

AULA 12/03:

Modelo Relacional - Para transações

Modelo Analítico - Para consultas

Data Mining - basicamente um relatório para melhorar negócios

Modelo Analítico:

Data Warehouse - totalização de informações úteis para gerência

Data Mart - subconjuntos do Data Warehouse

AULA 19/03:

Modelos Sem Esquemas Prévios:

- É modelado de uma forma não-tabular, seu esquema não precisa ser definido.
- Para aplicações que precisam lidar com grande volume de dados, exigem alto desempenho e escalabilidade horizontal. (Criar tabelas antes no MySQL é um exemplo de modelo tabular)

Exemplos e Tipos de Modelo NoSQL:

- **Chave Valor:**

“título” : “Esse é o título”

“numero_de_pgs” : “123”

- **Orientado a Colunas:**

(linha,coluna,timestamp)

Permite identificar diferentes versões de um mesmo dado

- **Orientado a Documentos:**

Coleção de Documentos

Um código único e um conjunto de campos

Prós:

Sem esquema prévio

Facilidade

Lida com Grandes Volumes (Big Data)

Contras:

Redundância

Sem Join

Sem Transações

Cenário De Uso:

Big Data

Escrita intensa

Busca simples, porém pesada

Alta escada e disponibilidade

Schema instável.

AULA 26/03 e 09/04:

Modelo Relacional: Linguagem padrão SQL, padrão ANSI

Conceito de domínio e relação

- **Domínio:** Possíveis valores para cada um dos atributos, cada atributo tem um domínio específico. Ex: “Texto, tamanho 200, obrigatório”. Dado uma característica, o domínio é o conjunto de número de matrículas possíveis.
- **Relação:** Uma coluna com outra. Ex: “Título” com “Texto, tamanho 200, obrigatório”.

O SQL pode ser dividido em 5 conjuntos de comando:

<i>Recuperação de dados</i>	Comando SELECT
<i>Linguagem de definição de dados (DDL - Data Definition Language):</i>	Comandos para criação e manutenção de objetos do banco de dados: CREATE, ALTER, DROP, RENAME e TRUNCATE
<i>Linguagem de manipulação de dados (DML - Data Manipulation Language):</i>	Comandos para inserções (INSERT), atualizações (UPDATE) e exclusões (DELETE)
<i>Linguagem para controle de transações</i>	Comandos COMMIT, ROLLBACK e SAVEPOINT
<i>Linguagem para controle de acesso a dados</i>	Comandos GRANT e REVOKE

Se dermos o comando commit ele se reflete no banco de dados, na disciplina foi configurado um “set commit on”, mas não é assim normalmente.

Resto da aula 26/03 - Professor ensinando comandos básicos de linguagem de definição de dados no Oracle.

COMANDO SELECT:

```
SELECT [] FROM []
```

COMANDO UPDATE:

```
UPDATE inf SET (condição)
```

COMANDO DELETE:

```
DELETE [] WHERE []
```

AULA 16/04 E 23/04

```
CREATE TABLE PESSOAS (
    cpf VARCHAR(20) NOT NULL,
    nome VARCHAR(150) NOT NULL,
    idade NUMBER(3) NULL,
    endereco VARCHAR(150)
);
```

```
INSERT INTO PESSOAS (cpf, nome, idade, endereco)
VALUES ('32809', 'Maria', 25, 'Rua A, 20');
```

```
INSERT INTO PESSOAS (idade, endereco, cpf, nome)
VALUES (25, 'Rua A, 20', '30599', 'Pedro');
```

```
INSERT INTO PESSOAS (cpf, nome, idade, endereco)
VALUES ('29385', 'Carlos', Null, NULL);
```

```
INSERT INTO PESSOAS (cpf, nome, idade, endereco)
VALUES ('32809', 'Maria', 25, 'Rua A, 20');

INSERT INTO PESSOAS (cpf, nome, idade, endereco)
VALUES ('39582', 'Alice', 80, NULL);

SELECT * FROM PESSOAS;

SELECT * FROM PESSOAS WHERE nome LIKE 'A%';

SELECT * FROM PESSOAS WHERE idade IN (25, 30, 40);

ALTER TABLE PESSOAS
ADD genero CHAR(1);

ALTER TABLE PESSOAS
DROP COLUMN idade;

ALTER TABLE PESSOAS
ADD data_nascimento DATE null;
```

```
INSERT INTO PESSOAS (cpf, nome, endereco, genero, data_nascimento)
VALUES ('29048', 'Roberto', 'Rua D, 80', 'M', DATE '1980-02-03');
```

FORMATO DATE:

- **Manipulando formato DATE no SQL:**

Adicionando na tabela:

```
ALTER TABLE PESSOAS
ADD data_nascimento DATE NULL
```

Inserir antes deve-se colocar **DATE**, utilizado para reconhecer datas, têm o padrão ANSI YYYY-MM-DD.

- **Pegar data atual:**

```
SELECT sysdate FROM dual;
```

Dual retorna uma variável de ambiente, como SELECT exige um FROM o oracle criou a tabela DUAL para satisfazer a condição.

- **Alterar visualização da data:**

```
SELECT TO_CHAR(SYSDATE, 'MONTH, DD, YYYY HH24:MI:SS')
FROM DUAL;
```

- **Alterar formato padrão de data:**

```
ALTER SESSION SET NLS_DATE_FORMAT = 'MONTH-DD-YYYY';
```

INTEGRIDADE DE ENTIDADE:

- **PRIMARY KEY E ALTERNATE KEY:**

Esse é o formato INLINE adicionado diretamente na linha do campo da tabela:

```
CREATE TABLE ALUNOS
(
    nroMatricula  VARCHAR(10)  PRIMARY KEY,
    cpf           VARCHAR(20)   UNIQUE,
    email         VARCHAR(100)  UNIQUE,
    nome          VARCHAR(150)  NOT NULL,
    anoIngresso  NUMBER(4)    NOT NULL,
    endereco     VARCHAR(150)  NULL,
    sexo          CHAR(1)      NOT NULL
);
```

Outra maneira (preferível) é com a chave CONSTRAINT, onde se cria normalmente os campos e depois explica como funcionam as chaves, também é útil pois pode-se dar um nome que aparece no campo KEYS, facilita a pesquisa.

```
CREATE TABLE ALUNOS
(
    nroMatricula  VARCHAR(10)  NOT NULL,
    cpf           VARCHAR(20)   NOT NULL,
    email         VARCHAR(100)  NOT NULL,
    nome          VARCHAR(150)  NOT NULL,
    anoIngresso  NUMBER(4)    NOT NULL,
    endereco     VARCHAR(150)  NULL,
    sexo          CHAR(1)      NOT NULL,

    CONSTRAINT PK_ALUNOS PRIMARY KEY (nroMatricula),
    CONSTRAINT AK1_ALUNOS UNIQUE (cpf),
    CONSTRAINT AK2_ALUNOS UNIQUE (email)
);
```

- **Manipulação da chave CHECK:**

Usada para restringir valores de inserção, como por exemplo, o campo sexo só poder ser preenchido com os caracteres 'F' ou 'M':

```
ALTER TABLE ALUNOS ADD CONSTRAINT CK_sexo
CHECK (sexo IN ('M', 'F'));
```

Ou que o campo “anoIngresso” só possa ser superior ao ano 200:

```
ALTER TABLE ALUNOS ADD CONSTRAINT CK_AnoIngr
CHECK (anoIngresso > 2000);
```

- Para o caso de querer adicionar restrições em um campo que já possui informações inseridas:

O banco não aceita adicionar restrições após inserir informações, para esse caso deve ser feito um update na tabela.

- Relação um para muitos:

Exemplo: Um estado pode ter várias cidades mas uma cidade só pode ter um estado. Notação: 1-n.

DEFINIR FOREIGN KEY:

```
CONSTRAINT FK_CRM FOREIGN KEY(crm) REFERENCES MEDICOS(crm)
```

Problema: A Tabela Estados tem a lista de estados cadastrados, como fazer com que a tabela Cidades só aceite os estados da lista?

Criar a ligação Foreign Key

```
ALTER TABLE CIDADES {  
ADD {  
    CONSTRAINT FK_EST_CID  
    FOREIGN KEY(uf)  
    REFERENCES ESTADOS(uf)  
}}
```

Se quiser deletar um Estado que tenha filhos com UF registrado na tabela de cidades?

Não seria possível retirar o pai sem antes deletar os filhos:

```
DELETE FROM CIDADES  
WHERE uf = RS
```

Depois disso o comando de deletar o estado na lista estaria disponível:

```
DELETE FROM ESTADOS  
WHERE uf = RS
```

Como ver número total de itens inseridos em uma tabela:

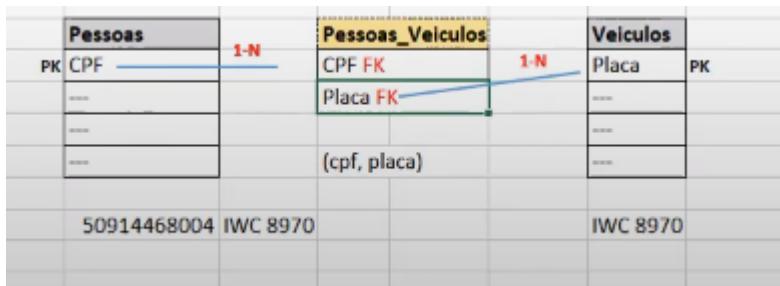
Select count(*) from Exemplo

- Relação muitos para muitos:

Uma pessoa pode ter vários carros e um carro pode ter vários donos.
Notação: n-n.

Como representar essa relação nas tabelas?

Será necessário criar uma terceira tabela



Qual seria a PK de “Pessoas_Veiculos”?

Crie uma chave dupla com CPF e Placa

ATIVIDADES DO DIA 23/04

```
CREATE TABLE MEDICOS (
    crm VARCHAR(50)
    nome VARCHAR(50)
    especializacao VARCHAR(50)
    CONSTRAINT PK_CRM PRIMARY KEY(crm)
)

CREATE TABLE PACIENTES (
    cpf NUMBER(11)
    nome VARCHAR(50)
    sexo VARCHAR(1)
    CONSTRAINT PK_CPF PRIMARY KEY(cpf)
    CONSTRAINT CK_SEXO CHECK (sexo IN('M', 'F'))
)

CREATE TABLE MEDICOS_PACIENTES (
    crm VARCHAR(50)
    cpf VARCHAR(50)
    CONSTRAINT PK_MEDICOPACIENTE PRIMARY KEY(crm, cpf)
    CONSTRAINT FK_CRM FOREIGN KEY(crm) REFERENCES MEDICOS(crm)
    CONSTRAINT FK_CPF FOREIGN KEY(cpf) REFERENCES PACIENTES(cpf)
)

CREATE TABLE CONSULTAS (
    crm VARCHAR(50)
    cpf NUMBER(11)
    data_consulta DATE
    cod_mp VARCHAR(50)
    CONSTRAINT PK_CONSULTA PRIMARY KEY(crm, data_consulta)
```

```

        CONSTRAINT  FK_CRM  FOREIGN  KEY (crm)  REFERENCES
MEDICOS (crm)
        CONSTRAINT  FK_CPF  FOREIGN  KEY (cpf)  REFERENCES
PACIENTES (cpf)
        CONSTRAINT  FK_CODMP  FOREIGN  KEY (cod_mp)  REFERENCES
MEDICOS_PACIENTES (cod_mp)
    )

```

AULA 30/04

Conceito de Junções: Consulta de dados de mais de uma tabela através da relação entre as chaves estrangeiras e as pk. A junção de duas tabelas é equivalente a um produto cartesiano e aplicando projeção e seleção aos resultados.

2. Produto Cartesiano

O produto cartesiano é uma operação da teoria de conjuntos. Executar um produto cartesiano entre duas tabelas resulta na combinação de todas as linhas (registros) da primeira tabela com todas as linhas da segunda tabela.

Relação R		Relação S		
A	B	B	C	D
1	2	2	5	6
3	4	4	7	8

Resultado de R x S (Prod. Cartesiano)

A	R.B	S.B	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

O produto cartesiano é representado pelo símbolo de operação “X” entre os conjuntos.

Em SQL, chamamos essa operação de CROSS JOIN.

Repare que as tuas tabelas possuem uma coluna com o nome “B”, o que cria ambiguidade na tabela resultante. Usamos o nome da tabela e um ponto na frente de um nome de coluna para resolver a ambiguidade.

- CROSS JOIN:**

```
SELECT EST.uf, EST.nome, CID.uf, CID.nome FROM ESTADOS EST
CROSS JOIN CIDADES CID;
```

O cross join apresentará um resultado como “RESULTADO DE R X S” no produto cartesiano, para o exemplo dos estados e cidades apresentará:

ESTADOS		CIDADES	
UF	NOME	UF	NOME
AC	Acre ???	MG	Central de Minas
AC	Acre	MA	Central do Maranhão
AC	Acre	MG	Centralina
AC	Acre	MA	Centro do Guilherme
AC	Acre	MA	Centro Novo do Maranhão
AC	Acre	RO	Cerejeiras
AC	Acre	GO	Ceres
...		...	

Eles juntam de qualquer maneira e não tem uma grande utilidade, o que faria sentido é tirar um relacionamento entre R.B e S.B iguais, o comando para essa relação é **EQUI-JOIN**.

EQUI-JOINS:

- **INNER JOIN**

```
SELECT EST.uf, EST.nome, CID.uf, CID.nome FROM ESTADOS EST
INNER JOIN CIDADES CID
ON EST.uf = CID.uf;
```

O Equi-join relaciona linhas entre tabelas com um critério de equivalência, em SQL se usa o comando **INNER JOIN**. Os resultados desse comando no exemplo dos estados e cidades serão:

ESTADOS		CIDADES	
UF	NOME	UF	NOME
MG	Minas Gerais	MG	Central de Minas
MA	Maranhão	MA	Central do Maranhão
MG	Minas Gerais	MG	Centralina
MA	Maranhão	MA	Centro do Guilherme
MA	Maranhão	MA	Centro Novo do Maranhão
RO	Rondônia	RO	Cerejeiras
...		...	

É possível calcular antecipadamente o valor de registros totais do EQUI-JOIN?

```
SELECT COUNT(*)
FROM ESTADOS EST INNER JOIN CIDADES CID
ON EST.uf = CID.uf;
```

- **Consultas a partir de mais de duas tabelas:**
- **Joins encadeados:**

```
SELECT AU.nome, PROD.titulo
FROM
```

AUTORES AU

```
JOIN AUTORES_PRODUTOS AP
    ON(AU.cod_autor = AP.cod_produto)
JOIN PRODUTOS PROD
    ON(AP.cod_produto = PROD.cod_produto)
```

NOME	TÍTULO
Jacques Marcovitch	COOPERAÇÃO INTERNACIONAL: Estratégia e Gestão
João Baptista Borges Pereira	COR, PROFISSÃO E MOBILIDADE
Wander Melo Miranda	CORPOS ESCRITOS: Graciliano Ramos e Silviano Santiago
Pascal Fouché	CORRESPONDÊNCIA (1912-1922): Marcel Proust, Gaston Gallimard
Oswaldo Galotti	CORRESPONDÊNCIA DE EUCLIDES DA CUNHA
Walnice N. Galvão	CORRESPONDÊNCIA DE EUCLIDES DA CUNHA
Marcos Antonio de Moraes	CORRESPONDÊNCIA MÁRIO DE ANDRADE MANUEL BANDEIRA
Aracy Amaral	CORRESPONDÊNCIA MÁRIO DE ANDRADE TARSILA DO AMARAL
Pascal Fouché	CORRESPONDÊNCIA PROUST/GALLIMARD
Maria Inéz M.B. Pinto	COTIDIANO E SOBREVIVÊNCIA
Maria Luiza Marcilio	CRESCIMENTO DEMOGRÁFICO E EVOLUÇÃO AGRÁRIA PAULISTA: 1700-1836
Roseli Fischmann	crianças e adolescentes: Construindo uma Cultura da Tolerância
Zélia Maria Biasoli-Alves	crianças e adolescentes: Construindo uma Cultura da Tolerância
Boris Fausto	CRIME E COTIDIANO
August Strindberg	CRIMES E CRIMES
Franco Lombardi	CRISE DO NOSSO TEMPO
Maria Hermínia T. de Almeida	CRISE ECONÔMICA E INTERESSES ORGANIZADOS
Stendhal	CRÓNICAS ITALIANAS

● *Precedência de JOINS:*

ATIVIDADES DO DIA 07/05

1. O título e o nome do autor dos produtos importados, ordenados pelo título:

```
SELECT PROD.titulo, AU.nome, PROD.importado
FROM
    AUTORES AU
    JOIN AUTORES_PRODUTOS AP
        ON(AU.cod_autor = AP.cod_produto)
    JOIN PRODUTOS PROD
        ON(AP.cod_produto = PROD.cod_produto)
    WHERE PROD.importado = 'S'
    ORDER BY PROD.titulo;
```

2. A quantidade de cidades cadastradas no estado de São Paulo:

```
SELECT COUNT(*) nome
FROM CIDADES WHERE uf = 'SP'
```

3. Os nomes, e-mails e números de telefone dos clientes cujos telefones são do código de área (ddd) 51:

```
SELECT US.nome, US.email, TELE.ddd, TELE.numero
FROM
    USUARIOS US
    JOIN CLIENTES CLI
        ON(US.cod_usuario = CLI.cod_cliente)
    JOIN TELEFONES TELE
        ON(CLIENTES.cod_cliente = TELE.cod_cliente)
```

```
    ON(CLIENTE.cod_cliente = TELE.cod_cliente)
    WHERE TELEddd = '51';
```

- 4. O código, o nome e a região de cada cidade dos estados do Rio Grande do Sul, São Paulo e Pernambuco cujo nome inicia por “Santa”. Ordenar pelo nome da cidade de forma decrescente:**

```
SELECT CID.cod_cidade, CID.nome, EST.regiao, EST.uf
FROM ESTADOS EST INNER JOIN CIDADES CID
ON EST.uf = CID.uf
WHERE (CID.nome LIKE 'Santa%') and (EST.uf
IN('SP','RS','PE'))
ORDER BY CID.nome DESC;
```

- 5. Os nomes dos autores (ordenados e sem repetir) cujos produtos já foram vendidos para algum cliente da região norte:**

```
SELECT DISTINCT AU.nome,
FROM Autores AU
JOIN Autores_Produtos AP
    ON(AU.cod_autor = AP.cod_autor)
JOIN Produtos PROD
    ON(PROD.cod_produto = AP.cod_produto)
JOIN Pedidos_Produtos PD
    ON(PRD.cod_produto = PD.cod_produto)
JOIN Pedidos PED
    ON(PD.num_pedido = PED.num_pedido)
JOIN Clientes_Enderecos CE
    ON(PED.cod_cliente = CE.cod_cliente)
JOIN Enderecos ENDE
    ON(CE.cod_endereco = ENDE.cod_endereco)
JOIN CIDADES CID
    ON(ENDE.cod_cidade = CID.cod_cidade)
JOIN ESTADOS EST
    ON(CID.uf = EST.uf)
WHERE EST.regiao = 'N'
ORDER BY AU.nome;
```

AULA 14/05

Junções Externas: Quando o usuário quer que apareçam cidades com estados cadastrados e cidades sem estados cadastrados, deve-se usar o OUTER JOIN. Ele possui lado esquerdo e direito (LEFT OUTER JOIN, RIGHT OUTER JOIN e FULL OUTER JOIN).

Exemplo - Selecionar nome de todos os clientes e quando tem número de telefone:

```
SELECT USU.nome,
       DECODE(TEL.NUMERO, NULL, 'SEM TELEFONE', TEL.NUMERO)
  FROM USUARIOS USU INNER JOIN CLIENTES CLI
    ON USU.COD_USUARIO = CLI.COD_CLIENTE
   LEFT OUTER JOIN TELEFONES TEL
    ON CLI.COD_CLIENTE = TEL.COD_CLIENTE
```

Estados	Join	Cidades
Left		Right
Select Estados.uf, Cidades.uf From Estados Left outer Join Cidades on Estados.uf = Cidades.uf		
RS	+	Porto Alegre RS
PR		Canoas RS
		Gramado RS
		Niteroi null

Para o caso de dar um join “normal”, o campo Niteroi não apareceria no Select.

- Com o **LEFT OUTER JOIN** a tabela retornaria:

RS	Porto Alegre
RS	Canoas
RS	Gramado
RJ	Niterói
PR	NULL

Pois ele retorna o join normal mais os estados que estão na tabela mas não possuem cidades registradas.

- Com o **RIGHT OUTER JOIN** a tabela retornaria:

Porto Alegre	RS
Canoas	RS
Gramado	RS
Niterói	RJ
Florianópolis	NULL

Retorna o join normal mais as cidades que estão na tabela mas não possuem estados registradas - pois agora se concentra no lado direito do join.

- Com o **FULL OUTER JOIN** a tabela retornaria:

RS	Porto Alegre
RS	Canoas
RS	Gramado
RJ	Niterói
PR	NULL
NULL	Florianópolis

AULA 21/05

Select (*) - considera os nulls

Select (tabela) - não considera os nulls

Select sum(tabela) from tabela - soma do preço(exemplo) de todos os elementos

Select sum(tabela) **SOMA** from tabela - soma muda nome da tabela

Select concat(elemento, ' Olá') from produtos - retorna resultado com o "Olá"

- Arrumar média:

SELECT AVG (ddd) MEDIA FROM telefones - *considera nulls na soma*

SELECT AVG (NVL(ddd, 0)) MEDIA FROM telefones - *exclui nulls na soma*

como fazer Group By:

```
SELECT elementos
FROM tabela
GROUP BY elemento desejado
```

Usando having

```
SELECT elementos
FROM tabela
GROUP BY elemento desejado
HAVING (condição)
```

AULA 28/05

AULA 04/06

como fazer VIEW

rownum = número da linha. Ex: Selecionar todos resultados que o número da linha (índice) seja de até 3.

AULA 11/06

Aprendendo sobre **subconsultas**, existem dois tipos:

- **Sub Consultas que retornam um único valor:**

Usa valores básicos de comparação (>, <, =...)

```
SELECT preco
FROM PRODUTOS
WHERE cod_produto = 142;
```

- **Sub Consultas que retornam um conjunto de valores (ou registros)**

Como retornar todos os produtos com preço maior que o do produto de código 142:

```
SELECT preco
FROM PRODUTOS
WHERE cod_produto = 142;
```

--RETORNA 221,00-

```
SELECT titulo
FROM PRODUTOS
WHERE preco > 221.00;
```

Existe uma maneira de fazer essa consulta usando uma única query:

```
SELECT titulo
FROM PRODUTOS
WHERE preco > (SELECT preco
                  FROM PRODUTOS
                  WHERE cod_produto = 142);
```

A query dentro dos parênteses é chamada subconsulta e usam comparadores mais complexos como IN, ANY, SOME ou ALL e EXISTS.

- Se tem dois campos **antes do operador** devem ter dois campos **depois**

```
SELECT PED.num_pedido
FROM pedidos PED
WHERE (PED.cod_cliente, PED.cod_endereco)
IN (SELECT END.cod_cliente, END.cod_endereco
      FROM enderecos END
      WHERE END.cod_cidade = 20);
```

Alguns operadores:

IN - estão em

SELECT MAX - encontrar maior preço

SELECT ANY - encontrar qualquer elemento da condição

1	SELECT PROD.titulo FROM produtos PROD WHERE PROD.importado = 'N' AND PROD.preco > ANY (SELECT PROD1.preco FROM produtos PROD1 WHERE PROD1.importado = 'S');					
2						
3						
4		Titulo	Preco produto nacional	For		Preco produto importado
5		A	200			700
6		B	25			199
7		C	5			10
8		D	600			500
9						
10						
11						
12						
13						
14						

Subconsulta pode operar em duas tabelas sem dar join

```
SELECT PED.num_pedido
FROM pedidos PED
WHERE ped.cod_cliente IN (SELECT ADM.cod_administrador
                           FROM administradores ADM);
```

AULA 18/06

Revisando aulas e fazer exercícios de subconsultas

AULA 25/06

INSERT com Subconsultas:

```
INSERT INTO PRODS (codigo, nome, preco, tipo)
SELECT
cod_produto
SUBSTR(titulo, 1, 15),
preco,
'L' -- coluna constante para todos os registros
FROM produtos WHERE importado = 'N'
AND titulo LIKE 'M%'
AND cod_produto > 100;
```

UPDATE com SubConsultas:

```
UPDATE PRODUTOS
SET preco = preco - (10/100 * preco)
WHERE cod_produto IN ( SELECT cod_produto
                        FROM PRODUTOS
                        WHERE prazo_entrega > 30 );
```

DELETE com SubConsultas:

```
DELETE FROM PRODS
WHERE codigo IN
(
    SELECT cod_produto
    FROM PRODUTOS
    WHERE importado = 'N' AND titulo LIKE 'A%' AND
cod_produto > 100
);
```

AULA 02/07

Operadores de Conjunto:

Operadores de conjunto

Operador	Descrição
UNION ALL	Retorna todas as linhas recuperadas pelas consultas, incluindo as duplicadas.
UNION	Retorna todas as linhas não duplicadas recuperadas pelas consultas.
INTERSECT	Retorna as linhas recuperadas pelas duas consultas.
MINUS	Retorna as linhas restantes, quando as linhas recuperadas pela segunda consulta são subtraídas das linhas recuperadas pela primeira.

Exemplos:

- UNION ALL:

A = 1,2,3,4,5

B = 5,6,7,8,9

UNION ALL = 1,2,3,4,**5,5**,6,7,8,9

```
SELECT cod_produto
FROM PRODUTOS
WHERE cod_produto > 8 AND cod_produto < 20
UNION ALL
SELECT codigo FROM PRODS;
```

- **UNION:**

A = 1,2,3,4,5

B = 5,6,7,8,9

UNION ALL = 1,2,3,4,**5,6,7,8,9**

```
SELECT cod_produto
FROM PRODUTOS
WHERE cod_produto > 8 AND cod_produto < 20
UNION
SELECT codigo FROM PRODS;
```

- **INTERSECT:**

A = 1,2,3,4,5

B = 5,6,7,8,9

INTERSECT = 5

```
SELECT cod_produto
FROM PRODUTOS
WHERE cod_produto > 8 AND cod_produto < 20
INTERSECT
SELECT codigo FROM PRODS;
```

- **MINUS:**

A = 1,2,3,4,5

B = 5,6,7,8,9

MINUS = 1,2,3,4

```
SELECT cod_produto
```

```

FROM PRODUTOS
WHERE cod_produto > 8 AND cod_produto < 20
MINUS
SELECT codigo FROM PRODS;

```

ROTINAS ARMAZENADAS:

Aspecto de segurança, se derrubar o cliente as rotinas continuam rodando no servidor

SEQUÊNCIA: Para ir incrementando o valor e poder, por exemplo, usar auto incremento em uma PK.

- NEXTVAL = próximo valor
- CURRVAL = valor atual

TRIGGER: Para definir regras no insert de um elemento, por exemplo, colocar uma sequência no ID.

```

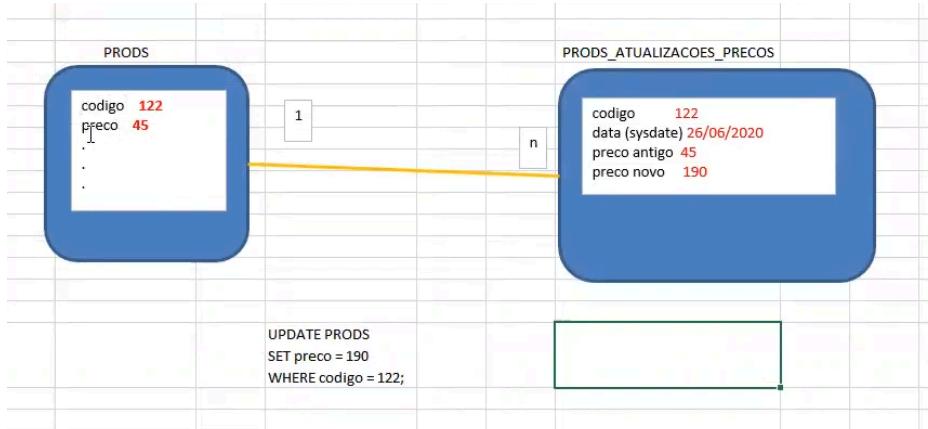
CREATE OR REPLACE TRIGGER AUTO_COD_PRODS
BEFORE INSERT ON prods
FOR EACH ROW
WHEN (NEW.codigo IS NULL)
BEGIN
    SELECT SEQ_PRODS NEXTVAL
    INTO: NEW.codigo
    FROM dual;
END;

```

Então para inserir os itens não listar o campo “codigo” no values. E ele faz isso ANTES DE INSERIR o valor do INSERT.

Qualificadores :Old :New		
DML	:Old	:New
INSERT	Nulo	Valores novos
DELETE	Valores antigos	Nulo
UPDATE	Valores antigos	Valores novos

- Exemplo de Gatilho (Trigger) - Log de alterações



Criar uma tabela para atualização:

```

CREATE TABLE PRODS_ATUALIZACOES_PRECOS
(
    codigo NUMBER(3) NOT NULL,
    data DATE NOT NULL,
    preco_anterior NUMBER(5,2) NOT NULL,
    preco_novo NUMBER(5,2) NOT NULL,
    CONSTRAINT FK_PRODS_ATUALIZACOES FOREIGN KEY(codigo) REFERENCES PRODS(codigo)
);

```

Criar o trigger:

```

11 CREATE OR REPLACE TRIGGER PRODS_PRECOS_UPDATE
12 BEFORE UPDATE OF preco ON PRODS
13 REFERENCING NEW AS NEW OLD AS OLD
14 FOR EACH ROW
15 BEGIN
16     INSERT INTO PRODS_ATUALIZACOES_PRECOS (codigo, data, preco_anterior, preco_novo)
17     VALUES (:NEW.codigo, sysdate, :OLD.preco, :NEW.preco);
18 END;
19
30 SELECT * FROM PRODS;
31

```

CRIAR PROCEDURE:

```

SELECT preco FROM PRODUTOS
WHERE cod_produto = 120;

CREATE OR REPLACE PROCEDURE aumenta_preco
(pcod_produto IN produtos.cod_produto %TYPE,
ppercentual IN NUMBER DEFAULT 10)

IS
BEGIN

UPDATE PRODUTOS
SET preco = preco * (1 + (ppercentual / 100))
WHERE cod_produto = pcod_produto;

```

Para rodar procedure:

```
Exec aumenta_preco (120,10);
```

CRIAR FUNCTION:

```

CREATE OR REPLACE FUNCTION pega_preco
(p_id IN produtos.cod_produto%TYPE)
RETURN NUMBER
IS
    v_preco produtos.preco%Type :=0;
BEGIN
    SELECT preco|
        INTO v_preco
        FROM produtos
        WHERE cod_produto = p_id;
    RETURN v_preco;
END pega_preco;

```

Imprimir:

nilha	Query Builder
<pre> variable valor number execute :valor := pega_preco(120); print valor; </pre>	

Ou:

```
SELECT pega_preco FROM dual;
```

Ou dentro do dl.sql

