Desenvolvimento Banco Dados - SQL

SQL - Structured Query Language

Base Oracle

Ana Leda

SQL - Structured Query Language

- **Definida por D. D. Chamberlin e outros, nos laboratórios de pesquisa da IBM em San Jose, California, em 1974.**
- **Fundamentos no modelo relacional de Edgar Frank Codd, criado em 1970.**
- **Primeira versão recebeu o nome de SEQUEL - Structured English Query Language.**
- **Órgãos como ANSI e ISO adotaram a SQL como o padrão oficial de linguagem em ambiente relacional.**

SQL - Structured Query Language

- **ANSI publicou as padronizações SQL ANSI-89 e ANSI-92.**
- **Vários dialetos SQL adotam o padrão ANSI com extensões proprietárias.**
- **Interface ODBC (Open Database Connectivity) converte sintaxe SQL de um produto em outro.**
- **Independência de fabricante**
  - ✓ incorporada em quase todos os SGBD's em seu padrão ANSI, com as extensões proprietárias de cada fabricante.

SQL - Structured Query Language

- **Portabilidade entre computadores**
  - ✓ pode ser usada desde um PC até um mainframe.
- **Redução de custos com treinamento**
  - ✓ aplicações podem migrar de ambiente com custo reduzido em treinamento.
- **Facilidade no entendimento**
  - ✓ comandos escritos em um inglês estruturado de alto nível.
- **Consulta interativa**
  - ✓ acesso rápido aos dados através de comandos interativos.

SQL - Structured Query Language

- **Múltiplas visões de dados**
  - ✓ possibilita levar diferentes visões dos dados a diferentes usuários.

- **Definição dinâmica de dados**
  - ✓ possibilita a alteração dinâmica das estruturas dos dados armazenados, com flexibilidade.

SQL - Structured Query Language

- **Pesquisar dados no banco.**
- **Adicionar, modificar e remover dados.**
- **Criar, modificar e remover estruturas de dados.**

SQL - Structured Query Language

- **Linguagem de Manipulação de Dados (DML)**
  - ✓ Insert
  - ✓ Update
  - ✓ Delete
- **Linguagem de Consulta de Dados (DQL)**
  - ✓ Select

SQL - Structured Query Language

- **Linguagem de Definição de Dados (DDL)**
  - ✓ Create Table
  - ✓ Alter Table
  - ✓ Drop Table
  - ✓ Rename

Regras para escrever comandos SQL

- **Comandos podem ser escritos em mais de uma linha.**
- **Cláusulas diferentes são colocadas usualmente em linhas diferentes.**
- **Comandos podem ser escritos em letras maiúsculas e minúsculas.**

SQL - Tipos de Dados

- **Char (S)**
  - ✓ Alfanumérico de tamanho fixo, máximo 2000 caracteres.
- **Varchar2 (S)**
  - ✓ Alfanumérico com tamanho máximo de 4000 caracteres. O que não for utilizado não ocupa espaço no banco de dados.
- **Number (P,S)**
  - ✓ Valor numérico com tamanho máximo de 38 caracteres, sendo **P** o nº total de dígitos (inteiros + decimais) e **S** o nº de casas decimais.
- **Date**
  - ✓ Valor de data e hora (01/01/4712 AC até 31/12/4712 DC).

Definir Restrições de Integridade de Dados

- **Constraint ou Restrição**
  - ✓ representa um mecanismo capaz de implementar controles que garantam a consistência dos dados (integridade de dados referencial).
  - ✓ pode ser definido tanto para uma coluna (afeta apenas um campo), como para toda a tabela (afeta todos os campos).

Definir Restrições de Integridade de Dados

Constraint	Descrição
Not Null	Especifica que a coluna não pode conter valores nulos.
Unique	Especifica uma coluna ou combinação de colunas que terão seus valores únicos na tabela.
Primary Key	Identifica a unicidade de cada linha na tabela.
Foreign Key (references)	Estabelece e força um relacionamento entre tabelas.
Check	Especifica uma condição que deve ser verdadeira obedecendo uma regra do negócio.

SQL - Comandos

- **Estrutura física das tabelas - comandos DDL que afetam as colunas das tabelas (campos)**
  - ✓ Criação da tabela: Comando CREATE TABLE
  - ✓ Alteração da tabela: Comando ALTER TABLE
  - ✓ Exclusão da tabela: Comando DROP TABLE

SQL - Comandos

- **Create Table - Criar tabelas p/armazenar dados**
  - ✓ Sintaxe:

Create Table [User.]Nome_Tabela

(Nome_Coluna DataType [(Tamanho)] [Default expr] [Constraint] [,

Nome_Coluna DataType [(Tamanho)] [Default expr] [Constraint]] [,...)

NOME_TABELA - nome da tabela

NOME_COLUNA - nome do campo

DATATYPE - tipo de dado da coluna

CONSTRAINT_TABELA - "CONSTRAINT" de dado para a coluna

SQL - Comandos

- **Create - Regras de Nomeação**
  - ✓ nomes da tabelas e colunas devem ser de 1-30 caracteres;
  - ✓ o primeiro caracter deve ser alfabético;
  - ✓ nomes devem conter somente os caracteres de a-z, A-Z, 0-9, _(underscore), $ e #(seu uso não é recomendado);
  - ✓ nomes não podem ser iguais as palavras reservadas do Oracle;
  - ✓ nome da tabela não pode ser igual ao nome de um outro objeto criado pelo mesmo usuário Oracle.

SQL - Comandos

- **Create Table**
  - ✓ Exemplo 1:

```
CREATE TABLE Regiao
(
  Codigo Number(7),
  Nome   Varchar(30) CONSTRAINT Regiao_Nome_NN NOT NULL,
  CONSTRAINT Regiao_Codigo_PK PRIMARY KEY(Codigo),
  CONSTRAINT Regiao_Nome_UK UNIQUE (Nome)
);
```

SQL - Comandos

- **Create Table**
  - ✓ Exemplo 2:

```
CREATE TABLE Empregado
(
  Codigo Number(7),
  Nome Varchar(25) CONSTRAINT Emp_Nome_NN NOT NULL ,
  DtAdm Date,  CPF Number(15), Comentarios Varchar(25),
  Salario Number(11,2), PctComis Number(4,2),
  CONSTRAINT Emp_Cod_PK PRIMARY KEY (Codigo),
  CONSTRAINT Emp_CPF_UK UNIQUE (CPF),
  CONSTRAINT Emp_PctComis CHECK (PctComis IN (10, 12.5, 15))
  );
```

SQL - Comandos

- **Alter Table – Alterar estrutura de uma tabela**
  - ✓ criar, alterar ou eliminar colunas;
  - ✓ renomear a tabela;
  - ✓ criar ou eliminar constraints;

SQL - Comandos

- ALTER TABLE
  - ADD
  - ADD CONSTRAINT
  - MODIFY
  - DROP CONSTRAINT
  - DROP COLUMN
  - RENAME COLUMN
  - RENAME

SQL - Comandos

- **Alter Table – Criando nova coluna na tabela**
  - ✓ Sintaxe:

```
Alter Table Nome_Tabela Add
  (Nome_Coluna DataType [Constraint_ Coluna],...)
```

  - ✓ Exemplo:

```
ALTER TABLE Regiao ADD
  (Comentarios Varchar2(100));
```

SQL - Comandos

- **Alter Table – Alterando uma coluna na tabela**
  - ✓ Sintaxe:

```
Alter Table Nome_Tabela Modify
  (Nome_Coluna DataType|[Constraint_ Coluna],...)
```

  - ✓ Exemplo:

```
ALTER TABLE Regiao MODIFY
   (Comentarios Varchar2(255));
```

OBS: A modificação do tipo e/ou tamanho só deve ser feita se não houver dados cadastrados na coluna.

SQL - Comandos

- **Alter Table – Adicionando uma Constraint numa coluna já existente na tabela**

  - ✓ Exemplo:

```
ALTER TABLE Departamento ADD CONSTRAINT
    Depto_CodReg_FK Foreign Key (Cod_Regiao)
    REFERENCES Regiao(Codigo);
```

SQL - Comandos

- **Alter Table – Eliminando Constraint em uma tabela**

  - ✓ Exemplo:

```
ALTER TABLE Departamento DROP CONSTRAINT
    Depto_CodReg_FK;
```

SQL - Comandos

- **Alter Table – Renomeando coluna em uma tabela**

  - ✓ Exemplo:

```
ALTER TABLE Regiao RENAME COLUMN
    Comentarios TO Observacao;
```

SQL - Comandos

- **Alter Table – Eliminando coluna em uma tabela**

  - ✓ Exemplo:

```
ALTER TABLE Regiao DROP COLUMN Observacao;
```

SQL - Comandos

- **Alter Table – Renomeando uma tabela**

  - ✓ Exemplo:

    ALTER TABLE Regiao RENAME TO Regioes;

  OBS: Também é possível renomear uma tabela desta forma:

  RENAME Regiao TO Regioes;

SQL - Comandos

- **Drop Table – Eliminar uma tabela**
  - ✓ Sintaxe:

    Drop Table Nome_Tabela;

  - ✓ Exemplo:

    DROP TABLE Departamento;

SQL - Comandos

- **Drop Table – Eliminar uma tabela e constraints de FK**
  - ✓ Sintaxe:

    Drop Table Nome_Tabela CASCADE CONSTRAINT;

  - ✓ Exemplo:

    DROP TABLE Dept CASCADE CONSTRAINT;

    OBS: A tabela DEPT é eliminada, bem como, todos os relacionamentos que houverem com essa tabela, em que a chave primária de DEPT seja chave estrangeira em alguma delas.

SQL - Comandos

- **Alterações nos dados - comandos DML que afetam as linhas de uma tabela (registros)**
  - ✓ Inclusão de linhas: Comando INSERT
  - ✓ Alteração de linhas: Comando UPDATE
  - ✓ Exclusão de linhas: Comando DELETE

SQL - Comandos

- **Insert – Adicionar linha numa tabela**
  - ✓ Sintaxe:

```
INSERT INTO Nome_Tabela (coluna1, coluna2, colunaN)
VALUES (valor1, valor2, valorN);
```

  - ✓ Exemplo 1:

```
INSERT into Aluno (matricula, nome_aluno)
VALUES (8,'Milton Soares');
```

OBS:Se não forem definidos todos os campos, eles ficarão com valor nulo (se permitido).

SQL - Comandos

- **Insert – Inserindo linhas sem relacionar colunas**
  - ✓ Exemplo 2:

```
INSERT into Funcionario VALUES (8,'Milton',2000,'M');
```

OBS:Neste caso, os valores correspondem aos campos de acordo com a ordem em que foram definidos na tabela. A relação de colunas da tabela só pode ser omitida quando forem inclusos valores para todas as colunas dessa tabela.

SQL - Comandos

- **Insert – Inserindo linhas com variáveis**
  - ✓ Exemplo 3:

```
INSERT into Aluno (cod_matric, nome_aluno, data_matric)
VALUES (&Matricula, '&Nome', '&Data');
```

O comando INSERT terá a seguinte execução:

Informe o valor para matricula:
Informe o valor para nome:
Informe o valor para data:

Este comando poderá ser executado diversas vezes, pedindo novos valores para os campos, bastando para isso, ao final de cada execução, digitar / e <Enter>.

Quando os campos são do tipo caractere ou data, a entrada dos valores precisam estar entre aspas simples. Para facilitar, coloque o nome das variáveis entre aspas, dispensando o uso de aspas pelo usuário.
Ex: ('&Nome', '&Data')

SQL - Comandos

- **Update – Alterar dados numa tabela**

  - ✓ Sintaxe:

```
UPDATE <Tabela>
SET <Coluna> = <Novo Valor>
    [, <Coluna> = <Novo Valor>...]
WHERE <Condição>
```

SQL - Comandos

- **Update**
  - ✓ Na cláusula SET são especificados os campos que terão seu valor alterado e qual será seu novo valor;
  - ✓ A cláusula WHERE especifica qual(is) a(s) linha(s) que terá(ão) esse valor alterado. Caso não haja cláusula WHERE, os valores serão alterados para todas as linhas da tabela;
  - ✓ A expressão de atualização pode envolver a própria coluna;
  - ✓ O WHERE pode ter qualquer condição válida, inclusive subconsultas. O WHERE é avaliado antes das atualizações, uma vez que pode envolver valores sendo alterados.

SQL - Comandos

- **Update**

  - ✓ Exemplo 1: Atualizar todos os salários aumentando 5%.

```
UPDATE Funcionario SET Salario = Salario * 1.05;
```

  - ✓ Exemplo 2: Atualizar o salário de todos os funcionários para 10000.

```
UPDATE Funcionario SET Salario = 10000;
```

SQL - Comandos

- **Update com WHERE**

  - ✓ Exemplo 1: Atualizar os salários aumentando 5% para os funcionários que ganham menos que 1000.

```
UPDATE Funcionario SET Salario = Salario * 1.05
WHERE Salario < 1000;
```

SQL - Comandos

- **Delete – Excluir linhas de uma tabela**

  - ✓ Sintaxe:

    DELETE from <Tabela> [WHERE <Condição>]

    OBS: Caso a cláusula WHERE seja omitida, todas as linhas da tabela serão eliminadas. CUIDADO!

  - ✓ Exemplo:

    DELETE from Funcionario;

    DELETE from Produto;

SQL - Comandos

- **Delete**
  - ✓ Quando usado com o WHERE, pode especificar quais linhas apagar;
  - ✓ O WHERE do DELETE comporta quase todas as condições possíveis do Select, desde que seja somente referente a uma tabela;
  - ✓ Subconsultas são permitidas;

SQL - Comandos

- **Delete**
  - ✓ Exemplos:

```
DELETE from Funcionario WHERE NumDepto= 003;

DELETE from Funcionario WHERE
Salario between 1000 and 2000;

DELETE from EMP WHERE deptno in
(Select deptno from DEPT where dname = 'SALES');
```

SQL - Comandos

- **Delete**
  - ✓ O teste da condição é feito antes das linhas serem removidas.
  - ✓ Na consulta a seguir, se isso não fosse feito, poderíamos ter resultados imprevisíveis:
    - ✓ Exemplo:

```
DELETE from funcionario WHERE salario >
(Select avg(salario) from Funcionario)
```

SQL - Transações

- **Commit**

  Finaliza a transação corrente, tornando permanente todas as mudanças pendentes no banco de dados feitas pelos comandos insert, update e delete.

  Toda mudança de dados efetuada durante a transação é temporária até que a transação seja efetivada, ou seja, dado um "COMMIT".

  - ✓ Sintaxe:

    COMMIT

SQL - Transações

- **Show autocommit**

  Mostra como está o commit, se ON ou OFF.

  - ✓ Sintaxe:
    SHOW AUTOCOMMIT

- **Set autocommit on**

  Um Commit é feito a cada inserção, alteração e deleção.

- **Set autocommit off**

  É necessário fazer o Commit para efetivar as transações pendentes.

SQL - Transações

- **Rollback**

  Finaliza a transação corrente descartando todas as mudanças pendentes no banco desde o último COMMIT EFETIVADO.

  - ✓ Sintaxe:

    ROLLBACK

SQL - Comandos

- **Comando SELECT – consultar dados numa tabela**

  O SELECT possui uma cláusula obrigatória (FROM) e outras cláusulas opcionais (INTO, WHERE, GROUP BY, HAVING, UNION, ORDER BY).

  - ✓ Sintaxe:

    SELECT <coluna1>, <coluna2>,...,<colunan>

    FROM <tabela1>, <tabela2>,...<tabelam>

SQL - Comandos

- **Comando SELECT**
  - ✓ O sistema consegue descobrir de onde cada coluna vem pelo dicionário de dados.
  - ✓ No select, escolhemos quais colunas farão parte do resultado.
  - ✓ As colunas devem pertencer as tabelas presentes na cláusula FROM.
  - ✓ Quando há colunas com o mesmo nome, diferencia-se colocando o nome da tabela antes do nome da coluna, separadas por um ponto.
  - ✓ Quando desejamos escolher todas as colunas, podemos usar o asterisco (*). Ex: sql> SELECT * FROM funcionario
  - ✓ As repetições são mantidas. Para eliminá-las podemos usar a cláusula distinct. Ex: SELECT DISTINCT nome FROM Aluno

SQL - Comandos

- **Select - Alterando o Cabeçalho das Colunas**
  - ✓ Sintaxe:

SELECT <campo1 [AS] nome1, campo2 [AS] nome2, ...>
FROM <tabela>

OBS: Exibe todos os campos constantes na <lista de campos> e exibe também um nome alternativo para cada campo, que será exibido como cabeçalho da coluna correspondente.

SQL - Comandos

- **Select - Alterando o Cabeçalho das Colunas**

  - ✓ Exemplo 1: Exibir nome, salário e uma coluna contendo "não classificado" com o cabeçalho "Classificação" para todos os empregados.

```
SELECT ename , sal, 'não classificado' as Classificação
FROM Emp
```

SQL - Comandos

- **Select – resultado do comando anterior**

```
SQL> /

ENAME      SAL       CLASSIFICAÇÃO
---------- -------   --------------------
SMITH        800       não classificado
ALLEN       1600       não classificado
WARD        1250       não classificado
JONE S      2975       não classificado
MARTIN      1250       não classificado
BLAKE       2850       não classificado
CLARK       2450       não classificado
SCOTT       3000       não classificado
KING        5000       não classificado
```

SQL - Comandos

- **Select - Alterando o Cabeçalho das Colunas**

  OBS:

  - ✓ Se o pseudônimo contém brancos, caracteres especiais como (# ou _), ou em casos similares, incluir o pseudônimo entre aspas dupla (“”).

  - ✓ Separe as colunas do pseudônimo usando espaços.

SQL - Comandos

- **Select - Alterando o Cabeçalho das Colunas**
  - ✓ Exemplo 2:

```
SELECT Ename Nome, Empno Registro, Job "Ocupação"
From Emp;
```

NOME	REGISTRO	Ocupação
SMITH	7369	CLERK
ALLEN	7499	SALESMAN
WARD	7521	SALESMAN
JONES	7566	MANAGER
MARTIN	7654	SALESMAN

SQL - Comandos

- **Select – ordenação das linhas**
  - ✓ As linhas aparecem ordenadas pela chave primária, se esta estiver definida.
  - ✓ Pode-se mudar a ordenação através do comando order by
  - ✓ Exemplo 1:

```
SELECT Codigo, Nome, Curso from Aluno ORDER BY Nome;
```

SQL - Comandos

- **Select – ordenação das linhas**
  - ✓ O padrão do ORDER BY é que a ordenação seja crescente, mas usando-se o DESC, pode-se inverter a ordem.
  - ✓ Exemplo 2:

```
SELECT Codigo, Nome, Curso from Aluno
ORDER BY Nome DESC;
```

SQL - Comandos

- **Select – ordenação das linhas**
  - ✓ Em caso de "empate", a ordem é arbitrária, a não ser que sejam definidas mais colunas para o ORDER BY.
  - ✓ Exemplo 3:

```
SELECT Nome, DataNasc from Pessoa
ORDER BY Nome, DataNasc;
```

SQL - Comandos

- **Select – usando a cláusula Where**

  Mostrar linhas específicas de uma tabela de acordo com uma condição de pesquisa.

  ✓ Sintaxe:

  SELECT <Nome_coluna> FROM <TABELA>

  WHERE (CONDIÇÃO)

SQL - Comandos

- **Select – usando a cláusula Where**

Operadores de Comparação

Operador	Descrição
=	Igual
<>	Diferente
>	Maior que
<	Menor que
>=	Maior ou igual
<=	Menor ou igual
Between and	Comparações entre dois valores

SQL - Comandos

- **Select – usando a cláusula Where**

Operadores de Comparação

Operador	Descrição
Not between and	Não comparação entre dois valores
IN (Lista)	Igual a qualquer membro da lista
Not IN (Lista)	Diferente a qualquer membro da lista
Like	Unir uma sequência de caracteres usando uma sequência

SQL - Comandos

- **Select – usando a cláusula Where**

Operadores Lógicos

Operador	Descrição
AND	Retorna apenas registros que atendam simultaneamente todas as condições solicitadas
OR	Retorna registros que atendam ao menos uma das condições solicitadas
NOT	Retorna registros que atendam a condição oposta a solicitada
AND/OR Combinados	Usados para combinar expressões lógicas

SQL - Comandos

- **Precedência dos operadores**

Ordem de Precedência	Operador
1	Todos de comparação
2	AND
3	OR

SQL - Comandos

- **Select – usando a cláusula Where**
  - ✓ Exemplo 1: Especificando linhas com a cláusula Where (**=**)

```
SELECT Ename Nome, Job Cargo from Emp

WHERE Deptno=20
```

```
SQL> /
NOME       CARGO
---------- -------
Smith      Clerk
Jones      Manager
Scott      Analyst
Adams      Clerk
Ford       Analyst
```

SQL - Comandos

- **Select – usando a cláusula Where**
  - ✓ Exemplo 2: Especificando linhas com a cláusula Where (**<>**)

```
SELECT Ename Nome, Job Cargo from Emp
WHERE Job <> 'CLERK'
```

```
SQL> /
NOME       CARGO
---------- -------
Allen      Salesman
Ford       Analyst
Jones      Manager
Martin     Salesman
King       President
```

SQL - Comandos

- **Select – usando a cláusula Where**

  - ✓ Exemplo 3: Especificando linhas com a cláusula Where between

```
SELECT Ename Nome, Job Cargo from Emp
WHERE hiredate between '01-JAN-87' AND '31-DEC-87'
```

```
SQL> /
NOME       CARGO
---------- -------
King       President
Allen      Salesman
Ward       Salesman
Martin     Salesman
Turner     Salesman
```

SQL - Comandos

- **Select – usando a cláusula Where**

  - ✓ Exemplo 3: Especificando linhas com a cláusula Where com intervalos. Para valores fora de um intervalo, pode-se usar o not between.

```
SELECT * from Aluno
WHERE Ingresso Not between '01/01/90' AND '31/12/99';
```

SQL - Comandos

- **Select – comparador LIKE**

  - ✓ O comparador like ajuda em consultas em que não temos o valor exato de uma cadeia de caracteres para encontrar
  - ✓ % significa qualquer substring (* do DOS)
  - ✓ _ compara único caracter (posição)

SQL - Comandos

- **Select – usando a cláusula Where**

  - ✓ Exemplos: Especificando linhas com a cláusula Where utilizando o comparador LIKE

```
SELECT * from Funcionario WHERE Nome LIKE 'José%';

SELECT * from Disciplina WHERE Nome LIKE '%cálculo%';

SELECT * from Aluno WHERE Nome LIKE '___ Paula%';

SELECT * from Cliente WHERE Nome LIKE '%Pedro%';
```

SQL - Comandos

- **Select – usando a cláusula Where**

  - ✓ Exemplos: Especificando linhas com a cláusula Where utilizando o comparador IN

```
SELECT Sobrenome, Cargo from Empregado
WHERE Cargo IN ('Vendedor','Representante');

SELECT Sobrenome, Cargo from Empregado
WHERE Cargo NOT IN ('Analista','Gerente');
```

SQL - Comandos

- **Criando uma tabela através do comando Select**

  - ✓ Sintaxe:

    CREATE TABLE nome_tabela As SELECT * FROM nome_tabela;

  - ✓ Exemplo:

    CREATE TABLE EmpCopia
    As
    SELECT * FROM Emp;

    Obs: Este comando faz uma cópia da tabela (estrutura e registros) mas não gera todas as constraints da tabela original para a nova tabela. Apenas as constraints de Not Null são copiadas para a nova tabela.

SQL - Comandos

- **Criando uma tabela através do comando Select com seleção de campos e cláusula where**

  - ✓ Exemplo:

```
CREATE TABLE EmpCopia
As
SELECT ename, empno FROM Emp where depto = 30;
```

SQL - Comandos

- **Insert - Inserindo linhas com o comando Select**

  - ✓ Podemos incluir várias linhas na tabela com o uso do comando select.

  - ✓ Exemplo 3:

    INSERT into EmpCopia SELECT ename, empno FROM EMP

    where deptno = 10;

SQL – Funções Caracteres

- **CONCAT** - concatena duas colunas ou cadeia de caracteres. Esta função é equivalente ao operador de concatenação(||).
  - ✓ Sintaxe:

    CONCAT (char1,char2)

  - ✓ Exemplo:

    Select concat('Lab','Banco') from Emp;

    CONCAT

    LabBanco
    LabBanco
    LabBanco

SQL - Funções Caracteres

- **LOWER** - retorna todas as letras minúsculas.
  - ✓ Sintaxe: LOWER(char)
  - ✓ Exemplo: Select lower(ename) from Emp;

LOWER(Ename)
smith
allen
ward
jones

SQL – Funções Caracteres

- **UPPER** - retorna todas as letras maiúsculas.
  - ✓ Sintaxe:

    UPPER(char)
  - ✓ Exemplo:

    Select upper(ename) from Emp;

```
     ename
----------
SMITH
ALLEN
WARD
JONES
```

SQL - Funções Caracteres

- **SUBSTRING** - retorna parte de uma string. Deve-se indicar a posição inicial (m) da string e quantidade de caracteres (n) a serem contados. Se a quantidade de caracteres não for especificada, a função contará os caracteres até o final da string.

  - ✓ Sintaxe:

    SUBSTR(char,m,n)

  - ✓ Exemplo:

    Select substr(dname,2) from Dept;

    dname (Retorna a partir da posição 2 a coluna selecionada)

    CCOUNTING

    ESEARCH

    ALES

SQL – Funções Datas

- **Realizar cálculos com Data**

Expressão	Descrição
Data + Número	Soma à data número de dias
Data - Número	Substrai à data número de dias
Data - Data	Retorna o número de dias entre as datas

SQL - Funções Datas

- **Exemplo:**

  Select ename, hiredate, hiredate + 90 from Emp
  where deptno = 30;

ename	hiredate	hiredate+90
ALLEN	20-FEB-81	21-MAY-81
WARD	22-FEB-81	23-MAY-81
MARTIN	28-SEP-81	27-DEC-81
BLAKE	01-MAY-81	30-JUL-81
TURNER	08-SEP-81	07-DEC-81
JAMES	03-DEC-81	03-MAR-82

SQL - Funções Datas

- **SYSDATE –** retorna a data do sistema.
  - ✓ Exemplo:

```
Select hiredate, (sysdate - hiredate)/7 semanas from emp;
```

HIREDATE	SEMANAS
17-DEC-80	1080.8207
20-FEB-81	1071.535
22-FEB-81	1071.2493
02-APR-81	1065.6778
28-SEP-81	1040.1064

SQL - Funções Datas

- **ADD_MONTHS –** retorna a data adicionada n meses especificado
  - ✓ Sintaxe:

    ADD_MONTHS (data, número)
  - ✓ Exemplo:

    Select hiredate, add_months(hiredate,6) from emp;

HIREDATE	ADD_MONTH
17-DEC-80	17-JUN-81
20-FEB-81	20-AUG-81
22-FEB-81	22-AUG-81
02-APR-81	02-OCT-81
28-SEP-81	28-MAR-82

SQL - Funções Datas

- **MONTHS_BETWEEN –** retorna o número de meses entre uma data e outra.
  - ✓ Sintaxe:

    MONTHS_BETWEEN (data, data)

  - ✓ Exemplo:

    Select hiredate, months_between(sysdate,hiredate) from emp;

HIREDATE	MONTHS_BETWEEN(
17-DEC-80	248.57263
20-FEB-81	246.47586
22-FEB-81	246.41134
02-APR-81	245.0565

SQL – Reformatação de Datas

Elemento	Descrição
DD	Dia do mês
DY	Nome do dia abreviado com 3 letras
DAY	Nome do dia com 9 letras
DDSP	Número do dia no mês por extenso
MM	Número do mês
MON	Nome do mês abreviado com 3 letras
MONTH	Nome do mês com 9 letras
YY	Dois dígitos do ano
YYYY	Quatro dígitos do ano
HH:MI:SS	Hora, Minuto e Segundos
FM	Coloca espaço extra em cada elemento da máscara
HH24	Hora do dia, formato 24 horas

SQL - Funções de Conversão

- **TO_CHAR –** converte um valor tipo data ou número para um valor char. Normalmente é utilizado para a formatação de datas e números.
  - ✓ Sintaxe:

    TO_CHAR (data [, formato_char]) ou TO_CHAR(num [,formato_char])

  - ✓ Exemplo:

    Select ename, to_char(hiredate,'MM/YY') "Admissão" from emp;

ENAME	Admissão
SMITH	12/80
ALLEN	02/81
WARD	02/81
JONES	04/81
MARTIN	09/81
BLAKE	05/81
CLARK	06/81

SQL - Funções de Conversão

- ✓ Exemplo:

  Select ename, to_char(hiredate, 'DD "de" month YYYY') "Admissão" from Emp;

ENAME	Admissão
SMITH	17 de december 1980
ALLEN	20 de february 1981
WARD	22 de february 1981
JONES	2 de april 1981
MARTIN	28 de september 1981
BLAKE	1 de may 1981
CLARK	9 de june 1981

SQL - Funções de Conversão

- **TO_DATE –** converte uma expressão char para date.
  - ✓ Sintaxe:

    TO_DATE (char [, formato_char])

  - ✓ Exemplo:

    Insert into Emp Values
    (222, 'Isa VonDix','Matriz', 3424,
    TO_DATE('070393','MMDDYY'),3000,0,10);

    Select ename, hiredate from Emp where empno = 222;

ENAME	HIREDATE
Isa VonDix	03-JUL-93

SQL - Funções Numéricas

- **ROUND –** arredonda um valor mantendo apenas a parte inteira ou com o nº de casas decimais solicitado.

  - ✓ Sintaxe:

    ROUND (num [, nº casas decimais])

  - ✓ Exemplo:

    Round (1456.8976) ➔ 1457

    Round (1456.8976, 2) ➔ 1456.90

SQL - Funções Numéricas

- **TRUNC –** traz o número sem arredondar, desprezando as casas não solicitadas.

  - ✓ Sintaxe:

    TRUNC (num [, nº casas decimais])

  - ✓ Exemplo:

    Trunc (1456.8976) ➔ 1456

    Trunc (1456.8976,2) ➔ 1456.89

SQL – Funções de Grupo

As funções de grupo agem sempre sobre um conjunto de linhas ao invés de agir sobre cada linha separada.

- **Função AVG –** calcula a média dos valores selecionados

  - ✓ Exemplo:

    SQL> Select avg(sal) from emp;

    Resultado do exemplo: 3456,345

SQL – Funções de Grupo

- **Função MIN/MAX –** seleciona o menor/maior valor dos valores selecionados.

  - ✓ Exemplo:

    SQL> Select min(sal) from emp where job = 'CLERK';

    Resultado do exemplo: MIN(SAL) = 400

SQL – Funções de Grupo

- **Função COUNT –** faz a contagem das linhas retornadas de uma query.

  Count (*) – O * faz contar todas as linhas.

  - ✓ Exemplo:

    SQL> Select count(sal) from emp;

SQL – Funções de Grupo

- **Função SUM –** faz a soma dos valores retornadas numa query**.**
  - ✓ Exemplo:

    SQL> Select sum(sal) from emp;

SQL – Cláusulas Group By e Having

- **Exibir totalização de resultado para grupos de linhas com a cláusula GROUP BY e HAVING.**

  - ✓ Sintaxe:

    SELECT nome_coluna FROM nome_tabela WHERE condição

    GROUP BY Expressão_Group_By

    Expressão_Group_By especifica as colunas pelos quais serão agrupados pela função de grupo.

SQL – Cláusulas Group By e Having

- **GROUP BY**
  - ✓ Exemplo:

    SELECT codequipe, count(*) "Nº de Funcionários" FROM funcionario GROUP BY codequipe;

CodEquipe	Nº de Funcionários
1	3
2	2
3	1
4	1
5	1
6	1

SQL – Cláusulas Group By e Having

- **GROUP BY –** Nunca selecione uma coluna simples com uma função de grupo sem a cláusula GROUP BY.

  - ✓ Exemplo:

    SELECT codequipe as equipe, funcao Cargo,
    count(*) "Nº de Funcionários" FROM funcionario;

    ERROR at line 1:

    ORA-00937: not a single-group group function

SQL – Cláusulas Group By e Having

- Exemplo:

```
SELECT codequipe as equipe, funcao Cargo,
count(*) "Nº de Funcionários" FROM funcionario
GROUP BY codequipe, funcao;
```

Equipe	Cargo	Nº de Funcionários
1	Gerente	1
1	Engenheiro	2
2	Engenheiro	2
3	Mecânico	1
4	Mecânico	1
5	Gerente	1
6	Mecânico	1

SQL – Cláusulas Group By e Having

- **Mostrar linhas específicas ou grupos específicos utilizando a cláusula HAVING.**

  - ✓ Sintaxe:

    SELECT nome_coluna FROM nome_tabela WHERE condição
    GROUP BY Expressão_Group_By HAVING condição

    Having condição *restringe o grupo de linhas retornadas daqueles grupos para especificar uma condição TRUE.*

    A cláusula **HAVING** filtra linhas **após** o agrupamento e só pode ser utilizada se houver a cláusula **GROUP BY** no comando.

SQL – Cláusulas Group By e Having

- ✓ Exemplo:

```
SELECT funcao, sum(salario) "Soma Sal" FROM funcionario
GROUP BY funcao HAVING sum(salario) > 5000;
```

Funcao	Soma Sal
Gerente	12300.00
Engenheiro	8200.00
Mecânico	2500.00

SQL – Cláusulas Group By e Having

- ✓ Exemplo:

```
SELECT funcao, sum(salario) "Soma Sal" FROM funcionario
WHERE funcao not like 'Mec%'
GROUP BY funcao HAVING sum(salario) < 10000;
```

Funcao	Soma Sal
Engenheiro	8200.00

SQL – Sub-Queries

- **Uma subquery é um comando select dentro de qualquer comando DML.**
  - ✓ Sintaxe:

    SELECT col1, col2 FROM table1 WHERE columm =

    (SELECT nome_coluna FROM nome_tabela WHERE *condição*)

  - ✓ Exemplo: Quando queremos encontrar o funcionário que recebe o menor salário da empresa, sendo que não sabemos qual é o menor salário. Poderíamos dividir essa consulta em duas:

    SELECT MIN(sal) FROM emp;

    SELECT ename, job, sal from emp where sal = 800;

    Ou fazer as duas consultas em uma só:

    SELECT ename, job, sal FROM emp WHERE sal = (SELECT MIN(sal) FROM emp);

    A query interna é executada primeiro e traz o valor necessário para execução da segunda consulta.

SQL – Sub-Queries

- **Subqueries que retornam mais de um valor**
  - ✓ Exemplo: Encontrar os empregados que recebem o menor salário em cada departamento.

```
SELECT ename, sal, deptno FROM emp WHERE sal IN
(SELECT  MIN(sal) FROM emp GROUP BY deptno);
```

A cláusula **IN** substitui o operador **=** para que todas as coincidências sejam retornadas. Se usar o operador **=** terá uma mensagem de erro, dizendo que reorna mais que uma linha.

SQL – Sub-Queries

- **Comparando mais de um valor em uma subquery**
  - ✓ Exemplo:

    SELECT ename, sal, deptno FROM emp WHERE (sal, deptno) IN

    (SELECT MIN(sal), deptno FROM emp GROUP BY deptno);

  - ➢ As colunas são comparadas em pares. O número de colunas e tipo de dados das colunas da cláusula WHERE tem que ser os mesmos nos dois Selects.
  - ➢ Uma query interna deve ser usada dentro de parênteses.
  - ➢ A subquery não pode conter a cláusula Order By.

SQL – Join

- **Em muitos casos é necessário consultar informações que estão em tabelas distintas. Denomina-se Join (junção), o relacionamento entre duas tabelas.**
- **Para conseguir uma consulta que faz junção de duas tabelas relacionadas, é preciso colocar a declaração de junção na cláusula WHERE. Este tipo de Join é conhecido como EQUIJOIN (campos iguais em tabelas diferentes).**
  - ✓ Exemplo: Buscar dados que estão na tabela **Emp**, como o nome do empregado, bem como o nome do departamento em que trabalha, que está na tabela **Dept**.

    SELECT empno, ename, job, emp.deptno Depart, dept.deptno, dname, loc FROM emp, dept **WHERE Emp.Deptno = Dept.Deptno**;

EMPNO	ENAME	JOB	DEPART	DEPTNO	DNAME	LOC
7369	SMITH	CLERK	20	20	RESEARCH	DALLAS
7499	ALLEN	SALESMAN	30	30	SALES	CHICAGO
7521	WARD	SALESMAN	30	30	SALES	CHICAGO

SQL – Join

- **Junções e Condições: Basta acrescentar as condições com o <u>AND</u>**
  - ✓ <u>Exemplo</u>:

    SELECT empno, ename, job, sal, dname, loc FROM emp, dept
    WHERE Dept.Deptno = Emp.Deptno **and** sal > 1500;

EMPNO	ENAME	JOB	SAL	DNAME	LOC
7499	ALLEN	SALESMAN	1600	SALES	CHICAGO
7566	JONES	MANAGER	2975	RESEARCH	DALLAS
7698	BLAKE	MANAGER	2850	SALES	CHICAGO
7782	CLARK	MANAGER	2450	ACCOUNTING	NEW YORK

SQL – Join

- **Diferenças de sintaxe nas versões**
  - ✓ **SQL-86**

    SELECT C.empno, C.ename, C.job, D.dname FROM emp C, dept D
    WHERE D.Deptno = C.Deptno;

  - ✓ **SQL-92**

    SELECT C.empno, C.ename, C.job, D.dname
    FROM emp C **INNER JOIN** dept D **USING**(Deptno);

    *A cláusula **USING** só pode ser usada quando as chaves primária e estrangeira tiverem o mesmo nome, senão, usar a cláusula **ON**.*

    SELECT C.empno, C.ename, C.job, D.dname
    FROM emp C **INNER JOIN** dept D **ON** C.Deptno = D.NumDep;

SQL – Join

- **OUTER JOIN – Join em que valores de uma tabela podem não possuir valores relacionados em outra tabela**
  - ✓ Sintaxe:

    SELECT tabela1.coluna, tabela2.coluna FROM tabela1, tabela2
    WHERE tabela1.coluna = tabela2.coluna **(+)**;

  - ✓ Exemplo:

    SELECT ename, job, dname, emp.deptno, dept.deptno
    FROM emp, dept **WHERE** emp.deptno = dept.deptno **(+)**;

    Acrescenta-se o (+) ao lado da tabela em que faltam elementos

    Além de trazer os empregados que possuem um departamento correspondente na tabela Dept, traz os registros de empregados sem código de departamento.

SQL – Join

- **Diferenças de sintaxe nas versões**
  - ✓ **SQL-86**

    SELECT ename, job, dname, emp.deptno, dept.deptno
    FROM emp, dept **WHERE** emp.deptno = dept.deptno **(+)**;

  - ✓ **SQL-92**

    SELECT ename, job, dname, emp.deptno, dept.deptno
    FROM emp **LEFT OUTER JOIN** dept **USING** (deptno);

    *A cláusula **USING** só pode ser usada quando as chaves primária e estrangeira tiverem o mesmo nome, senão, usar a cláusula **ON**.*

    SELECT ename, job, dname, emp.deptno, dept.deptno FROM emp
    LEFT OUTER JOIN dept **ON** emp.Deptno = Dept.NumDep;

SQL – Join

- **SELF JOIN – Esse tipo de Join é utilizado para implementar a seleção de dados por auto-relacionamento**

  - ✓ Exemplo: Apresentar cada funcionário com seu devido gerente

SQL-86

```
SELECT E.ename, E2.ename Gerente FROM emp E, emp E2
WHERE E.empno = E2.empnoger;
```

SQL-92

```
SELECT E.ename, E2.ename Gerente
FROM emp E INNER JOIN emp E2 on E.empno = E2.empnoger;
```

SQL – Join

- **Exemplo de um SELF JOIN**

Tabela Cursos

COD_CURSO	NOME_CURSO	PRE_REQUISITO
1	Introdução à Lógica de Programação	
2	Fundamentos da Modelagem de Dados	
3	Delphi: Recursos Básicos	1
4	Delphi: Acesso a Banco de Dados	1
5	Oracle: SQL*Plus e SQL	2
6	Oracle: PL/SQL	5

SELECT P.nome_curso, E.nome_curso "Pré-Requisito"
FROM cursos P, cursos E **WHERE** P.pre_requisito = E.cod_curso;

NOME_CURSO	PRE_REQUISITO
Delphi: Recursos Básicos	Introdução à Lógica de Programação
Delphi: Acesso a Banco de Dados	Introdução à Lógica de Programação
Oracle: SQL*Plus e SQL	Fundamentos da Modelagem de Dados
Oracle: PL/SQL	Oracle: SQL*Plus e SQL

SQL – View

- **Uma visão é uma forma pré-determinada de visualizar dados de uma ou mais tabelas, como se fosse uma única tabela.**
- **Uma view é um subconjunto de uma outra tabela ou view que pode ser manipulada como uma tabela.**
- **Uma view não existe fisicamente como uma tabela. Apenas o comando select que define a view é guardado no banco de dados.**

SQL – View

Para que usar Views?

- ***Restringir*** **o acesso a tabela original já que podemos usar a view que contém apenas algumas colunas da tabela.**
- ***Simplificar*** **a estrutura da tabela para os usuários.**
- ***Permitir*** **que grupos de usuários visualizem os dados de diferentes formas.**

SQL – View

- Sintaxe:

  CREATE VIEW Nome_View [nome_coluna [,nome_coluna]] AS
  Select..... WITH CHECK OPTION [CONSTRAINT, CONSTRAINT]
  [WITH READ ONLY];

  sendo:

  - **NOME_VIEW** – nome da visão.
  - **NOME_COLUNA** – especifica nomes para as colunas selecionadas pelas queries da visão. O número de colunas ou aliases deve ser igual ao número de expressões selecionadas pela visão.
  - **SELECT** – é uma cláusula SELECT completa, incluindo JOINS e SUBQUERIES. Não é possível especificar a cláusula Order By.

SQL – View

- **WITH CHECK OPTION** – caso exista algum filtro especificado na cláusula WHERE do comando SELECT, a condição será utilizada para impedir atualizações realizadas diretamente sobre a View que contrariem este filtro. Sem esta cláusula, as alterações poderiam ser feitas, porém não seriam vistas ao listar o conteúdo retornado pela View.

- **CONSTRAINT** – é o nome da CONSTRAINT assinalada ao CHECK OPTION.

- **WITH READ ONLY** – indica que não podem ser executados comandos de DML (insert, delete, update) sobre a view.

SQL – View

Os dados de uma view nem sempre podem ser alterados. Vejamos situações em que as atualizações normalmente são proibidas:

- **Tentativa de inserção em uma view que não possua todos os campos obrigatórios da tabela base (chave primária e não nulos);**
- **Tentativa de atualização de um campo calculado ou chave da tabela primária;**
- **Exclusão de registros que possuam outros relacionados.**

SQL – View

- Exemplo:

```
CREATE VIEW Emp2 AS
Select empno, ename, sal FROM Emp WHERE deptno = 20;
```

- Para selecionar os dados da VIEW criada:

```
SELECT * FROM Emp2;
```

SQL – View

- **Podemos criar views com funções de grupo e dados de duas tabelas. É possível controlar os cabeçalhos dos dados pela inclusão de alias na própria sintaxe do SELECT**

  - ✓ Exemplo:

    CREATE VIEW Resumo_Depto AS
    Select dname Nome, min(sal) "Mínimo", max(sal) "Máximo",
    avg(sal) "Média" FROM Emp, Dept
    WHERE Emp.Deptno = Dept.Deptno Group By dname;

  - ✓ Para selecionar os dados da VIEW criada:

    SELECT * FROM Resumo_Depto;

SQL – View

- **É possível utilizar-se de pseudônimos de colunas (alias) na cláusula *SELECT* da visão, sendo que terá o mesmo resultado no uso de um *SELECT "NORMAL"***

  - ✓ Exemplo:

    CREATE VIEW Emp_Sal (Nome, Cargo, Salario) AS
    Select ename, job, sal FROM Emp;

  - ✓ Para selecionar os dados da VIEW criada:

    SELECT * FROM Emp_Sal;

SQL – View

- **Opção WITH CHECK OPTION – a utilização desta cláusula assegura que os comandos "insert" e "update" não serão validados, caso o efeito deles invalidem a SELECT da visão, isto é, o usuário da visão só poderá manipular dados referentes as linhas que a visão está habilitada a retornar.**
  - ✓ Exemplo:

    CREATE VIEW Emp2 AS

    Select * FROM Emp WHERE deptno = 20 WITH CHECK OPTION;

    A cláusula WITH CHECK OPTION impedirá que, por meio da view, sejam inseridos ou alterados empregados que tenham o número do departamento diferente de 20. Caso isso ocorra, a seguinte mensagem será apresentada:

    ORA-01402: violação da cláusula where da view WITH CHECK OPTION

SQL – View

- **Remover uma visão do Banco de Dados**
  - ✓ Sintaxe:

    DROP VIEW Nome_Visão;

  - ✓ Exemplo:

    DROP VIEW Emp_Sal;

SQL - FIM

APROVEITEM O APRENDIZADO E TENHAM SUCESSO.

BOA SORTE!!!!!!