SQL – Structured Query Language

Oracle 11g

DDL (Data Definition Language)

Linguagem de Definição de Dados

CREATE TABLE	Cria uma tabela
ALTER TABLE	Altera uma tabela
DROP TABLE	Elimina uma tabela
CREATE VIEW	Cria uma visão
DROP VIEW	Elimina uma visão
CREATE INDEX	Cria um índice
DROP INDEX	Elimina um índice
TRUNCATE	Corta uma tabela (elimina os dados e mantém a estrutura)

DML (Data Manipulation Language)

Linguagem de Manipulação de Dados

INSERT	Insere linhas na tabela
DELETE	Exclui linhas da tabela
UPDATE	Atualiza as linhas tabela

DQL (Data Query Language)

Linguagem de Consulta de Dados

SELECT	Seleciona (recupera) dados de uma tabela ou visão

DCL (Data Control Language)

Linguagem de Controle de Dados

GRANT	Concede privilégios ao usuário
REVOKE	Revoga (retira) privilégios do usuário

DTL (Data Transaction Language)

Linguagem de Transação de Dados

COMMIT	Grava os dados das mudanças no banco de dados
ROLLBACK	Descarta as mudanças desde o último COMMIT ou ROLLBACK

TIPOS DE DADOS (DATATYPES)

NUMBER
Valores numéricos. Tamanho máximo 38.

NUMBER(4) armazena valores até 9999
NUMBER(5,2) armazena valores até 999.99

CHAR
Cadeia de caracteres de tamanho fixo. O default é 1 e o máximo é 2000.

VARCHAR2
Cadeia de caracteres de tamanho variável. Tamanho máximo 4.000.

DATE
Datas entre 01/01/4712 AC até 31/12/9999 DC.

RESTRIÇÕES (CONSTRAINTS)

CHAVE PRIMÁRIA (PK – PRIMARY KEY)
Coluna ou grupo de colunas que permite identificar uma única linha da tabela.

CHAVE ESTRANGEIRA (FK – FOREIGN KEY)
Coluna que estabelece o relacionamento entre duas tabelas. Corresponde à chave primária da "tabela-pai".

Cláusulas que podem ser aplicadas às chaves estrangeiras:

ON DELETE SET NULL
Quando for removido um valor da "tabela-pai" o Oracle define os valores correspondentes da "tabela-filho" como NULL.

ON DELETE CASCADE
Quando for removido um valor da "tabela-pai" o Oracle remove as linhas correspondentes da "tabela-filho".

CHAVE ÚNICA (UNIQUE)
Indica que não poderá haver repetição no conteúdo da coluna.

NÃO NULO (NOT NULL)
Indica que o conteúdo de uma coluna não poderá ser nulo (vazio).

CHECK
Define um domínio de valores para o conteúdo da coluna.

DEFAULT
Atribui um conteúdo padrão a uma coluna caso nenhum valor seja informado.

CRIAÇÃO DE TABELAS

```
CREATE TABLE NOME_DA_TABELA (
COLUNA_1 TIPO_DE_DADO(TAMANHO) NOT NULL | CHECK | DEFAULT,
COLUNA_1 TIPO_DE_DADO(TAMANHO) NOT NULL | CHECK | DEFAULT,
CONSTRAINT NOME_CONSTRAINT PRIMARY KEY(NOME_DA_COLUNA),
CONSTRAINT NOME_CONSTRAINT FOREIGN KEY(NOME_DA_COLUNA)
REFERENCES NOME_DA_TABELA_PAI(NOME_DA_COLUNA_NA_TABELA_PAI));
```

Nota: *Quando o nome da coluna na "tabela-filho" for idêntico ao da "tabela-pai" não há necessidade de informá-lo no final da criação da chave estrangeira:*

```
...
CONSTRAINT NOME_CONSTRAINT FOREIGN KEY(NOME_DA_COLUNA)
REFERENCES NOME_DA_TABELA_PAI);
```

CLIENTE				
COD_CLIENTE	**NOME**	**UF**	**DATA_NASC**	**CPF**
NUMBER(4)	VARCHAR2(50)	CHAR(2)	DATE	CHAR(11)
PRIMARY KEY	NOT NULL	CHECK ('SP', 'RJ','MG')		UNIQUE

```
CREATE TABLE CLIENTE (
COD_CLIENTE NUMBER(4),
NOME VARCHAR2(50) NOT NULL,
UF CHAR(2) CHECK (UF IN ('SP', 'RJ','MG')),
DATA_NASC DATE,
CPF CHAR(11) UNIQUE,
CONSTRAINT CLIENTE_PK PRIMARY KEY(COD_CLIENTE));
```

PEDIDO			
NR_PEDIDO	**DATA_EMISSAO**	**VALOR**	**COD_CLIENTE**
NUMBER(5)	DATE	NUMBER(6,2)	NUMBER(4)
PRIMARY KEY	DEFAULT SYSDATE	NOT NULL	FOREIGN KEY

```
CREATE TABLE PEDIDO (
NR_PEDIDO NUMBER(4),
DATA_EMISSAO DATE DEFAULT SYSDATE,
VALOR NUMBER(6,2) NOT NULL,
COD_CLIENTE NUMBER(4) NOT NULL,
CONSTRAINT PEDIDO_PK PRIMARY KEY(NR_PEDIDO),
CONSTRAINT PEDIDO_CLIENTE_FK FOREIGN KEY(COD_CLIENTE)
REFERENCES CLIENTE (COD_CLIENTE));
```

Para apresentar a estrutura da tabela utilize:
`DESC NOME_DA_TABELA;`

Para apresentar TODAS as tabelas do usuário utilize:
`SELECT TABLE_NAME FROM USER_TABLES;`

RENOMEAR TABELAS

```
RENAME NOME_DA_TABELA TO NOVO_NOME_DA_TABELA;
```

ELIMINAÇÃO DE TABELAS

```
DROP TABLE NOME_DA_TABELA;
```

ELIMINAR A TABELA E AS CONSTRAINTS

```
DROP TABLE NOME_DA_TABELA CASCADE CONSTRAINTS;
```

ALTERAÇÃO DE TABELAS

ADICIONAR UMA COLUNA

```
ALTER TABLE NOME_DA_TABELA
ADD NOME_DA_COLUNA TIPO_DE_DADO(TAMANHO);
```

ALTERAR UMA COLUNA

```
ALTER TABLE NOME_DA_TABELA
MODIFY NOME_DA_COLUNA TIPO_DE_DADO(NOVO_TAMANHO);
```

RENOMEAR UMA COLUNA

```
ALTER TABLE NOME_DA_TABELA
RENAME COLUMN NOME_DA_COLUNA TO NOVO_NOME_DA_COLUNA;
```

ELIMINAR UMA COLUNA

```
ALTER TABLE NOME_DA_TABELA
DROP COLUMN NOME_DA_COLUNA;
```

ADICIONAR UMA CONSTRAINT (CHAVE PRIMÁRIA)

ALTER TABLE NOME_DA_TABELA
ADD CONSTRAINT NOME_DA_CONSTRAINT PRIMARY KEY(NOME_DA_COLUNA);

ADICIONAR UMA CONSTRAINT (CHAVE ESTRANGEIRA)

ALTER TABLE NOME_DA_TABELA
ADD CONSTRAINT NOME_DA_CONSTRAINT FOREIGN KEY(NOME_DA_COLUNA)
REFERENCES NOME_DA_TABELA_PAI(NOME_DA_COLUNA_NA_TABELA_PAI);

ELIMINAR UMA CONSTRAINT

ALTER TABLE NOME_DA_TABELA
DROP CONSTRAINT NOME_DA_CONSTRAINT;

DESABILITAR UMA CONSTRAINT

ALTER TABLE NOME_DA_TABELA DISABLE CONSTRAINT NOME_DA_CONSTRAINT;

HABILITAR UMA CONSTRAINT

ALTER TABLE NOME_DA_TABELA ENABLE CONSTRAINT NOME_DA_CONSTRAINT;

Nota: *Desabilitar uma CONSTRAINT, inserir valores que violem a restrição de integridade e tentar habilitar a restrição posteriormente com o comando ENABLE provocará um erro.*
Até a versão 11g não há possibilidade de incluir a cláusula ON UPDATE CASCADE em uma restrição do tipo Foreign Key. Portanto, para alterar os valores em colunas Primary Key que contém referencias em tabelas-filho é preciso desabilitar a restrição FK na tabela-filho, alterar os valores na coluna PK na tabela-pai, alterar (se necessário) os valores na coluna FK na tabela-filho e habilitar novamente a restrição Foreign Key.

TRUNCATE

Comando DDL utilizado para 'cortar' uma tabela.

TRUNCATE TABLE NOME_DA_TABELA;

Nota: *Os dados não poderão ser recuperados (a não ser através do backup).*

ÍNDICES

Fornecem acesso mais rápido a linhas específicas, eliminando a necessidade de varreduras completas de tabelas.

CRIAR UM ÍNDICE

CREATE INDEX NOME_DO_INDICE ON NOME_DA_TABELA (NOME_DA_COLUNA);

ELIMINAR UM ÍNDICE

DROP INDEX NOME_DO_INDICE;

INSERT

Para inserir uma nova linha na tabela:

INSERT INTO NOME_DA_TABELA (NOME_DA COLUNA_1,NOME_DA_COLUNA_2)
VALUES (VALOR_1,VALOR_2);

INSERT INTO NOME_DA_TABELA VALUES (VALOR_1,VALOR_2);

Exemplo:

INSERT INTO CLIENTE (COD_CLI,NOME,UF,DATA_NASC,CPF)
VALUES (1001,'ANTONIO ALVES','SP','15/12/1980','111222333444');

Nota: *Os tipos CHAR, VARCHAR2 e DATE devem ser declarados entre ' ' (aspas simples).*

DELETE

Para excluir todas as linhas da tabela:

DELETE FROM NOME_DA_TABELA;

Para excluir linhas específicas utilize a cláusula WHERE:

DELETE FROM NOME_DA_TABELA WHERE NOME_DA_COLUNA = VALOR;

Exemplo:

DELETE FROM CLIENTE WHERE COD_CLIENTE = 1001;

UPDATE

Para atualizar uma linha da tabela:

UPDATE NOME_DA_TABELA SET NOME_DA_COLUNA = NOVO_VALOR
WHERE NOME_DA_COLUNA_DE_REFERENCIA = VALOR;

Nota: *É recomendável que a COLUNA_DE_REFERENCIA seja a Chave Primária.*

Exemplo:

UPDATE CLIENTE SET NOME = 'ANTONIO ALVARES' WHERE COD_CLI = 1001;

COMMIT E ROLLBACK

Utilize **COMMIT** para confirmar as transações no banco de dados ou **ROLLBACK** para descartá-las.

SELECT

Para apresentar todos os dados de uma tabela:

SELECT * FROM NOME_DA_TABELA;

Para apresentar os dados de uma determinada linha utilize a cláusula WHERE:

SELECT * FROM NOME_DA_TABELA WHERE NOME_DA_COLUNA = VALOR;

Para apresentar os dados de determinadas colunas:

SELECT NOME_DA_COLUNA1, NOME_DA_COLUNA_2 FROM NOME_DA_TABELA;

SELECT NOME_DA_COLUNA1, NOME_DA_COLUNA_2 FROM NOME_DA_TABELA
WHERE NOME_DA_COLUNA = VALOR;

Exemplo:

SELECT NOME,CPF FROM CLIENTE WHERE COD_CLIENTE = 1001;

Cláusula WHERE

Permite que sejam especificadas linhas sobre as quais será aplicada determinada instrução. É sempre seguida por uma expressão lógica que pode conter os seguintes operadores:

TIPO	OPERADORES
COMPARAÇÃO	= , < , <=, >, >=, <>
LÓGICO	AND, OR, NOT
SQL	IS NULL, LIKE, IN, BETWEEN, EXISTS

OPERADORES SQL	
IS NULL	Verifica se o conteúdo da coluna é NULL (vazio)
IS NOT NULL	Negação do operador IS NULL
LIKE	Compara cadeia de caracteres utilizando padrões de comparação: % substitui zero, um ou mais caracteres _ substitui um caractere LIKE 'A%' inicia com a letra A LIKE '%A' termina com a letra A LIKE '%A%' tem a letra A em qualquer posição LIKE 'A_' string de dois caracteres, inicia com a letra A LIKE '_A' string de dois caracteres, termina com a letra A LIKE '_A_' string de três caracteres, letra A na segunda posição LIKE '%A_' tem a letra A na penúltima posição LIKE '_A%' tem a letra A na segunda posição
NOT LIKE	Negação do operador LIKE
IN	Verifica se um valor pertence a um conjunto de valores
NOT IN	Negação do operador IN
BETWEEN	Determina um intervalo de busca: BETWENN 'valor1' AND 'valor2'
NOT BETWEEN	Negação do operador BETWEEN
EXISTS	Verifica se um valor existe em um conjunto, levando em conta os valores nulos. Fato não considerado pelo operador IN
NOT EXISTS	Negação do operador EXISTS

ALIAS

Utilizado para dar apelidos aos nomes de colunas.

SELECT NOME **NOME_CLIENTE,** DATA_NASC **"DATA NASCIMENTO"** FROM CLIENTE;

Utilizado para dar apelidos aos nomes de tabelas.

SELECT **C**.NOME**, C**.DATA_NASC FROM CLIENTE **C**;

DISTINCT

Utilizado para não repetir dados em uma consulta.

SELECT **DISTINCT** NOME_DA_COLUNA FROM NOME_DA_TABELA;

Exemplo:

SELECT **DISTINCT** UF FROM CLIENTE;

ORDER BY

Utilizado para ordenar os dados em uma consulta.

Ordem ascendente (A-Z):

SELECT NOME_DA_COLUNA FROM NOME_DA_TABELA
ORDER BY NOME_DA_COLUNA ASC;

Exemplo:

SELECT NOME FROM CLIENTE **ORDER BY** NOME ASC;

Nota: *ASC é padrão, pode ser omitido.*

Ordem descendente (Z-A):

SELECT NOME_DA_COLUNA FROM NOME_DA_TABELA
ORDER BY NOME_DA_COLUNA DESC;

Exemplo:

SELECT NOME FROM CLIENTE **ORDER BY** NOME DESC;

GROUP BY

Agrupa as linhas selecionadas e retorna uma única linha para cada grupo.

A consulta abaixo retorna a soma dos valores dos pedidos agrupados por cliente:

```
SELECT SUM(VALOR), COD_CLIENTE FROM CLIENTE GROUP BY COD_CLIENTE;
```

Nota: *Para outras opções consulte o tópico FUNÇÕES DE GRUPO.*

Cláusula HAVING

Seleciona **grupos** de linhas que satisfazem determinado critério.

A consulta abaixo retorna a soma dos valores dos pedidos do cliente cujo COD_CLIENTE é igual a 1001:

```
SELECT SUM(VALOR), COD_CLIENTE FROM CLIENTE
GROUP BY COD_CLIENTE HAVING COD_CLIENTE = 1001;
```

Nota: *Quando ocorre o agrupamento de linhas a cláusula WHERE não pode ser utilizada. Deve-se utilizar, conforme observado, a cláusula HAVING.*

FUNÇÕES

FUNÇÕES DE GRUPO

SUM Retorna a somatória de uma coluna ou expressão.

```
SELECT SUM(COLUNA_OU_EXPRESSÃO)
FROM NOME_DA_TABELA GROUP BY NOME_DA_COLUNA;
```

Para obter o valor total dos pedidos emitidos por cada cliente:

```
SELECT COD_CLIENTE,SUM(VALOR)
FROM PEDIDO GROUP BY COD_CLIENTE;
```

AVG Retorna a média aritmética de uma coluna ou expressão.

```
SELECT AVG(COLUNA_OU_EXPRESSÃO)
FROM NOME_DA_TABELA GROUP BY NOME_DA_COLUNA;
```

Para obter o valor médio dos pedidos emitidos por cada cliente:

```
SELECT COD_CLIENTE,AVG(VALOR)
FROM PEDIDO GROUP BY COD_CLIENTE;
```

MAX Retorna o maior valor de uma coluna ou expressão.

```
SELECT MAX(COLUNA_OU_EXPRESSÃO)
FROM NOME_DA_TABELA GROUP BY NOME_DA_COLUNA;
```

Para obter o valor do maior pedido emitido por cada cliente:

```
SELECT COD_CLIENTE,MAX(VALOR)
FROM PEDIDO GROUP BY COD_CLIENTE;
```

MIN Retorna o menor valor de uma coluna ou expressão.

```
SELECT MIN(COLUNA_OU_EXPRESSÃO)
FROM NOME_DA_TABELA GROUP BY NOME_DA_COLUNA;
```

Para obter o valor do menor pedido emitido por cada cliente:

```
SELECT COD_CLIENTE,MIN(VALOR)
FROM PEDIDO GROUP BY COD_CLIENTE;
```

COUNT Retorna o número de vezes que o argumento mudou de valor.

```
SELECT COUNT(COLUNA_OU_EXPRESSÃO)
FROM NOME_DA_TABELA GROUP BY NOME_DA_COLUNA;
```

Para obter a quantidade de pedidos emitidos por cada cliente:

```
SELECT COD_CLIENTE,COUNT(VALOR)
FROM PEDIDO GROUP BY COD_CLIENTE;
```

FUNÇÕES DE LINHA

UPPER Retorna todas as letras em maiúsculas.

```
SELECT UPPER(COLUNA_OU_VALOR) FROM NOME_DA_TABELA;
SELECT UPPER('tEstE') FROM DUAL; -- RETORNA TESTE
```

Para alterar todos os valores de uma coluna para maiúsculas:

```
UPDATE TESTE SET NOME = UPPER(NOME);
```

Para pesquisar valores em colunas quando não se tem certeza da forma como foram armazenados (em maiúsculas, minúsculas, etc.):

```
SELECT * FROM NOME_DA_TABELA
WHERE UPPER(NOME_DA_COLUNA) = 'VALOR_EM_MAIUSCULAS';
```

LOWER Retorna todas as letras em minúsculas.

```
SELECT LOWER(COLUNA_OU_VALOR) FROM NOME_DA_TABELA;
SELECT LOWER('tEstE') FROM DUAL; -- RETORNA teste
```

INITCAP Retorna todas as primeiras letras de cada palavra em maiúsculas.

```
SELECT INITCAP(COLUNA_OU_VALOR) FROM NOME_DA_TABELA;
SELECT INITCAP('tEstE') FROM DUAL; -- RETORNA Teste
```

LPAD Preenchimento à esquerda.

```
SELECT LPAD(COLUNA_OU_VALOR,TAMANHO,'CARACTERE')
FROM NOME_DA_TABELA;
SELECT LPAD('TESTE',10,'*') FROM DUAL; -- RETORNA *****TESTE
```

RPAD Preenchimento à direita.

```
SELECT RPAD(COLUNA_OU_VALOR, TAMANHO,'CARACTERE')
FROM NOME_DA_TABELA;
SELECT RPAD('TESTE',10,'*') FROM DUAL; -- RETORNA TESTE*****
```

SUBSTR Retorna uma "substring".

```
SELECT(COLUNA_OU_VALOR,POSIÇÃO_INICIAL,TAMANHO)
FROM NOME_DA_TABELA;
SELECT SUBSTR('TESTE',1,3) FROM DUAL; -- RETORNA TES
SELECT(COLUNA_OU_VALOR,TAMANHO_A_PARTIR_DO_FINAL)
FROM NOME_DA_TABELA;
SELECT SUBSTR('TESTE', 3) FROM DUAL; -- RETORNA STE
```

FUNÇÕES NUMÉRICAS

ABS Retorna o valor absoluto do número do argumento.

```
SELECT ABS(-5) FROM DUAL; -- RETORNA 5
```

CEIL Retorna próximo número inteiro maior ou igual ao número do argumento.

```
SELECT CEIL(5.25) FROM DUAL; -- RETORNA 6
```

FLOOR Retorna próximo número inteiro menor ou igual ao número do argumento.

```
SELECT FLOOR(5.25) FROM DUAL; -- RETORNA 5
```

ROUND Retorna o número com arredondamento.

```
SELECT ROUND(5.25,1) FROM DUAL; -- RETORNA 5.3
```

TRUNC Retorna o número truncado (cortado).

```
SELECT TRUNC(5.25,1) FROM DUAL; -- RETORNA 5.2
```

MOD Retorna o resto da divisão.

```
SELECT MOD(5,2) FROM DUAL; -- RETORNA 1
```

POWER Retorna o valor da potenciação.

```
SELECT POWER(5,2) FROM DUAL; -- RETORNA 25
```

SQRT Retorna a raiz quadrada.

```
SELECT SQRT(25) FROM DUAL; -- RETORNA 5
```

SIGN Retorna 1 para valores maiores que 0 (zero), 0 (zero) quando o valor for 0 (zero) e -1 quando o valor for menor que 0 (zero).

```
SELECT SIGN(-5) FROM DUAL; -- RETORNA -1
```

FUNÇÕES DE CONVERSÃO

TO_CHAR Converte um valor tipo DATE ou NUMBER para um valor CHAR.

Para retornar o dia (conforme a data do sistema):

```
SELECT TO_CHAR(SYSDATE,'DD') FROM DUAL;
```

Para retornar o mês por extenso (conforme a data do sistema):

```
SELECT TO_CHAR(SYSDATE,'MONTH') FROM DUAL;
SELECT TO_CHAR(1250,'999,999.99') FROM DUAL;
-- RETORNA 1,250.00
```

TO_DATE Converte uma expressão tipo CHAR para DATE.

```
INSERT INTO TESTE (DATA_NASC)
VALUES (TO_DATE('10/20/1980','MM/DD/YYYY'));
```

Nota: *Utilize a função TO_DATE para inserir datas em tabelas quando não souber o formato utilizado no servidor (DD/MM/YYYY ou MM/DD/YYYY).*

TO_NUMBER Converte uma string (válida) em um valor NUMBER.

```
SELECT TO_NUMBER('$1,250.00','$999,999.99') FROM DUAL;
-- RETORNA 1250
```

OUTRAS FUNÇÕES

NVL Retorna um valor não nulo quando o termo nulo for encontrado.

Para retornar 0 (zero) quando o termo nulo for encontrado:

```
SELECT NVL(NOME_DA_COLUNA,0) FROM NOME_DA_TABELA;
```

NULLIF Retorna NULL se os dois valores forem iguais, senão retorna o primeiro valor.

```
SELECT NULLIF(15,15) FROM DUAL; -- RETORNA NULL
SELECT NULLIF(15,20) FROM DUAL; -- RETORNA 15
```

DECODE Implementa a lógica condicional if-then-else.

```
SELECT DECODE(15,15,'IGUAIS','DIFERENTES') FROM DUAL;
-- RETORNA IGUAIS
SELECT DECODE(15,20,'IGUAIS','DIFERENTES') FROM DUAL;
-- RETORNA DIFERENTES
```

CASE

Implementa uma estrutura de controle. Há dois tipos de construções:

TIPO 1

```
SELECT UF,
CASE UF
WHEN 'SP' THEN 'SÃO PAULO'
WHEN 'RJ' THEN 'RIO DE JANEIRO'
ELSE 'OUTRO'
END
FROM ESTADO;
```

TIPO 2

```
SELECT COD_PRODUTO, VALOR,
CASE
WHEN VALOR < 10.00 THEN VALOR * 0.9
WHEN VALOR >= 15.00 THEN VALOR * 0.8
ELSE VALOR * 0.7
END VALOR_PROMOCIONAL
FROM PRODUTO;
```

As tabelas a seguir serão utilizadas para os tópicos SUBQUERIES e JOINS:

CLIENTE				
COD_CLIENTE	**NOME**	**UF**	**DATA_NASC**	**CPF**
1001	ANTONIO ALVARES	SP	20/10/1980	11122233344
1002	BEATRIZ BERNARDES	RJ	15/05/1982	22233344455
1003	CLAUDIO CARDOSO	SP	10/06/1981	33344455566
1004	DANIELA DANTAS	RJ	25/01/1982	44455566677

PEDIDO			
NR_PEDIDO	**DATA_EMISSAO**	**VALOR**	**COD_CLIENTE**
1	10/10/2011	1500.00	1002
2	11/10/2011	2800.00	1001
3	11/10/2011	1200.00	1004
4	14/10/2011	1700.00	

CLASSE_PEDIDO		
CLASSE	**VALOR_MINIMO**	**VALOR_MAXIMO**
A	1000.00	1500.00
B	1500.01	2000.00
C	2000.01	3000.00

CURSO		
COD_CURSO	**NOME_CURSO**	**COD_PRE_REQUISITO**
11	WORD BÁSICO	
12	WORD AVANÇADO	11
51	EXCEL BÁSICO	
52	EXCEL AVANÇADO	51

SUBQUERIES (SUBCONSULTAS)

Consulta dentro de um comando SQL (SELECT, UPDATE, DELETE ou INSERT).

Apresentar os números dos pedidos emitidos pelo cliente 'ANTONIO ALVARES':

```
SELECT NR_PEDIDO FROM PEDIDO WHERE COD_CLIENTE =
(SELECT COD_CLIENTE FROM CLIENTE WHERE NOME = 'ANTONIO ALVARES');
```

Para consultas que possam retornar mais de uma linha utilize o operador **IN**:

```
SELECT NR_PEDIDO FROM PEDIDO WHERE COD_CLIENTE IN
(SELECT COD_CLIENTE FROM CLIENTE WHERE NOME LIKE '%A%');
```

JOINS (JUNÇÕES)

EQUI JOIN

Reúne campos iguais de tabelas diferentes.

Apresentar os nomes dos clientes e os números dos seus respectivos pedidos:

```
SELECT C.NOME, P.NR_PEDIDO
FROM CLIENTE C
INNER JOIN PEDIDO P
ON C.COD_CLIENTE = P.COD_CLIENTE;
```

OUTER JOIN

Além de mostrar registros cujos campos em comum estejam presentes nas duas tabelas, mostra também os que faltam.

Apresentar os nomes dos clientes e os números dos seus respectivos pedidos (incluir os nomes dos clientes que não efetuaram pedidos):

```
SELECT C.NOME, P.NR_PEDIDO
FROM CLIENTE C
LEFT OUTER JOIN PEDIDO P
ON C.COD_CLIENTE = P.COD_CLIENTE;
```

Apresentar os nomes dos clientes e os números dos seus respectivos pedidos (incluir os pedidos sem o código de cliente):

```
SELECT C.NOME, P.NR_PEDIDO
FROM CLIENTE C
RIGHT OUTER JOIN PEDIDO P
ON C.COD_CLIENTE = P.COD_CLIENTE;
```

Apresentar os nomes dos clientes e os números dos seus respectivos pedidos (incluir os nomes dos clientes que não efetuaram pedidos e os pedidos sem o código do cliente):

```
SELECT C.NOME, P.NR_PEDIDO
FROM CLIENTE C
FULL OUTER JOIN PEDIDO P
ON C.COD_CLIENTE = P.COD_CLIENTE;
```

NON EQUI JOIN

Reúne campos de tabelas que não têm valores em comum.

Classificar os pedidos (em A, B ou C) de acordo com seus valores.

```
SELECT P.NR_PEDIDO, CP.CLASSE
FROM PEDIDO P, CLASSE_PEDIDO CP
WHERE P.VALOR BETWEEN CP.VALOR_MINIMO AND CP.VALOR_MAXIMO;
```

SELF JOIN

Relaciona dois campos de uma mesma tabela que sejam do mesmo tipo.

Apresentar os nomes dos cursos que têm pré-requisitos e os nomes dos respectivos cursos que são pré-requisitos.

```
SELECT C.NOME_CURSO, P.NOME "PRE-REQUISITO"
FROM CURSO C
INNER JOIN CURSO P
ON C.COD_PRE_REQUISITO = P.COD_CURSO;
```

CROSS JOIN

Apresenta todas as combinações possíveis entre elementos de duas tabelas.

```
SELECT T1.NOME_DA_COLUNA, T2.NOME_DA_COLUNA
FROM TABELA_1 T1
CROSS JOIN TABELA_2 T2;
```

NATURAL JOIN

Utilizada em consultas que envolvem tabelas que apresentam colunas com o mesmo nome. Não é necessário explicitar o nome das colunas.

```
SELECT NOME, NR_PEDIDO
FROM CLIENTE
NATURAL INNER JOIN PEDIDO;
```

OPERAÇÕES DE CONJUNTO

UNION (UNIÃO)

Apresenta como resultado a união de todas as linhas que foram recuperadas por dois comandos SQL realizados em separado.

Os comandos devem retornar o mesmo número de colunas.
As colunas correspondentes devem possuir os mesmos tipos de dados.

```
SELECT COLUNA_1, COLUNA_2 FROM TABELA_1
UNION [ALL]
SELECT COLUNA_3, COLUNA_4 FROM TABELA_2;
```

O predicado ALL fará com que apareçam linhas repetidas (se for o caso).

INTERSECT (INTERSEÇÃO)

Apresenta como resultado a interseção de todas as linhas que foram recuperadas por dois comandos SQL realizados em separado.

Os comandos devem retornar o mesmo número de colunas.
As colunas correspondentes devem possuir os mesmos tipos de dados.

```
SELECT COLUNA_1, COLUNA_2 FROM TABELA_1
INTERSECT
SELECT COLUNA_3, COLUNA_4 FROM TABELA_2;
```

MINUS (DIFERENÇA)

Apresenta como resultado a diferença entre as linhas que foram recuperadas por dois comandos SQL realizados em separado.

A ordem de declaração das consultas com relação ao predicado MINUS altera o resultado final.

Os comandos devem retornar o mesmo número de colunas.
As colunas correspondentes devem possuir os mesmos tipos de dados.

```
SELECT COLUNA_1, COLUNA_2 FROM TABELA_1
MINUS
SELECT COLUNA_3, COLUNA_4 FROM TABELA_2;
```

Outros SGBDs utilizam EXCEPT (em vez de MINUS) para esta operação.