lógicos; Ordenação de resultados. **+DESAFIO**

Módulo 2

Comandos INSERT, DELETE e UPDATE; Controle de Transações; Consolidação de Dados; Principais Funções SQL. **+DESAFIO**

Módulo 3

Subconsultas; JOINS - Consultas com mais de uma tabela; Modelo de Dados - Diagrama ER. **+DESAFIO**

Módulo 4

Schemas; Sinônimos; Views; Materialized Views; SEQUENCESs; PROCEDUREs; FUNCTIONs; TRIGGERs; ÍNDICES; CONSTRAINTS. **+DESAFIO**

Extras

SQL Injection; Database Link; Oracle Golden Gate; SQL Loader; Tabelas Particionadas; Métodos de Conexão ao Oracle

1 - Configuração do Ambiente

Instalação do Oracle XE:

https://www.oracle.com/br/database/technologies/xe-downloads.html

Arquivos Auxiliares:

https://github.com/felipedossantos/treinamentooracle

Setup inicial:

```
sqlplus / as sysdba
show pdbs
alter session set container=XEPDB1;
@criaambiente.sql
```

2 - Suite de Produtos Oracle

ORACLE DATABASE

ORACLE E-BUSINESS SUITE

ORACLE FUSION MIDDLEWARE WEBLOGIC SERVER

ORACLE DATA GUARD

Java

ORACLE FUSION MIDDLEWARE GOLDENGATE

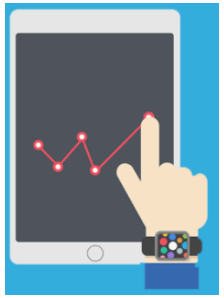
ORACLE RAC Real Application Clusters

ORACLE Application Express

ORACLE CLOUD

3 - O banco de dados Oracle

Afinal, o que é um banco de dados?



SGBD

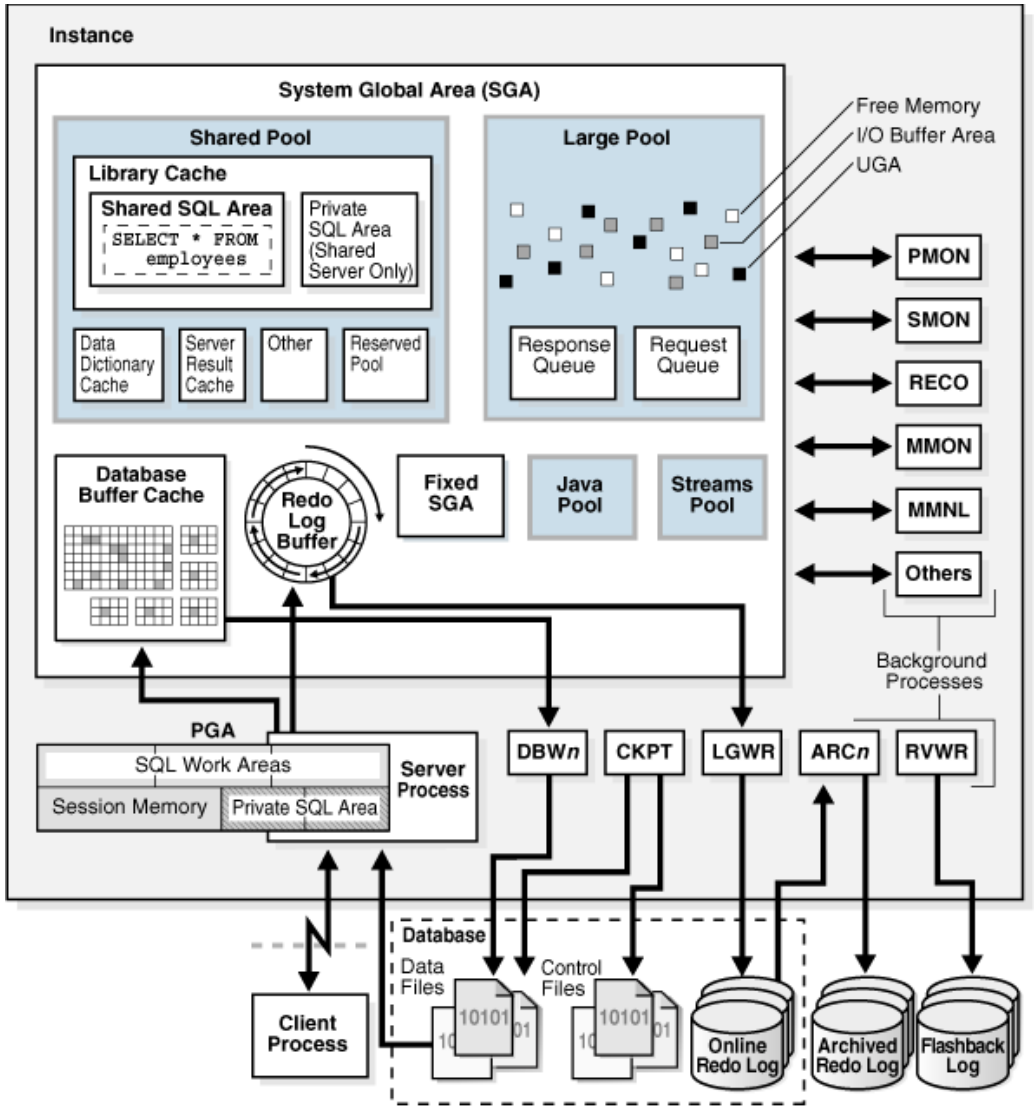
TABELAS

Apresentação

Aplicação

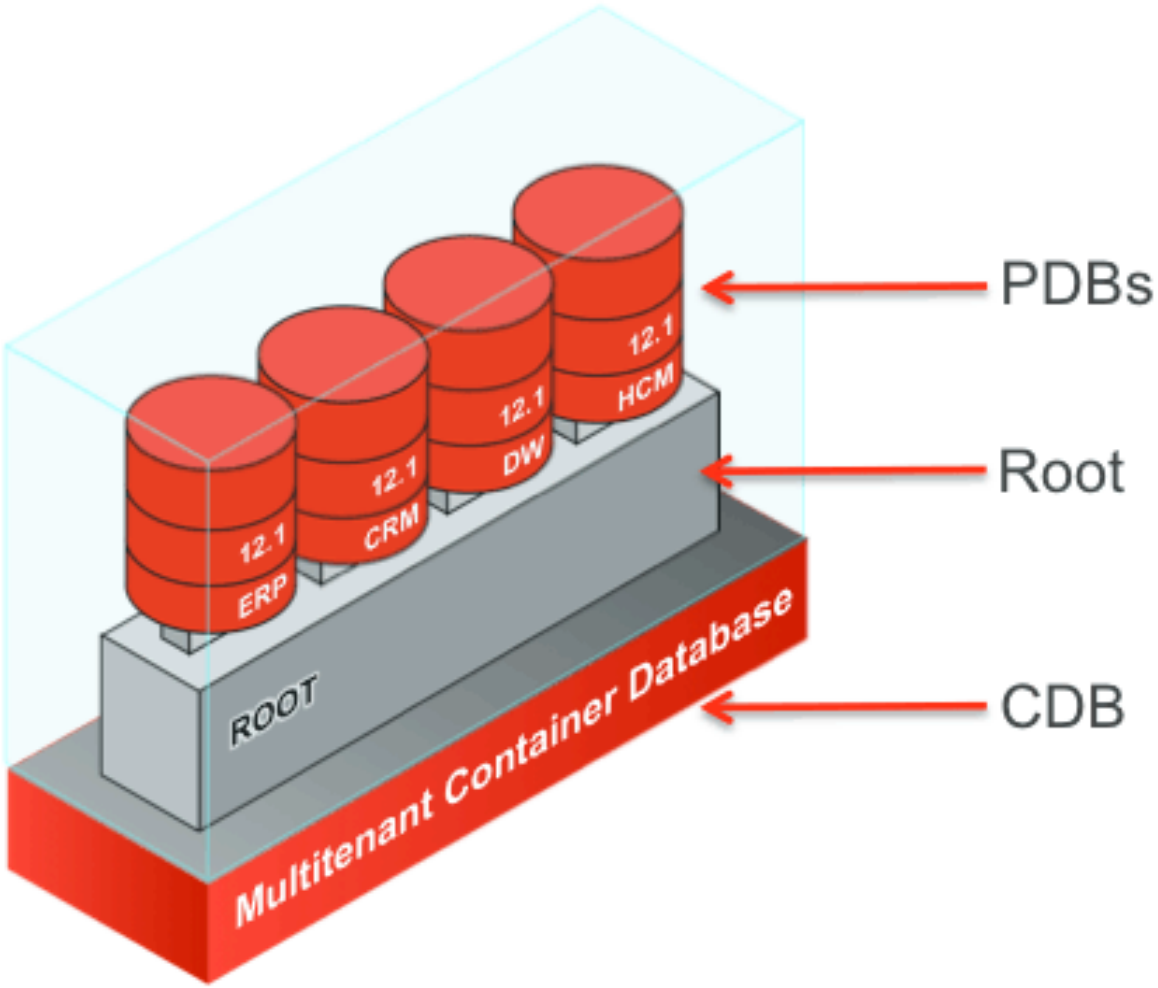
Dados

Instância e Banco



https://docs.oracle.com/en/database/oracle/oracle-database/19/cncpt/introduction-to-oracle-database.html

Oracle Multitenant



https://www.oracle.com/technical-resources/articles/multitenant-on-sparc-solaris.html

SQL (Structured Query Language)

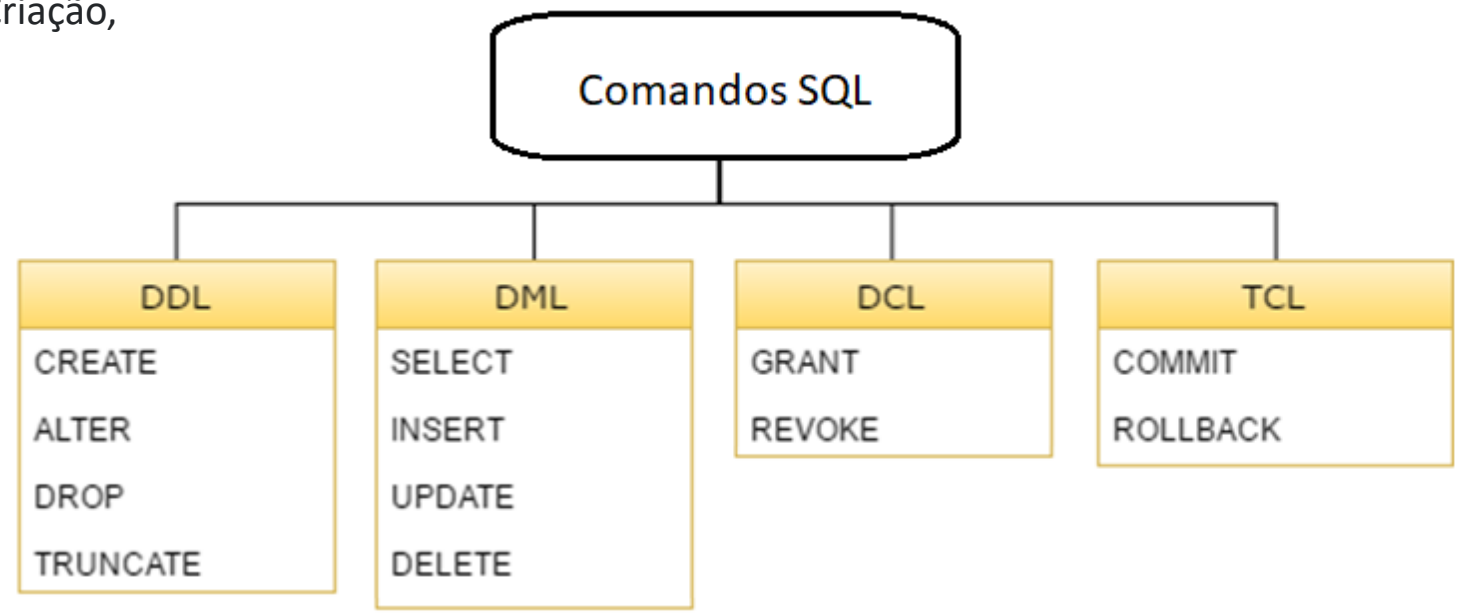
```
select nome,
       data_nascimento,
       rg,
       cpf,
       cliente_desde
    from cliente
    where cliente_desde >= "01/01/2022"
    order by nome
```

DDL x DML x DCL x TCL

• **DDL** - Data Definition Language - Linguagem de Definição de Dados.
São os comandos direcionados aos objetos do banco. Criação, alteração e remoção de estruturas.

• **DML** - Data Manipulation Language
- Linguagem de Manipulação de Dados.
São os comandos que interagem com os dados dentro das tabelas.

• **DCL** - Data Control Language
- Linguagem de Controle de Dados.
São os comandos para controlar a parte de acesso aos dados. Quem pode ver ou manipular cada dado.



• **TCL** - Transaction Control Language - Linguagem de Controle de Transação.
São os comandos para controle de transação.

Conhecendo alguns comandos SQL

- DDL - **CREATE TABLE**: Criando a primeira tabela:

```
create table cliente (idcliente number,
                      nome      varchar(50));
```

- DML – **INSERT**: Inserindo dados na tabela criada:

```
insert into cliente values (1, 'Ana da Silva');
insert into cliente values (2, 'Marcos Ribeiro');
insert into cliente values (3, 'Bruna da Silva');
insert into cliente values (4, 'Pedro Henrique Silva dos Santos');
```

- TCL – **COMMIT**: Confirmando a gravação:

```
commit;
```

- DML – **SELECT**: Consultando dados :

```
select * from cliente;                    col nome format a40
```

CREATE TABLE

```
create table nometabela( campo1  NUMBER(5),
                         campo2  VARCHAR2(15),
                         campo3  CHAR(1),
                         campo4  NUMBER(7,2),
                         campo5  NUMBER(3),
                         campo6  INT,
                         campo7  FLOAT,
                         campo8  DATE,
                         campo9  CLOB           )
```

Tipos de Dados

Tipo de Dado	Descrição
CHAR (size) ★	Quantidade fixa de caracteres armazenados. Máximo de 2 mil caracteres
NCHAR(size)	Quantidade fixa de caracteres armazenados. Máximo de 1 mil caracteres (alfabetos árabes, asiáticos, gregos ... Armazena Unicode)
VARCHAR (size)	Quantidade variável até o máximo de 32767 caracteres (não usar)
VARCHAR2 (size) ★	Quantidade variável até o máximo de 32767 caracteres
NVARCHAR2 (size)	Quantidade variável até o máximo de 16383 caracteres. Mesma regra no nchar
CLOB ★	Até 4 gb
NCLOB	Até 4 gb
BLOB	Até 4 gb
LONG	Até 2gb (não é mais usado)
NUMBER (p, s) ★	Máximo de 38 caracteres (mais detalhes próximos slides)
DATE ★	01/jan/4712 B.C até 31/dez/4712 A.D
ROWID	Endereço binário de cada registro. Pode ser apenas consultado
FLOAT	Ponto Flutuante (mais detalhes próximos slides)

Campo tipo NUMBER

Input Data	Specified As	Stored As
7,456,123.89	NUMBER	7456123.89
7,456,123.89	NUMBER (*,1)	7456123.9
7,456,123.89	NUMBER (9)	7456124
7,456,123.89	Number (9,2)	7456123.89
7,456,123.89	Number (9,1)	7456123.9
7,456,123.89	NUMBER (6)	(not accepted, exceeds precision)
7,456,123.89	Number (7,-2)	7456100

https://docs.oracle.com/cd/A58617_01/server.804/a58241/ch5.htm

```
create table TSTVALOR ( vlr1 integer,
                        vlr2 int,
                        vlr3 numeric,
                        vlr4 number,
                        vlr5 number(4,2) );

insert into TSTVALOR values ( 12.345, 12.345, 12.345, 12.345, 12.345);

DESC TSTVALOR    <<<<<<< Mostra as colunas de uma tabela

Nome                                      Nulo?    Tipo
 ----------------------------------------- -------- ----------------------------
 VLR1                                               NUMBER(38)
 VLR2                                               NUMBER(38)
 VLR3                                               NUMBER(38)
 VLR4                                               NUMBER          <<<<<< Ponto flutuante
 VLR5                                               NUMBER(4,2)

select * from TSTVALOR ;

      VLR1       VLR2       VLR3       VLR4       VLR5
---------- ---------- ---------- ---------- ----------
        12         12         12     12,345      12,35
```

```
insert into TSTVALOR values ( 12.345, 12.345, 12.345, 12.345, 99.994);

1 linha criada.

insert into TSTVALOR values ( 12.345, 12.345, 12.345, 12.345, 99.995);

ORA-01438: valor maior que a precisão especificada usado para esta coluna
```

Campo tipo FLOAT – Igual ao Number sem valor

```
create table tstpontoflutuante ( Preco float);

insert into tstpontoflutuante values (123.12345);
insert into tstpontoflutuante values (1234567.12);
insert into tstpontoflutuante values (1.12345678901234567890);

select * from tstpontoflutuante;

                 PRECO
----------------------
              123.12345
             1234567.12
 1.123456789012345679
```

col preco format 99.99999999999999999999

TRUNCATE, DROP e ALTER TABLE

- TRUNCATE - Limpa todos os dados da tabela de forma instantânea:

```
TRUNCATE TABLE Cliente;
```

- DROP TABLE - Remove a tabela (Dados e Estrutura):

```
DROP TABLE Cliente;
```

- ALTER TABLE NomeTabela [ADD, MODIFY entre outros]

```
ALTER TABLE tstpontoflutuante ADD    desconto number(4,2);
ALTER TABLE tstpontoflutuante MODIFY desconto number(6,4);
```

OBS: Modify pode ser feito sempre para aumentar um campo.
Só consegue diminuir se a coluna estiver vazia

Recriando objeto dropado

```
create table cliente (idcliente number,
                      nome      varchar(50),
                      estado    char(2),
                      cpf       varchar(14),
                      cidade    varchar(50));

insert into cliente values (1, 'Ana da Silva', 'SP','123.456.789-01','Sao Paulo');
insert into cliente values (2, 'Marcos Ribeiro','RJ','123.456.789-02','Rio de Janeiro');
insert into cliente values (3, 'Bruna da Silva','SP','123.456.789-03', 'Sao Paulo');
insert into cliente values (4, 'Pedro Henrique Silva dos Santos','SP','123.456.789-04','Sao
Paulo');
insert into cliente values (5, 'Catarina Ribeiro','RJ','123.456.789-05','Rio de Janeiro');
insert into cliente values (6, ‘Mario de Andrade','SC','123.456.789-06','Pomerode’);

commit;

select * from cliente;
```

set linesize 200

SELECT (Básico)

- Informa que queremos listar todas as colunas:

```
SELECT * FROM cliente;
```

- Ou podemos informar as colunas:

```
SELECT nome, idcliente FROM cliente;
```

- || (2 pipes) utilizado para concatenação:

```
SELECT 'Cliente: ' || idcliente || ' -> ' || nome FROM cliente;
```

- AS para nomear uma coluna:

```
SELECT idcliente as codigo, nome nomecliente, idcliente cod FROM cliente;
```

- rownum é utilizada pra limitar linhas:

```
SELECT nome "Nome do Cliente" FROM cliente WHERE rownum <=3;
```

- Distinct é utilizado para remover valores duplicados do resultado:

```
SELECT DISTINCT estado FROM cliente;
```

- Podemos dar um apelido para uma tabela. Esse apelido pode ser usado toda vez que formos referenciar uma coluna dessa tabela na query:

```
SELECT C.estado FROM cliente C;
```

A tabela auxiliar DUAL

- É uma tabela auxiliar presente em todos os bancos de dados Oracle.
- Pode ser utilizada sempre que precisar do retorno de alguma função, fazer algum cálculo etc.

```
SQL> desc dual

 Nome   Nulo?    Tipo
 -----  -------- ------------
 DUMMY           VARCHAR2(1)


SQL> select * from dual;

DUMMY
-----
X
```

```
SQL> select (10 + 5) / 2 from dual;

7,5

SQL> select sysdate from dual;

24/11/21

SQL> select length('TamanhoDesseTexto')
from dual;

17
```

SELECT – Funções de Agregação

- Conta quantidade de registros:

```
SELECT COUNT(*),  COUNT(1) FROM conta;
```

- Soma todos os valores da coluna:

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

- Retorna o menor valor da coluna:

```
SELECT MIN(saldo)  FROM conta;
```

- Retorna o maior valor da coluna:

```
SELECT MAX(saldo)  FROM conta;
```

- Retorna a média da coluna:

```
SELECT AVG(saldo)  FROM conta;
```

```
SELECT COUNT(conta),  COUNT(idconjuge) FROM conta;
Count por idconjuge está correto?
```

O famoso dado NULL

- NULL significa **ausência** de dado

- NULL não é o mesmo que “ “ << Um espaço em branco
- NULL não é o mesmo que “” << String com nenhum caracter

- Guardem bem essa informação!

Operadores

OPERADOR	TIPO	DESCRIÇÃO
AND	Lógico	Retorna verdadeiro se as duas condições forem verdadeiras
OR	Lógico	Retorna verdadeiro se uma das duas condições forem verdadeiras
NOT	Lógico	Retorna verdadeiro se condição for falsa
=	Comparação	Igual a
<>	Comparação	Diferente de
>	Comparação	Maior que
<	Comparação	Menor que
>=	Comparação	Maior ou Igual a
<=	Comparação	Menor ou igual a
Between And	Comparação	Entre o intervalo de dois valores
Like	Comparação	Padrão de comparação permitindo coringas
IN	Comparação	Dentro de uma lista de valores
IS NULL	Comparação	valida inesistência de dado
EXISTS	Comparação	valida se existe algum registro

SELECT – Condicionais

```
SELECT * FROM cliente WHERE idcliente > 3;
SELECT * FROM cliente WHERE idcliente <> 3;
SELECT * FROM cliente WHERE estado    IN ('SP','RJ');
SELECT * FROM cliente WHERE nome      LIKE '%Ribeiro';
SELECT * FROM cliente WHERE nome      LIKE '%Silva%';
SELECT * FROM cliente WHERE nome      LIKE 'Bruna%';
SELECT * FROM conta   WHERE idconjuge IS NULL;
SELECT * FROM conta   WHERE idconjuge IS NOT NULL;
SELECT * FROM conta   WHERE saldo     BETWEEN 2000 AND 3000;
```

SELECT – Operadores lógicos AND e OR

```
SELECT * FROM cliente WHERE estado NOT IN ('SP') AND nome       LIKE 'Catarina%';
SELECT * FROM conta   WHERE saldo  >= 2000       AND idconjuge IS NOT NULL;
SELECT * FROM cliente WHERE estado = 'RS'        OR  estado    = 'RJ';
```

- Vamos ver o que vai acontecer nesse caso: Existe alguém com sobrenome Ribeiro que é de RS ou SP?

```
SELECT * FROM cliente WHERE nome   LIKE '%Ribeiro%'
                        AND estado = 'RS'
                         OR estado = 'SP';


SELECT * FROM cliente WHERE (nome LIKE '%Ribeiro%' AND estado = 'RS' ) OR estado
= 'SP';


SELECT * FROM cliente WHERE nome LIKE '%Ribeiro%' AND (estado = 'RS' OR estado =
'SP');
```

SELECT – Ordenação de Colunas

- Ordena pelo nome:

```
SELECT * FROM cliente ORDER BY nome;
```

- Ordena em ordem crescente – default:

```
SELECT * FROM cliente ORDER BY nome ASC;
```

- Ordena em ordem decrescente:

```
SELECT * FROM conta ORDER BY saldo DESC;
```

- Ordena pelo alias da coluna:

```
SELECT Nome, IdCliente ID FROM cliente ORDER BY ID DESC;
```

- Ordena pela posição do campo no select. No caso, ordena por idCliente:

```
SELECT Nome, idCliente FROM cliente ORDER BY 2;
```

Mãos na Massa

Existe uma tabela chamada cartão. E o gerente da empresa pediu para você consultar algumas informações. Ele precisa:

- 1 – Da quantidade de cartões ativos da bandeira Visa.
- 2 – A última data em que um cartão da bandeira Master foi cancelado.
- 3 – Quantidade total de cartões cancelados.
- 4 – Um relatório ordenado por nome, de todos os clientes do estado de São Paulo. Usar a tabela cliente. Ele quer que o relatório seja exibido em inglês e tenha o seguinte layout de exemplo, onde devem ser concatenados o nome, um texto fixo e o id do cliente:

```
Customer
------------------------------
João is the customer number 5
Maria is the customer number 3
```

AGENDA

Módulo 1

Suite Oracle; Conceitos; Tipos de dados; Tipos de Comandos; Comandos SQL básicos; Operadores Condicionais e Lógicos; Ordenação de resultados. **+DESAFIO**

Módulo 2

Comandos INSERT, DELETE e UPDATE; Controle de Transações; Consolidação de Dados; Principais Funções SQL. **+DESAFIO**

Módulo 3

Subconsultas; JOINS - Consultas com mais de uma tabela; Modelo de Dados - Diagrama ER. **+DESAFIO**

Módulo 4

Schemas; Sinônimos; Views; Materialized Views; SEQUENCESs; PROCEDUREs; FUNCTIONs; TRIGGERs; ÍNDICES; CONSTRAINTS. **+DESAFIO**

Extras

SQL Injection; Database Link; Oracle Golden Gate; SQL Loader; Tabelas Particionadas; Métodos de Conexão ao Oracle

INSERT em detalhes

- Insere registro(s) em uma tabela:

```
SQL> desc cliente
 Nome          Nulo?    Tipo
 ------------ -------- -----------
 IDCLIENTE              NUMBER
 NOME                   VARCHAR2(50)
 ESTADO                 CHAR(2)
 CPF                    VARCHAR2(14)
 CIDADE                 VARCHAR2(50)
```

- Informando os campos na ordem que estão na tabela:

```
INSERT INTO cliente VALUES (7,'Valmir Correa', 'MG', '123.456.789-
07','Contagem');
```

- Informando a ordem dos campos e omitindo alguns:

```
INSERT INTO cliente(Nome, idCliente, CPF)
VALUES ('Isaura Mascarenhas',8,'123.456.789-08');
```

INSERT baseado em select

```
SQL> select * from agencia_produto;

IDAGENCIA PRODUTO
--------- ----------------------------------
0001      CARTAO MASTER
0001      CARTAO VISA
0001      CHEQUE ESPECIAL
0001      CREDITO PESSOAL
0001      FINANCIAMENTO IMOBILIARIO
0001      PIX
0001      SEGURO RESIDENCIAL
0001      INVESTIMENTO
```

```
INSERT INTO agencia_produto
(idagencia, produto)
SELECT '0002', produto
FROM agencia_produto;
```

Vai inserir 8 registros de uma só vez!

```
INSERT INTO cliente
SELECT 9, 'Viviane Camargo', 'RJ',
'123.456.789-09' , 'Niteroi'
FROM dual;
```

DELETE

Apaga registros de uma tabela. Se for um delete da tabela toda, será bem mais lento do que o TRUNCATE, pois com o DELETE ainda é possível fazer rollback, diferente do TRUNCATE. Usa as mesmas condições de where que já vimos. Exemplos:

```
DELETE FROM cliente WHERE cpf = '123.456.789-08';

      DELETE cliente WHERE cidade = 'Pomerode' AND estado = 'SC';

      DELETE cliente WHERE exists (
       select 1 from conta where encerrada = 'SIM'
           and conta.idcliente = cliente.idCliente );

Rollback;
```

UPDATE

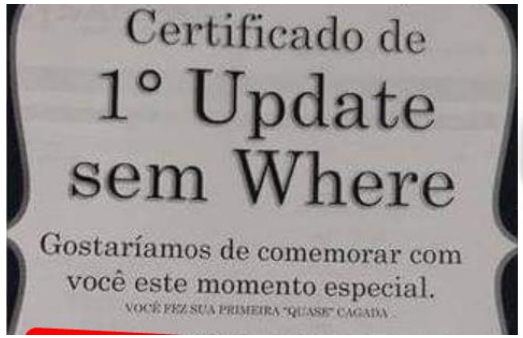
Atualiza dados de um ou mais registros. Usa as mesmas condições de where que já vimos. Exemplos:

```
update cliente
          set Nome = 'Catharina Ribeiro'
          where cpf = '123.456.789-05';

Commit;

update conta
          set saldo = 0,
              encerrada = 'SIM'
          where idcliente = 1;

Rollback;
```

DELETE FROM tabela / DELETE tabela
Apaga todos os dados da tabela

Ou

UPDATE tabela set campo = valor;
Atualiza todos os registros da tabela



TCL – Transaction Control Language

COMMIT;

Utilizado para **CONFIRMAR** uma transação.

ROLLBACK;

Utilizado para **DESFAZER** uma transação.

TCL – Transaction Control Language

Transferência de valor entre contas da mesma instituição:

Transação aberta:

```
Update conta
set saldo = saldo + 250
       where conta = '1111'
         and agencia = '0001'
```

```
Update conta
set saldo = saldo - 250
       where conta = '4444'
         and agencia = '0001';
```

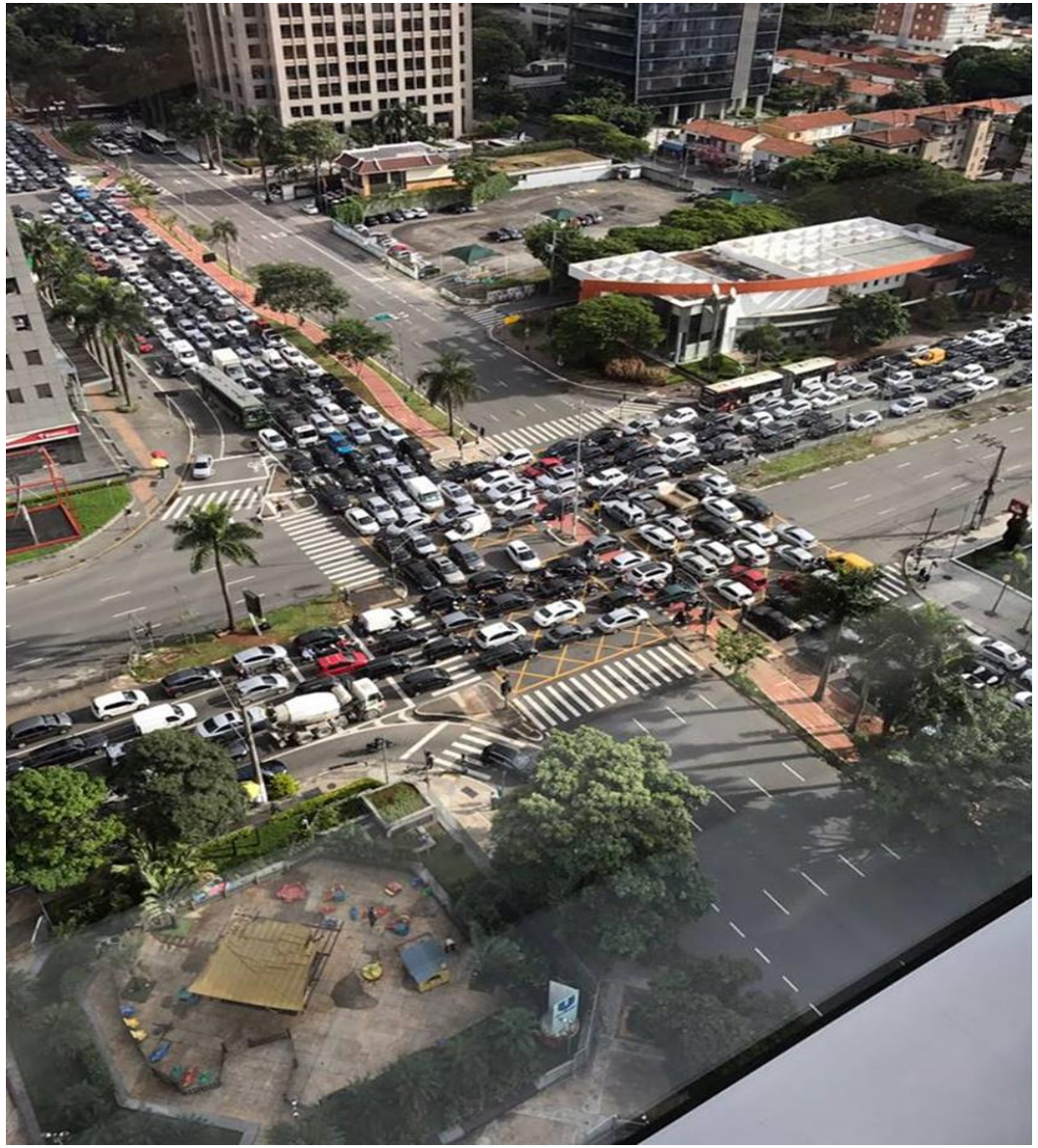
Transação SQL

Se sucesso nos dois updates:

```
Commit;
```

Senão:

```
Rollback;
```



DEAD LOCK

Transação 1 (PIX)

Instante 1:

```
Update conta
 set saldo = saldo + 250
 where conta = '1111'
   and agencia = '0001';
```

Instante 3:

```
Update conta
 set saldo = saldo - 250
 where conta = '4444'
   and agencia = '0001';
```

Transação 2 (TED)

Instante 2:

```
Update conta
 set saldo = saldo - 100
 where conta = '4444'
   and agencia = '0001';
```

Instante 4:

```
Update conta
 set saldo = saldo + 100
 where conta = '1111'
   and agencia = '0001';
```

```
ORA-00060: deadlock detected while waiting for resource
```

Se a rotina de TED fosse implementada igual a rotina de PIX, teríamos problema?

SELECT - Agrupamento

Agrupar informação por campo(s):

```
SQL> SELECT bandeira, count(*)
          FROM cartao where datacancelamento is null
          GROUP BY bandeira;


BANDEIRA      COUNT(*)
---------- ----------
MASTER               2
VISA                 2
```

SELECT - Agrupamento

Agrupando por mais de um campo:

```
SQL> SELECT estado, cidade, count(*) as qtd
          FROM cliente
          GROUP BY estado, cidade order by 1,2;


ESTADO CIDADE                QTD
------ -------------------- ---
MG     Contagem                1
RJ     Niteroi                 1
RJ     Rio de Janeiro          2
SC     Pomerode                1
SP     Sao Paulo               3
NULL   NULL                    1
```

SELECT - Agrupamento

Filtrando o agrupamento:

```
SQL> SELECT estado, cidade, count(*) as qtd
          FROM cliente
          GROUP BY estado, cidade
          HAVING COUNT(*) >= 2;

ESTADO CIDADE           QTD
------ ---------------- ----------
SP     Sao Paulo        3
RJ     Rio de Janeiro   2
```

Precisamos entender que o WHERE é para o filtro da listagem Pré-Agrupamento. Depois de agrupados os dados, o filtro deve ser feito no HAVING.

SELECT – Funções de Conversão de Tipos

TO_CHAR – Transforma data/número em texto

TO_DATE – Transforma texto em data

```
SQL> select TO_CHAR(123.05) || ' é um texto agora’ saida from dual;

saida
-------------------------
123,05 é um texto agora

select TO_DATE('17/11/2021','dd/mm/yyyy')    todate,
       TO_CHAR(sysdate,'dd/mm/yyyy hh24:mi') tochar  from dual;

TODATE   TOCHAR
-------- ---------------
17/11/21 27/12/2021 19:08
```

Todas as máscaras de data: https://docs.oracle.com/cd/B19306_01/server.102/b14200/sql_elements004.htm#i34924

SELECT – Funções de Conversão de Tipos

- TO_NUMBER - Transforma texto em número

```
select TO_NUMBER('1234,567') + 66 tonumber
      from dual;

TONUMBER
---------
1300,567
```

SELECT – Condicional (CASE e DECODE)

```
select CASE estado
        WHEN 'SC' THEN 'Santa Catarina'
        WHEN 'RJ' THEN 'Rio de Janeiro'
        ELSE 'Outro estado'
       END as "Nome do Estado"
   from cliente;

select CASE
        WHEN (estado = 'SC') THEN 'Catarinense'
        WHEN (estado = 'RJ' and cidade <> 'Rio de Janeiro') THEN 'Fluminense'
        WHEN (estado = 'RJ' and cidade =  'Rio de Janeiro') THEN 'Carioca'
        ELSE 'Outro'
       END as "gentilico"
   from cliente;
```

SELECT – Condicional (CASE e DECODE)

- Igual ao comando CASE VALOR WHEN ...

```
select DECODE (estado,'SC','Santa Catarina’,'RJ','Rio de Janeiro', 'Outro
estado’)
   from cliente;
```

- Primeiro parâmetro é o valor a ser testado.
- Valores seguintes em pares são sempre: Condição e o respectivo valor se Verdadeiro.
- Último parâmetro é sempre a condição ELSE (se não for nenhum dos valores anteriores, então coloca esse valor)

SELECT – Principais Funções

```
SELECT  TRIM('  teste  ') tudo,
       RTRIM('  teste  ') direita,
       LTRIM('  teste  ') esquerda,
       TRIM('#' from '#### teste') O
   FROM dual;

TUDO  DIREITA ESQUERDA O
----- ------- -------- ------
teste   teste teste     teste

SELECT ROUND(3.45, 1) A,
       ROUND(3.41, 1) B,
       TRUNC(3.45, 1) C,
       TRUNC(3.45)    D,
       NVL(null,0)    E FROM dual;

A      B      C      D      E
------ ------ ------ ------ ------
   3,5    3,4    3,4      3      0
```

TRIM - Remove espaços esquerda e direita
RTRIM - Remove espaços direita
LTRIM - Remove espaços esquerda
TRIM FROM – Remove o caractere informado da esquerda e direita

ROUND – Arredonda o valor para o número de casas decimais definido.
TRUNC – Trunca o valor para o número de casas decimais definido
NVL - Atribui um valor caso o parâmetro seja null

SELECT – Principais Funções

```
SELECT INITCAP ('mEu teSTe') initcap,
       LOWER('TESTE') lower,
       UPPER('teste') upper,
       LPAD('12,99', 10, '#') lpad
   FROM dual;

INITCAP   LOWER UPPER LPAD
--------- ----- ----- ----------
Meu Teste teste TESTE #####12,99

SELECT SUBSTR('Meu Teste', 5, 5) substr,
       INSTR('Meu Teste','Teste') instr,
       REPLACE( '20.99', '.' , ',' ) repl,
       CHR(65) chr,
       ASCII('A') ascii  FROM dual;

SUBSTR INSTR  REPL  CHR ASCII
------ ------ ----- --- -----
Teste       5 20,99 A        65
```

INITCAP – Deixa todas as palavras do texto com a primeira letra em maiúscula e as demais minúsculas
LOWER – transforma texto em minúsculo
UPPER – transforma texto em maiúsculo
LPAD - Preenche texto com caracteres a esquerda, na quantidade necessária para completar o tamanho de caracteres informado no segundo parâmetro.

SUBSTR – Extrai um pedaço do texto de tamanho Y começando na posição X.
REPLACE – Substitui um caracter por outro.
CHR – Exibe o caracter referente ao código ASC informado.
ASCII – Contrário do CHR, exibe o código ASC do caracter informado.

Mãos na massa!

Um arquiteto de software abriu uma requisição para que seja feita uma cópia da tabela cliente. A nova tabela deve se chamar **auxcliente**.

Essa tabela precisa ter todos os dados existentes hoje na tabela cliente, porém com alguns ajustes:

- O campo idcliente deve ser gravado com zeros a esquerda e ter o tamanho de 5 caracteres.
- O primeiro nome deve ser gravado todo em maiúsculo. O restante do nome mantém como está.
- Os campo estado fica como está.
- O campo CPF não deve mais guardar os dígitos '.' e '-'. Deve ser gravado apenas os 11 números do cpf.
- O campo cidade deve ser gravado todo em minúsculo.
- Um novo campo chamado **regiao** deve ser adicionado no fim da tabela. Ele deve receber o valor 'SUL' se o campo estado for RS, SC ou PR. Qualquer outro estado ele deve ser preenchido como 'OUTRO'.
- Um registro na nova tabela auxcliente deve parecer como esse:

Dicas

- create table
- desc cliente (estrutura da tabela)
  - Funções lpad, upper, substr, instr, replace, lower, case

```
idcliente nome           estado cpf         cidade    regiao
--------- -------------- ------ ----------- --------- ------
00003     BRUNA da Silva SP     12345678903 sao paulo  OUTRO
```

AGENDA

Módulo 1

Suite Oracle; Conceitos; Tipos de dados; Tipos de Comandos; Comandos SQL básicos; Operadores Condicionais e Lógicos; Ordenação de resultados. **+DESAFIO**

Módulo 2

Comandos INSERT, DELETE e UPDATE; Controle de Transações; Consolidação de Dados; Principais Funções SQL. **+DESAFIO**

Módulo 3

Subconsultas; JOINS - Consultas com mais de uma tabela; Modelo de Dados - Diagrama ER. **+DESAFIO**

Módulo 4

Schemas; Sinônimos; Views; Materialized Views; SEQUENCESs; PROCEDUREs; FUNCTIONs; TRIGGERs; ÍNDICES; CONSTRAINTS. **+DESAFIO**

Extras

SQL Injection; Database Link; Oracle Golden Gate; SQL Loader; Tabelas Particionadas; Métodos de Conexão ao Oracle

SUB CONSULTAS

Qual conta tem o maior saldo?

```
IDCLIENTE ENC CONTA  AGEN  IDCONJUGE      SALDO
--------- --- ------ ---- ---------- ----------
        1 NAO 1111   0001                  1100
        2 SIM 2222   0001          5       2000
        3 NAO 3333   0001                  3000
        4 NAO 4444   0001                  3900

select agencia, conta, saldo from conta
   where saldo = ( select max(saldo) from conta);

AGENCIA CONTA  SALDO
------- ------ ----------
0001    4444         3900
```

SUB CONSULTAS

Quanto cada conta representa do total de saldos?

```
select agencia, conta,
(saldo / ( select sum(saldo) from conta) ) * 100 as "%SALDO"
   from conta;

AGEN CONTA      %SALDO
---- ------ ----------
0001 1111           11
0001 2222           20
0001 3333           30
0001 4444           39
```

UNIÃO DE QUERIES

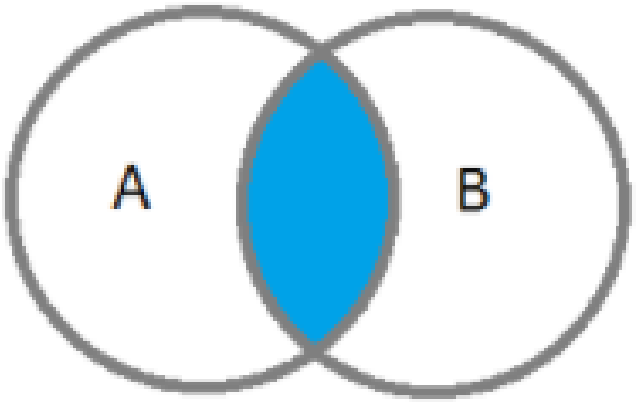
```
SELECT agencia, conta, saldo FROM conta
 where saldo = ( select max(saldo) from conta)
UNION ALL
SELECT agencia, conta, saldo FROM conta
 where saldo = ( select min(saldo) from conta)
ORDER BY saldo;

AGEN CONTA       SALDO
---- ------ ----------
0001 1111          1100
0001 4444          3900
```

OBS: Sempre que possível usar UNION ALL

INNER JOIN – JUNÇÃO - ANSI

Listar o nome dos clientes que possuem cartão VISA:

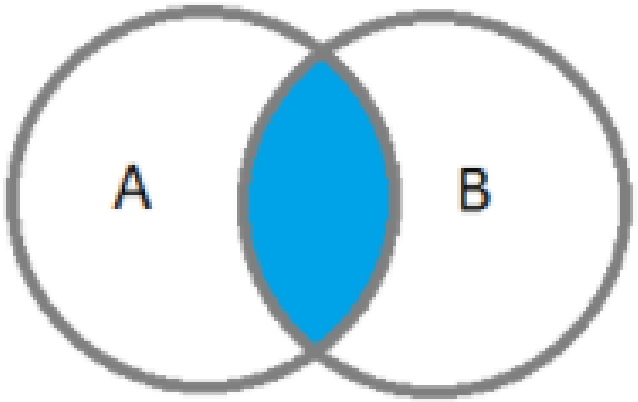
```
SELECT nome
    FROM cliente INNER JOIN cartao
      ON cliente.idcliente = cartao.idcliente
     AND bandeira          = 'VISA'
     AND datacancelamento  is null;


NOME
------------------------------
Pedro Henrique Silva dos Santos
Catharina Ribeiro
```



INNER JOIN – NÃO ANSI

Listar o nome dos clientes que possuem cartão VISA:

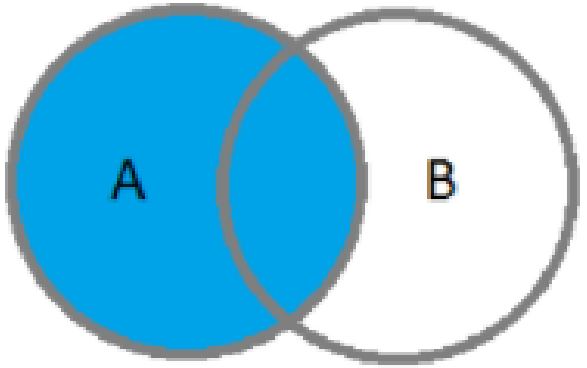
```
SELECT nome
    FROM cliente, cartao
    WHERE cliente.idcliente = cartao.idcliente
     AND bandeira           = 'VISA'
     AND datacancelamento   is null;


NOME
------------------------------
Pedro Henrique Silva dos Santos
Catharina Ribeiro
```



LEFT JOIN - ANSI

Listar o nome do cliente e se ele tiver cartão, listar também os cartões:

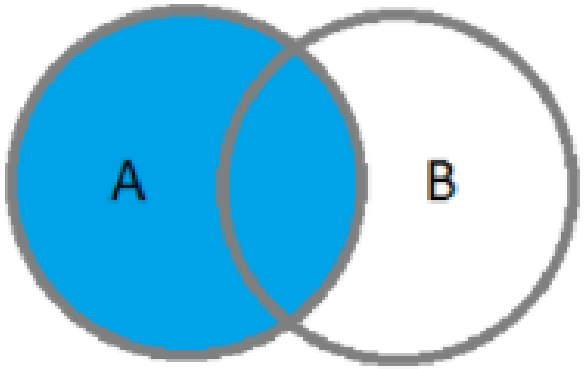
```
SELECT nome, bandeira
    FROM cliente LEFT JOIN cartao
      ON cliente.idcliente = cartao.idcliente
    ORDER BY 1;
NOME                           BANDEIRA
------------------------------ ----------
Ana da Silva                   MASTER
Bruna da Silva                 MASTER
Catharina Ribeiro              VISA
Isaura Mascarenhas
Marcos Ribeiro                 VISA
Mario de Andrade               VISA
Pedro Henrique Silva dos Santos  MASTER
Pedro Henrique Silva dos Santos  VISA
Valmir Correa
Viviane Camargo
```



LEFT JOIN – NÃO ANSI

Listar o nome do cliente e se ele tiver cartão, listar também os cartões:

```
SELECT nome, bandeira
    FROM cliente, cartao
    WHERE cliente.idcliente = cartao.idcliente(+)
    ORDER BY 1;
NOME                           BANDEIRA
------------------------------ ----------
Ana da Silva                   MASTER
Bruna da Silva                 MASTER
Catharina Ribeiro              VISA
Isaura Mascarenhas
Marcos Ribeiro                 VISA
Mario de Andrade               VISA
Pedro Henrique Silva dos Santos  MASTER
Pedro Henrique Silva dos Santos  VISA
Valmir Correa
Viviane Camargo
```



JOIN com mais de 2 tabelas

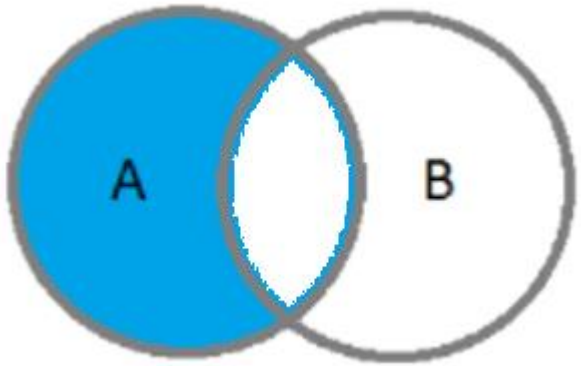
Listar a conta, nome do cliente e nome do cônjuge se existir:

```
SELECT c.conta, cl1.nome, cl2.nome conjuge
    FROM conta c INNER JOIN cliente cl1
      ON c.idcliente = cl1.idcliente
                LEFT JOIN cliente cl2
      ON c.idconjuge = cl2.idcliente;


CONTA  NOME                           CONJUGE
------ ------------------------------ --------------------
1111   Ana da Silva
2222   Marcos Ribeiro                 Catharina Ribeiro
3333   Bruna da Silva
4444   Pedro Henrique Silva dos Santos
```

ANTI JOIN

Listar o nome dos clientes que não possuem cartão:

```
SELECT nome, bandeira
    FROM cliente LEFT JOIN cartao
      ON cliente.idcliente = cartao.idcliente
    WHERE cartao.bandeira is null;


NOME                           BANDEIRA
------------------------------ ---------
Isaura Mascarenhas
Valmir Correa
Viviane Camargo
```

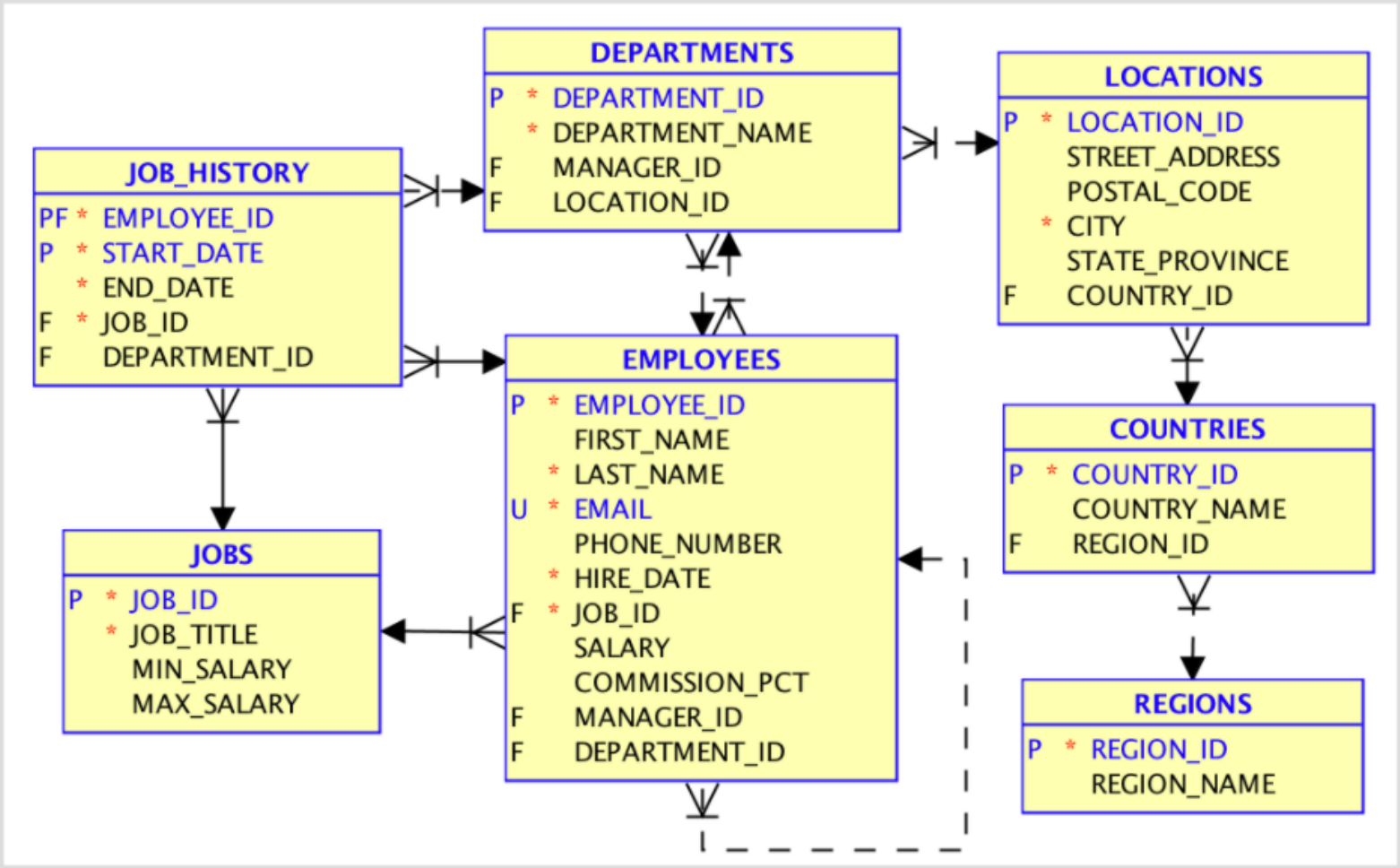


SEMI JOIN

Listar o nome dos clientes que possuem pelo menos 1 cartão

```
SELECT nome
    FROM cliente
    WHERE exists (
       select 1 from cartao
        where cartao.idcliente = cliente.idcliente );

 NOME
--------------------------------
Ana da Silva
Marcos Ribeiro
Bruna da Silva
Pedro Henrique Silva dos Santos
Catharina Ribeiro
Mario de Andrade
```

MODELO ER – Entidade Relacionamento



- **P - PRIMARY KEY**

Chave Primária: Valores únicos por registro

- **F - FOREIGN KEY**

Chave Estrangeira: Valor dessa coluna deve existir na tabela pai, onde na tabela pai esse campo é primary key.

- **U - UNIQUE KEY**

Chave única: Garantia de valor único em uma coluna.

https://docs.oracle.com/en/cloud/paas/autonomous-data-warehouse-cloud/dw_hr_analytics/run-analytics/run-analytics.html#setting-up-the-human-resource-analytics-project-in-oac

MODELO ER – Vamos montar?

CONTA
IDCLIENTE
ENCERRADA
CONTA
AGÊNCIA
IDCONJUGE
SALDO

TRANSACAO
CONTA
AGENCIA
OPERACAO
VALOR
DATA

FUNCIONARIO
IDFUNCIONARIO
NOME
IDCARGO

CARTAO
NUMEROCARTAO
BANDEIRA
DATACANCELAMENTO
CVV
IDCLIENTE

CLIENTE
IDCLIENTE
NOME
ESTADO
CPF
CIDADE

AGENCIA
IDAGENCIA
NOMEAGENCIA
IDGERENTE

CARGO
IDCARGO
CARGO

AGENCIA_PRODUTO
IDAGENCIA
PRODUTO

Mão na massa!

- 1 – A auditoria interna abriu uma requisição urgente dizendo que foi identificada uma operação duplicada. É o mesmo valor sendo creditado para a mesma conta no mesmo segundo. Eles precisam identificar essa operação na tabela TRANSACAO.
- 2 - O diretor financeiro pediu uma listagem com o nome da agência e o nome do gerente de cada agência bem como o somatório dos saldos de todas as contas de cada agência. Sabemos que uma agência é nova e portanto ainda não possui contas. Nesse caso, listar o nome da agência, o nome do gerente e um zero no total do saldo.
- 3 - O gerente da tesouraria pediu para criar uma consulta de conciliação onde mostre o saldo de cada conta e um somatório de todas as transações. Como os valores são todos positivos, deve se usada a coluna operação como auxiliar na query.
- A saída do relatório deve ser parecida com essa:

```
AGENCIA CONTA  SOMA  SALDO
------- ------ ----- -----
0003    1234   1000  1000
0004    7891   3000  2800
```

AGENDA

Módulo 1

Suite Oracle; Conceitos; Tipos de dados; Tipos de Comandos; Comandos SQL básicos; Operadores Condicionais e Lógicos; Ordenação de resultados. **+DESAFIO**

Módulo 2

Comandos INSERT, DELETE e UPDATE; Controle de Transações; Consolidação de Dados; Principais Funções SQL. **+DESAFIO**

Módulo 3

Subconsultas; JOINS - Consultas com mais de uma tabela; Modelo de Dados - Diagrama ER. **+DESAFIO**

Módulo 4

Schemas; Sinônimos; Views; Materialized Views; SEQUENCESs; PROCEDUREs; FUNCTIONs; TRIGGERs; ÍNDICES; CONSTRAINTS. **+DESAFIO**

Extras

SQL Injection; Database Link; Oracle Golden Gate; SQL Loader; Tabelas Particionadas; Métodos de Conexão ao Oracle

SCHEMA

Schema é o usuário/owner dos objetos.

Falamos “Qual o schema da tabela Cliente?” e não “Qual o usuário da tabela Cliente?”.

```
Select owner, table_name
  from all_tables
  where table_name = 'CLIENTE'
     or table_name = 'RELATORIOAUDITORIA';

OWNER                TABLE_NAME
-------------------- ------------------------------
SYS                  CLIENTE
AUDITOR              RELATORIOAUDITORIA
AUDITOR              CLIENTE
```

SYNONYMS

Sinônimo é um apelido que damos para um objeto.

Tentar acessar a tabela transacao que estamos usando:

```
conn auditor/Aud1tor#good@"localhost:1521/xepdb1"
```

Comando GRANT atribui uma permissão:

```
[SYS] GRANT SELECT ON sys.transacao to auditor;

[AUDITOR] SELECT * FROM transacao; << Erro, pois a tabela transacao não
está no schema AUDITOR
          SELECT * FROM sys.transacao;

[SYS] CREATE SYNONYM auditor.transacao FOR sys.transacao;
 Ou   CREATE PUBLIC SYNONYM transacao  FOR sys.transacao;

[AUDITOR] SELECT * FROM transacao;
```

VIEWS – Visão sensível aos dados online

Forma de abstrair uma query complexa, liberando para o usuário uma consulta simples:

```
CREATE VIEW VW_RELATORIO_CONSOLIDACAO AS
select C.agencia,
       C.conta,
       C.saldo
   from conta C inner join transacao T
        on C.agencia = T.agencia
       and C.conta   = T.conta;

SELECT * FROM VW_RELATORIO_CONSOLIDACAO;
```

MATERIALIZED VIEWS – Visão *estática dos dados

Cria uma tabela com o resultado de uma query. Isso nos permite criar outros índices ou até mesmo colocar o objeto em memória, para acelerar as consultas, sem impactar a tabela transacional original.

```
CREATE MATERIALIZED VIEW VW_RELATORIO_CONSOLIDACAO
BUILD IMMEDIATE/DEFERRED
REFRESH FORCE/FAST/COMPLETE
ON DEMAND/COMMIT
AS
select C.agencia ...;
```

MATERIALIZED VIEWS – Visão *estática dos dados

OPÇÕES	DESCRIÇÃO
BUILD IMMEDIATE	Preenche dados no momento da criação.
DEFERRED	Apenas cria a MVIEW vazia. Aguardando uma atualização manual.
REFRESH FAST	Atualiza a MVIEW apenas com as alterações.
COMPLETE	Atualização total dos dados.
FORCE	Tenta FAST e se não conseguir, faz o COMPLETE.
ON DEMAND	Atualização será feita sob demanda.
COMMIT	Atualiza os dados online (Usar apenas em tabelas com poucas transações).

Atualizando uma MVIEW na mão (ON DEMAND):

```
EXEC DBMS_MVIEW.refresh('MVW_RELATORIO_CONSOLIDACAO');
```

SEQUENCES

Estrutura com regras de incremento. Usado para gerar código únicos.

```
CREATE SEQUENCE seq_cargo
 START WITH 5
 INCREMENT BY 1
 CACHE 20;
```

Parâmetros	Descrição
START WITH	Número ao qual a sequência deve iniciar
INCREMENTE BY	Valor que o número deve ser incrementado a cada chamada. Incremento pode ser negativo também.
CACHE	Quantidade de números que deseja armazenar em memória. Em ambientes com alto volume transacional, é uma boa ideia usar cache para evitar contenção na sequence.

SEQUENCES

Para requisitar o próximo valor disponível, basta referenciar a sequence com o comando nextval:

```
INSERT INTO cargo(idcargo, cargo)
    VALUES(seq_cargo.nextval, 'analista investimento');

INSERT INTO cargo(idcargo, cargo)
    VALUES(seq_cargo.nextval, 'auditor');
```

Comando abaixo cria coluna com valor autoincremento. Mas na prática o oracle cria uma sequence:

```
create table cargo2 ( idcargo NUMBER GENERATED ALWAYS as
                                IDENTITY(START with 1 INCREMENT by 1),
                      cargo VARCHAR2(50)    );
```

SEQUENCES

DICA:

NÃO SE APEGUEM A ORDEM DE NUMERAÇÃO DE UMA SEQUENCE.

DESENVOLVAM COM A PREMISSA DE QUE É APENAS UM CODIGO ÚNICO SENDO GERADO PARA UMA CHAVE.

PROCEDURE

Procedimento com código PL/SQL executado dentro do banco

```
create or replace procedure spcriaproduto ( agencia char,
                                            descricao varchar2) as
begin
    insert into agencia_produto values (agencia,descricao);
end;
/
exec spcriaproduto('0004','novo produto');

SQL> select * from agencia_produto where IDAGENCIA=4;

IDAG PRODUTO
---- ------------------------------------
0004 novo produto
```

O QUE ACONTECEU???

```
SQL> exec spcriaproduto('000A','novo produto 2’);

Procedimento PL/SQL concluído com sucesso.

SQL> select * from agencia_produto where IDAGENCIA=4;
select * from agencia_produto where IDAGENCIA=4
                                  *
ERRO na linha 1:
ORA-01722: número inválido
```

FUNCTION

Bloco de código na forma de função, retornando um valor

```
CREATE OR REPLACE FUNCTION fu_saldo_conta( p_ag char, p_conta char )
RETURN number
IS
   varsaldo number;
BEGIN
    select saldo into varsaldo from conta where agencia = p_ag and conta =
p_conta;
    RETURN varsaldo;
END;
/

SQL> select fu_saldo_conta('0001','4444') saldo from dual;

SALDO
-----
 3900
```

TRIGGER

Gatilho. Executa automaticamente um código PL/SQL toda vez que um determinado evento ocorre.

```
CREATE OR REPLACE TRIGGER auditor.tr_bi_cliente
BEFORE INSERT ON auditor.cliente
FOR EACH ROW
BEGIN
    insert into auditor.log
      select :new.cod, sysdate from dual;
END;
/

select * from auditor.log;

insert into auditor.cliente values (111,'MARCELO','S');

select * from auditor.log;
```

TRIGGER

Triggering Statement	OLD.field Value	NEW.field Value
`INSERT`	`NULL`	Pos-insert value
`UPDATE`	Pre-update value	Pos-update value
`DELETE`	Pre-delete value	`NULL`

`:new  :old`

BEFORE INSERT/DELETE/UPDATE

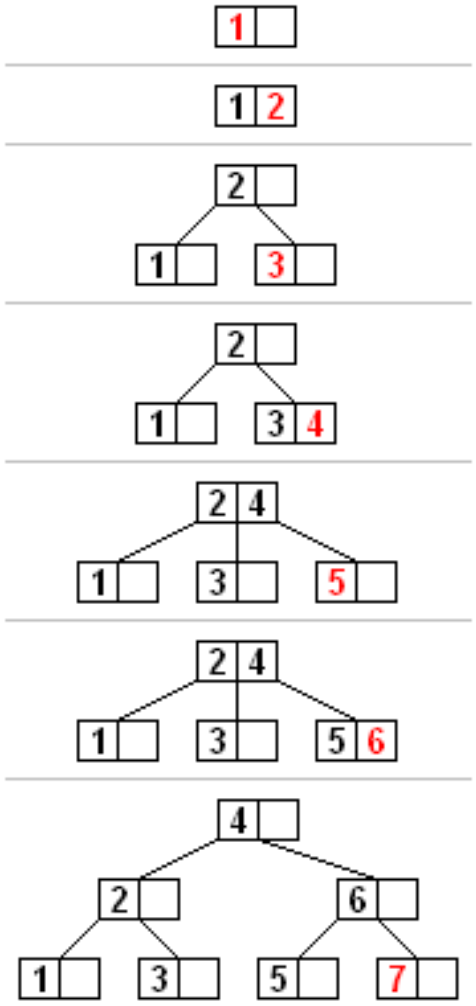
AFTER INSERT/DELETE/UPDATE

Timing Point	Section
Before the triggering statement runs	`BEFORE STATEMENT`
After the triggering statement runs	`AFTER STATEMENT`
Before each row that the triggering statement affects	`BEFORE EACH ROW`
After each row that the triggering statement affects	`AFTER EACH ROW`

FOR STATEMENT / FOR EACH ROW

https://docs.oracle.com/cd/E11882_01/appdev.112/e25519/triggers.htm#LNPLS2005

Índices



https://pt.wikipedia.org/wiki/%C3%81rvore_B

EXPLAIN PLAN FOR – Gera o plano de acesso de uma query

```
explain plan for
select * from transacoes2010 where id = 3455;
```

```
## Comando abaixo exibe o plano de acesso gerado
SELECT PLAN_TABLE_OUTPUT FROM TABLE(DBMS_XPLAN.DISPLAY());

Plano de Acesso:
------------------------------------------------------------------------------
| Id  | Operation                 | Name           | Rows  | Bytes | Cost (%CPU)|
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT          |                |     1 |    20 |     9   (0)|
|*  1 |  TABLE ACCESS STORAGE FULL| transacoes2010 |     1 |    20 |     9   (0)|
------------------------------------------------------------------------------
```

Índices

Criando índice chamado **ix_idtransacao2010** na tabela **TRANSACOES2010** pelo campo **ID**:

```
CREATE INDEX ix_idtransacao2010 on TRANSACOES2010 ( ID );

Plano de Acesso:
------------------------------------------------------------------------------------
| Id  | Operation                          | Name                  | Rows  | Bytes | Cost (%CPU)
------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                       |     1 |    20 |     2   (0)
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| TRANSACOES2010       |     1 |    20 |     2   (0)
|*  2 |   INDEX RANGE SCAN                 | IX_IDTRANSACAO2010    |     1 |       |     1   (0)
------------------------------------------------------------------------------------
```

Índices

```
DROP INDEX ix_idtransacao2010 ;
CREATE UNIQUE INDEX ix_idtransacao2010 on TRANSACOES2010 ( ID );
```

Testar novamente a query:

```
-----------------------------------------------------------------------------------
| Id  | Operation                    | Name                | Rows  | Bytes | Cost (%CPU)|
-----------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                     |     1 |    20 |     2   (0)|
|   1 |  TABLE ACCESS BY INDEX ROWID| TRANSACOES2010      |     1 |    20 |     2   (0)|
|*  2 |   INDEX UNIQUE SCAN          | IX_IDTRANSACAO2010 |     1 |       |     1   (0)|
-----------------------------------------------------------------------------------
```

ESTATÍSTICAS – Ajuda o otimizador de query a escolher o melhor plano de acesso. Maioria dos casos deve ter atualização frequente. Oracle pode atualizar sozinho.

```
exec dbms_stats.gather_table_stats(ownname => 'SYS', tabname => 'TRANSACOES2010');

select table_name, last_analyzed, num_rows from all_tables
   where table_name = 'TRANSACOES2010' and owner = 'SYS';
```

QUANDO CRIAR ÍNDICES?

- Em campos que são usados nos filtros da aplicação. Usar ordem de maior seletividade. (Índices podem ser compostos. Já vamos ver)
- Em todos os campos Foreign Keys.

Exceção quando se sabe que a tabela referenciada não sofre deletes.

Qual o melhor índice para a API de extrato, que recebe como parâmetro o número da conta, número da agência e o número de dias (default 7)?

```
CREATE INDEX ix_extrato on cliente (conta, agencia, data );
```

CONSTRAINTS - Restrições

```
CREATE TABLE investimento (
        codInvestimento       NUMBER   PRIMARY KEY  ,
        descricao             VARCHAR2(50) NOT NULL,
        datacriacao           DATE         DEFAULT (sysdate),
        liquidacaodias        NUMBER(3)    DEFAULT 0,
        aplicacaominima       NUMBER(9,2)  CHECK (aplicacaominima > 100),
        taxaadministracao     NUMBER(4,2),
        idanalistaresponsavel NUMBER,
        labelliquidacao       VARCHAR(50)  AS ('D+' || to_char(liquidacaodias)),
   CONSTRAINT pkinvestimento  PRIMARY KEY (codInvestimento),
   CONSTRAINT fkinvest_funcio FOREIGN KEY (idanalistaresponsavel)
      REFERENCES funcionario(idfuncionario))
   TABLESPACE USERS; <<<<<< Conjunto de datafiles que formam uma área segregada
```

```
ALTER TABLE investimento ADD CONSTRAINT ck_taxaadmin
   CHECK (taxaadministracao > 0.5);
```

OBS: CHECK Constraint no escopo de tabela pode referenciar outras colunas!!!!

TESTANDO CONSTRAINTS

```
INSERT INTO investimento (codinvestimento, descricao, liquidacaodias,
                          aplicacaominima, taxaadministracao, idanalistaresponsavel)
    VALUES ( 1, 'Fundo Petrobras', 3, 1000, 1.15, 6);  <<ERRO>>

INSERT INTO funcionario (idfuncionario, nome, idcargo) VALUES (6,'FELIPE CERATI',5);


Se tentar inserir mesma pk novamente:

INSERT INTO investimento (codinvestimento, descricao, liquidacaodias,
                          aplicacaominima, taxaadministracao,idanalistaresponsavel)
    VALUES ( 1, 'Fundo Petrobras', 3, 1000,1.15,6);

ORA-00001: restrição exclusiva (SYS.PKINVESTIMENTO) violada
```

TESTANDO CONSTRAINTS

```
INSERT INTO investimento (codinvestimento, descricao, liquidacaodias,
                          aplicacaominima, taxaadministracao, idanalistaresponsavel)
    VALUES ( 2, 'Fundo CDI MAIS', 3, 1000, 0.4, 5);

ORA-02290: restrição de verificação (SYS.CK_TAXAADMIN) violada


INSERT INTO investimento (codinvestimento, descricao, liquidacaodias,
                          aplicacaominima, taxaadministracao, idanalistaresponsavel)
    VALUES ( 3, 'Tesourdo Direto', 3, 90, 1.15, 5);

ORA-02290: restrição de verificação (ADMIN.SYS_C0027979) violada
```

Mãos na massa!

DICAS

Erro ORA-02438 – Ver Slide sobre constraints

Erro ORA-02270 – Validar se a tabela mãe é primary key. Se não for, deve ser adicionada a PK nela e tentar novamente ☺

- 1 – Criar uma materialized view chamada MVIEWREPORTX e um sinônimos para que todos os usuários consigam acessar ela pelo nome RELATORIOX. Essa Mview será atualizada apenas sob demanda, e deve listar o extrato de todas as contas, ordenados por agencia, conta e data decrescente, nessa ordem. A lista deve conter todas as colunas da tabela transação e também o CPF que deve ser exibido mascarado conforme exemplo: 917.XXX.235-15.
- 2 – A consulta nessa Mview é feita sempre pelo CPF, nesse formato mesmo com máscara. Você deve criar índice direto sobre a Mview para que essa consulta seja rápida.
- 3 – Criar uma nova tabela chamada consorcio que deverá ter as colunas idConsSeq, idcliente, tipo de consórcio, valor do consórcio, data de contratação e data previsão de termino. Essa tabela deverá ter algumas restrições:
  - idConsSeq deve ser um valor gerado automaticamente. Esse valor deve iniciar em 101 e pular de 2 em 2: 103, 105 ..
  - Idcliente deve existir na tabela cliente
  - O campo tipo de consórcio só pode permitir os caracteres M ou C (Moto e Carro respectivamente)
  - A data de contratação deve ser sempre menor que a data de previsão de término do contrato.

AGENDA

Módulo 1

Suite Oracle; Conceitos; Tipos de dados; Tipos de Comandos; Comandos SQL básicos; Operadores Condicionais e Lógicos; Ordenação de resultados. **+DESAFIO**

Módulo 2

Comandos INSERT, DELETE e UPDATE; Controle de Transações; Consolidação de Dados; Principais Funções SQL. **+DESAFIO**

Módulo 3

Subconsultas; JOINS - Consultas com mais de uma tabela; Modelo de Dados - Diagrama ER. **+DESAFIO**

Módulo 4

Schemas; Sinônimos; Views; Materialized Views; SEQUENCESs; PROCEDUREs; FUNCTIONs; TRIGGERs; ÍNDICES; CONSTRAINTS. **+DESAFIO**

Extras

SQL Injection; Database Link; Oracle Golden Gate; SQL Loader; Tabelas Particionadas; Métodos de Conexão ao Oracle

EXTRAS

- Comandos SQL Plus:

```
conn user/senha@alias ou string connection
```

- Seta formato de exibição de data:

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MM-YYYY HH24:MI:SS';
```

- Formata a coluna para exibir com largura de 20 caracteres:

```
col nomecoluna format a20
```

- Formata a coluna coluna para exibir número com essa máscara:

```
col nomecoluna format 999999
```

- Define tamanho da linha em caracteres:

```
set linesize 4000
```

- Define a quebra de páginas em retorno de comandos de select:

```
set pagesize 4000
```

- Exibe o último comando executado:

```
L
```

SQL INJECTION

varLogin = payload.Login <<< joao_mino

varSenha = payload.Senha <<< 12345' or '1' = '1

strMontaSQL = "Select 1 from tabela usuario where login = '" +

varLogin + "' and senha = '" + varSenha + "'"

? strMontaSQL

Select 1 from tabela usuario

where login = 'joao_mino' and senha ='12345' or '1' = '1'

DATABASE LINKS – Comunicação entre bancos


```
CREATE DATABASE LINK bancocloud
CONNECT TO CONSULTA_AG_CLOUDUSER IDENTIFIED BY Agencias$908070#
USING 'dbciti_high';
```

```
SELECT * FROM agencias@bancocloud;
```

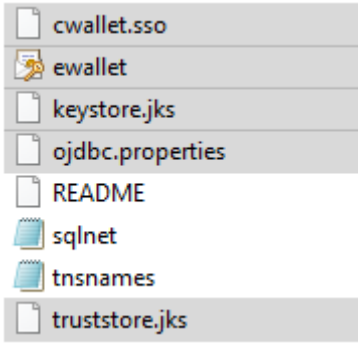
Evitar uso transacional! Priorizar sempre uma arquitetura com comunicação entre microsserviços!

DATABASE LINKS – Configurar TNS

Confirmar caminho dos arquivos de configuração de conexões do oracle:

```
Tnsping XEPDB1
```

Descompactar em outro diretório os arquivos de conexão para banco OCI e copiar os seguintes arquivos:



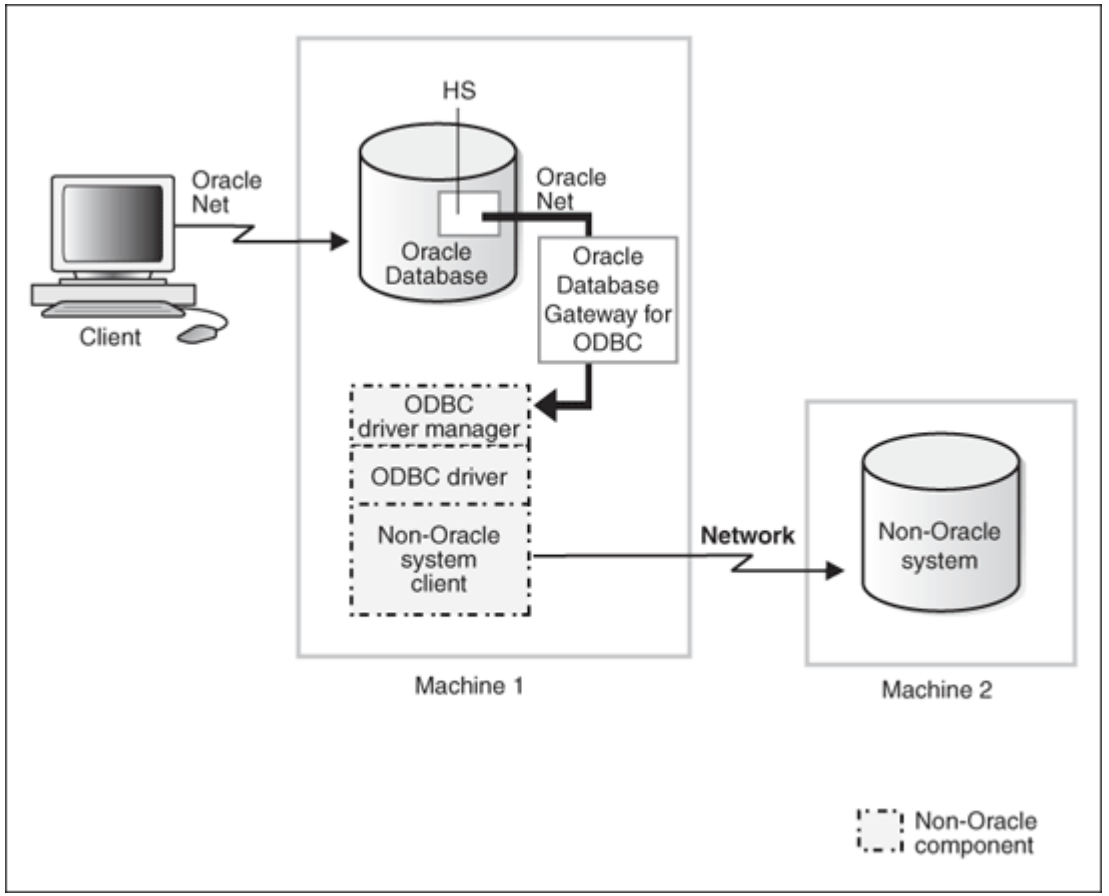
Inserir as linhas abaixo no arquivo sqlnet.ora da instalação:

WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA = (DIRECTORY="?/network/admin")))
SSL_SERVER_DN_MATCH=yes

Copiar o conteúdo do arquivo tnsnames.ora para dentro do tnsnames.ora da instalação

Testar conexão no banco em CLOUD: sqlplus consulta_ag_clouduser/Agencias$908070#@dbciti_high

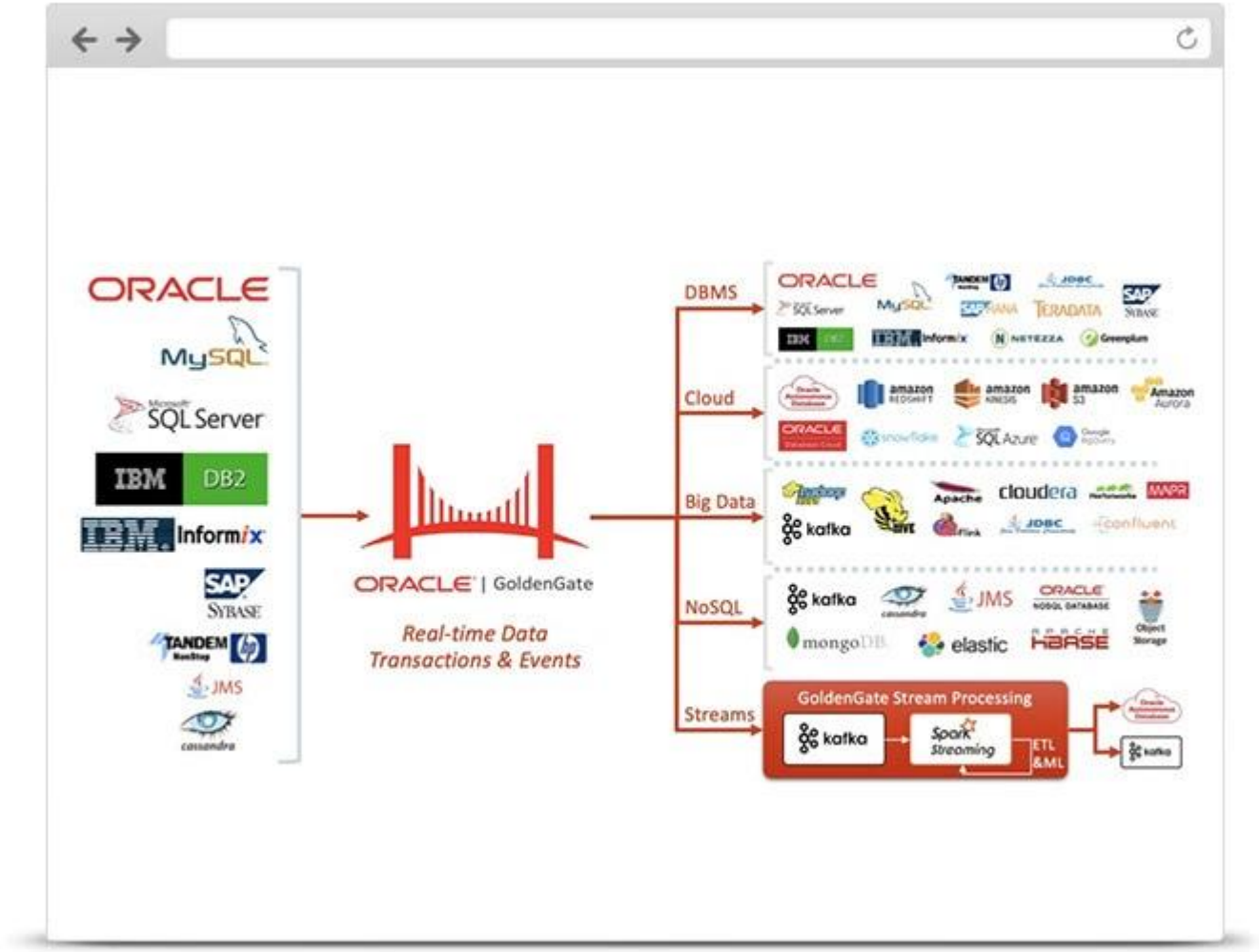
É possível conectar em bancos diferentes de Oracle?



Oracle Database Gateway

https://docs.oracle.com/en/database/oracle/oracle-database/12.2/odbcu/database-gateway-for-odbc-introduction.html

Oracle Golden Gate – (ETL – Extract, Transform and Load)



https://www.oracle.com/a/ocom/img/cw20v1-goldengate-big-data-1.jpg

SQL Loader

Arquivo importa.ctl

```
load data
 infile 'importaprodutos.txt'
 into table sys.agencia_produto_new
 fields terminated by ","
 ( idagencia, produto )
```

Arquivo importaprodutos.txt

```
0003,CARTAO MASTER
0003,CARTAO VISA
0003,CHEQUE ESPECIAL
0003,CREDITO PESSOAL
0003,FINANCIAMENTO
0003,PIX
0003,SEGURO RESIDENCIAL
0003,INVESTIMENTO
```

```
sqlldr usercarga/U$ercarga2022@XEPDB1 control=importa.ctl
```

Revisar se o arquivo tnsnames.ora tem a entrada XEPDB1

Tabelas Particionadas

Conjunto de partições físicas para quebrar os dados em objetos menores.

```
create table logerrosapp (  idlog                              number,
                            descricao                          varchar(4000),
                            datalog                            DATE not null )
                  partition by range(datalog)
                  interval (numtoyminterval(1,'MONTH'))
                 (PARTITION T0 VALUES LESS THAN (TO_DATE('01/01/2022',
'DD/MM/YYYY')));

insert into logerrosapp
select 1,'Erro1',to_timestamp('01012022 08:00','ddmmyyyy hh24:mi') from dual
union
select 2,'Erro2',to_timestamp('01022022 08:00','ddmmyyyy hh24:mi') from dual
union
select 3,'Erro3',to_timestamp('01032022 08:00','ddmmyyyy hh24:mi') from dual;

select PARTITION_NAME, SEGMENT_TYPE
  from dba_segments where segment_name = 'LOGERROSAPP';
```

Exemplos de URIs

- jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=hostname)(PORT=1531)))(CONNECT_DATA=(SERVICE_NAME=servicename)(SERVER=DEDICATED)))

- jdbc:oracle:thin:@hostname:1521/servicename

- jdbc:oracle:thin:@(description=(retry_count=20)(retry_delay=3)(address=(protocol=tcps)(port=1522)(host=adb.us-ashburn-1.oraclecloud.com))(connect_data=(service_name=kjsa8j5g3hfa1_dbciti_high.adb.oraclecloud.com))(security=(ssl_server_cert_dn="CN=adwc.uscom-east-1.oraclecloud.com, OU=Oracle BMCS US, O=Oracle Corporation, L=Redwood City, ST=California, C=US")))

OBRIGADO!

fdsantos@gmail.com

https://www.linkedin.com/in/fdsantosdba