



VASCO COUTINHO
VILA VELHA

Unidade 3 – Introdução ao SQL

Structured Query Language

Parte 1

Disciplina Projeto de Banco de Dados
Prof^a Me. Renata Cristina Laranja Leite
Curso Técnico em Informática
Módulo II



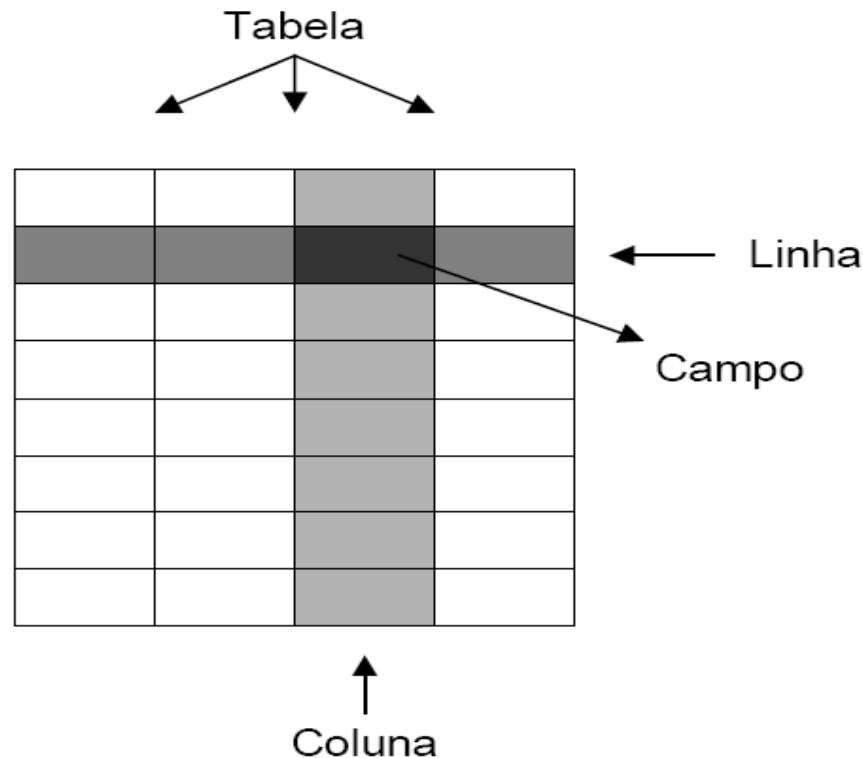
Tópicos da Aula

- **1.** Introdução às Bases de Dados Relacionais
- **2.** Linguagem de Programação SQL
- **3.** DDL - Linguagem para definição de dados



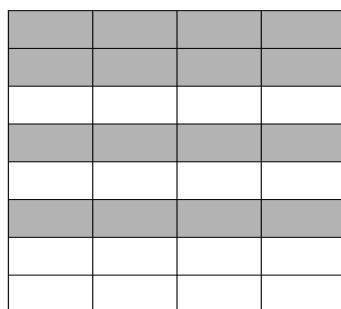
1. Introdução às Bases de Dados Relacionais

Uma base de dados relacional consiste num conjunto de tabelas a duas dimensões. Ao todo existem apenas 4 conceitos a compreender:

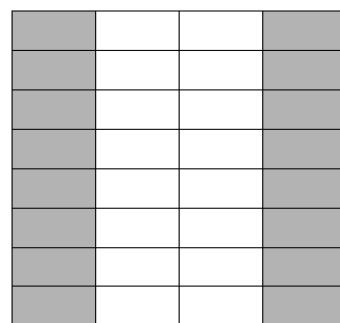


1.1. Relações

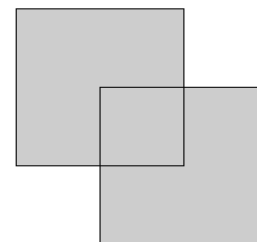
Podem definir-se relações entre as tabelas. As relações podem, por sua vez, ser vistas como novas tabelas. As relações possíveis são:



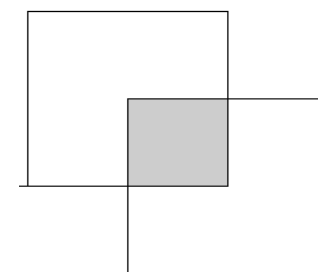
Restrição



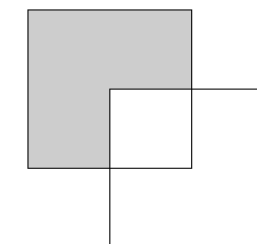
Projectão



União



Intersecção



Diferença

1.2. Propriedades de uma base de dados relacional

- Para o usuário/programador, a base de dados é vista apenas como um conjunto de tabelas;
- Existe um conjunto de operadores para separar e combinar as relações;
- Não existem ligações explícitas entre as tabelas. Elas são feitas unicamente através dos dados;
- Os comandos de extração, inserção e manipulação de dados bem como os de alterações da base de dados estão todos incluídos numa linguagem, a SQL;
- A SQL é uma linguagem com paradigma Funcional ou Declarativo, adaptada à língua inglesa;
- O usuário não sabe ou não precisa saber o formato ou a localização dos dados. Nem tão pouco precisa saber como (que algoritmo é usado para) obter o acesso aos dados.

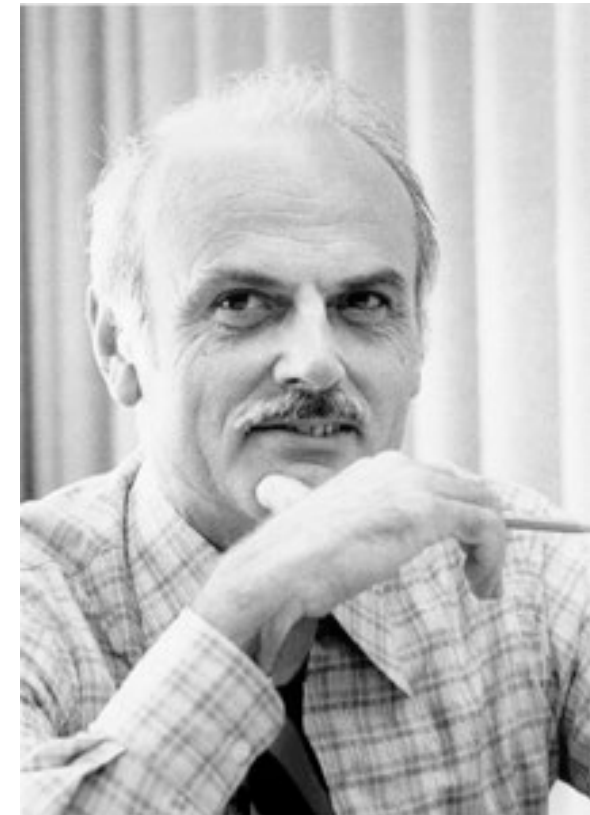
1.3. Propriedades das tabelas

Para cada tabela:

- Não existem registros (linhas) duplicados;
- Não existem nomes de colunas duplicados;
- A ordem das colunas é irrelevante;
- A ordem das linhas é irrelevante;
- Os valores dos dados guardados nos campos são não decomponíveis.

2. Linguagem de Programação SQL

- SQL (Structured Query Language) é uma linguagem de pesquisa de dados declarativa que é utilizada em conjunto com bancos de dados relacionais.
- Essa linguagem foi desenvolvida no início dos anos 70 pela IBM, para demonstrar a viabilidade da implementação do Modelo Relacional proposto por Edgar Frank Codd naquela época.



2.1 Um pouco de história

- O seu nome original era SEQUEL (Structured English Query Language), modificado para que se transformasse em um padrão mundial. Isso aconteceu em 1986 quando a ANSI (American National Standard Institute) padronizou a linguagem e depois em 1987.
- Em 1987 ISO (International Standard Organization) a tornou o SQL um padrão mundial. Isso aconteceu também devido a sua simplicidade e facilidade de uso. É uma linguagem declarativa, onde se determina a forma do resultado esperado e não o caminho para chegar até o resultado (isso diminui o ciclo de aprendizado).
- Apesar dessa padronização em nível mundial, existem muitas variações e extensões que dependem do fabricante do sistema gerenciador de banco de dados. Normalmente, os comandos básicos são sempre os mesmos.

2.1 Um pouco de história (cont.)

- Uma outra característica importante do SQL é que ela é uma linguagem hospedeira, ou seja, para acessar dados dos bancos de dados relacionais, as linguagens de programação permitem que comandos na linguagem SQL sejam “embutidos” no programa que está sendo criado.
- Normalmente, a criação dos bancos de dados e das tabelas fica a cargo de uma determinada pessoa (o DBA – Database Administrator), que tem acesso direto aos recursos do sistema gerenciador de bancos de dados. Já a inclusão e o acesso (consulta) aos dados ficam a cargo de outra pessoa (o desenvolvedor do sistema), que normalmente não poderá pedir para que o usuário acesse diretamente o sistema gerenciador de bancos de dados para realizar os seus trabalhos. Sendo assim, o desenvolvedor necessita “embutir” comandos no seu programa para realizar essas tarefas. Esses comandos são criados utilizando a linguagem SQL.

2.2 Comandos da Linguagem SQL

- São divididos em categorias, de acordo com o tipo de operação que realizam. Os tipos são:
 - **DDL (Data Definition Language)** – comandos que são usados para definir (criar e alterar) bancos de dados e tabelas.
 - **DML (Data Manipulation Language)** – comandos que são usados para inserir dados e para realizar consultas aos dados dos bancos de dados.
 - **DCL (Data Control Language)** - comandos que controlam os aspectos de autorização de dados e licenças de usuários para controlar quem tem acesso para ver ou manipular dados dentro do banco de dados.

2.3 Comandos DDL

- Comandos de criação e exclusão de bancos de dados (database), tabelas (table) e índices (index). Além disso, especificam ligações entre as tabelas e impõem restrições entre as tabelas. Os comandos mais importantes são:
- CREATE DATABASE – cria um novo banco de dados;
- ALTER DATABASE – modifica um banco de dados;
- CREATE TABLE – cria uma nova tabela;
- ALTER TABLE – modifica uma tabela;
- DROP TABLE – exclui uma tabela;
- CREATE INDEX – cria um índice (chave de busca);
- DROP INDEX – exclui um índice;
- CREATE VIEW - cria uma visão de uma ou mais tabelas;
- DROP VIEW - elimina uma visão já criada.

2.4 Comandos DML

Comandos de manipulação dos dados em tabelas. Os comandos mais importantes são:

- INSERT – insere uma nova tupla (linha) em uma tabela existente.
- SELECT – é o comumente mais usado do DML, comanda e permite ao usuário especificar uma query como uma descrição do resultado desejado.
- DELETE – apaga uma ou mais tupla em uma tabela ;
- UPDATE – altera os valores de uma tupla em uma tabela.

2.5 Comandos DCL

Uma subclasse de comandos DML, a DCL (Data Control Language), dispõe de comandos de controle como

- GRANT – autoriza o usuário a executar ou setar operações;
- REVOKE - remove ou restringe a capacidade de um usuário de executar operações.

2.6 Características do SQL

- A SQL não é *case-sensitive*, ou seja, é independente escrever os comandos com letras minúsculas ou maiúsculas ou mesmo misturando ambas. Os nomes das tabelas e das colunas também não são *case-sensitive*. A única coisa que é *case-sensitive* são os dados do tipo caracter.

2.6 Características do SQL (cont.)

Os comandos de SQL bem como os nomes e as colunas das tabelas podem ser escritos tanto em minúsculas como em maiúsculas. Assim temos que:

`SELECT nome, nemp, sal FROM empregado;`

É o mesmo que escrever

`Select Nome, Nemp, SAL from empregado;`

Ou

`seLEct nOMe, NEMP, sal fROM EmPreGado;`


Mas,

`SELECT nome, nemp, sal FROM empregado WHERE nome = 'JORGE';`

já é diferente de ,

`SELECT nome, nemp, sal FROM empregado WHERE nome = 'jorge';`

porque 'JORGE' (e 'jorge') representa um dado do tipo caracter.



Letra
maiúscula é
diferente de
letra
minúscula.

2.6 Características do SQL (cont.)

- Os comandos em SQL terminam apenas com o sinal de ponto e vírgula e podem ocupar mais do que uma linha.
- Assim, qualquer uma das seguintes formas representa o mesmo comando e todas elas estão corretas sintaticamente. No entanto, para facilitar a leitura aconselhamos que se adapte o estilo usado no último exemplo.

```
SELECT nome, nemp, sal FROM empregado WHERE nome = 'JORGE';
```

ou

```
SELECT nome, nemp, sal  
FROM empregado  
WHERE nome = 'JORGE';
```


Tipos de dados mais conhecidos

- **CHARACTER(x) (CHAR)** - representa um string de tamanho x. Se x for omitido, então é equivalente a CHAR(1). Se um string a ser armazenado é menor do que x, então o restante é preenchido com brancos. O *comprimento* máximo é de 255 bytes e o comprimento por omissão é de 1 byte.
- **VARCHAR(*comprimento*)** - Conjunto de caracteres (string) de tamanho variável. O *comprimento* varia entre um máximo de 2000 caracteres e um mínimo de 1. É possível armazenar até 4000 bytes.
- **BINARY LARGE OBJECT (BLOB)** - para armazenar grande quantidade de bytes com fotos, vídeo, áudio, gráficos, mapas etc.
- **INTEGER** - para representar número inteiro.

Tipos de dados mais conhecidos (cont.)

- **NUMERIC(p,s)** - tem uma precisão e uma escala (número de dígitos na parte fracionária) . A escala não pode ser maior que a precisão. Muito usado para representar dinheiro.
- **DECIMAL** - também tem precisão e escala. A precisão é fornecida pela implementação (SGBD).
- **DATE** - armazena ano (quatro dígitos) , mês (dois dígitos) e dia (dois dígitos).

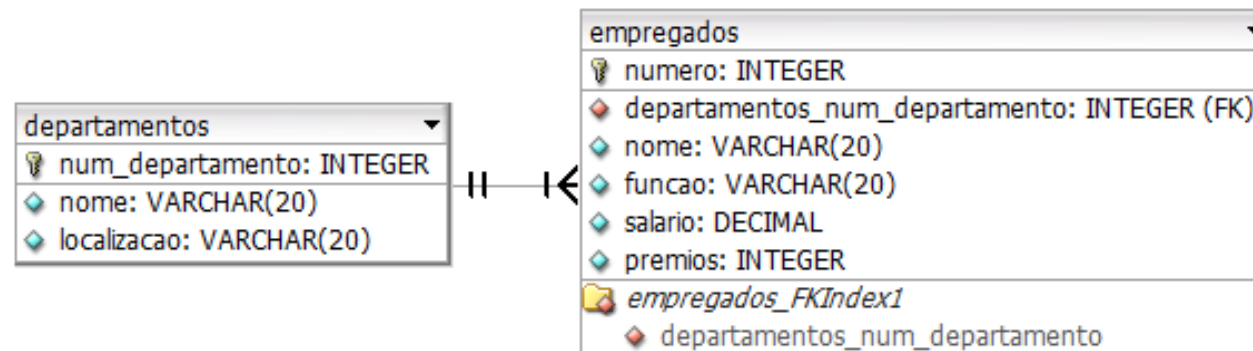
**Obs.: Consulte a documentação do Oracle
para saber sobre outros tipos de dados.**

3. DDL - LINGUAGEM PARA DEFINIÇÃO DE DADOS

- Os comandos do tipo DML são usados frequentemente durante a vida da BD. Os comandos do tipo DDL, por provocarem mudanças significativas na estrutura da BD são de uso muito menos frequente.
- Normalmente, usam-se os comandos DDL na criação e destruição da BD ou para mudanças pontuais da sua estrutura. Pode estabelecer-se um paralelo com a instalação, desinstalação e *upgrades* de *software* para os comandos DDL e uso das funções do *software* para os comandos DML.

Exemplo Prático

- Utilizaremos as tabelas do modelo de dados abaixo para executarmos as instruções SQL.
- As tabelas pretendem modelar uma empresa com uma base de dados muito simples. Existe apenas informação sobre empregados e departamentos. **Todos os empregados pertencem a um departamento.** Essa relação é indicada através da **coluna de chave estrangeira, num_departamento, na tabela empregado**, que indica um valor existente na **coluna de chave primária num_departamento da tabela departamentos.**
- Existe também um **auto-relacionamento na tabela empregados**, onde **todo empregado é subordinado a outro empregado**, exceto **o presidente da empresa.**



3.1.1 Criar Tabelas (CREATE TABLE)

- Uma tabela define-se pelo seu nome, pelo nome das suas colunas, pelo tipo de dados que comporta cada coluna, por restrições de colunas e por restrições de tabela.
- Na realidade existem ainda outras opções avançadas que se podem definir nas tabelas. Essas opções avançadas tem haver com a forma como o servidor guarda a tabela em disco, com parâmetros de otimização de acesso aos dados e outros. Por agora vamos ficar apenas pelas características básicas.

Sintaxe do Comando CREATE TABLE

```
CREATE TABLE < nome_tabela >  
(nome_atributo1 < tipo > [NOT NULL],  
nome_atributo2 < tipo > [NOT NULL],  
...  
nome_atributoN < tipo > [NOT NULL] );
```

onde:

nome_table - indica o nome da tabela a ser criada.

nome_atributo - indica o nome do campo a ser criado na tabela.

tipo - indica a definição do tipo de atributo (integer(n), char(n), decimal (n, m), date...).

Criando um DataBase

Antes de criar tabelas, devemos criar um Banco de Dados. A instrução para a criação de um banco de dados utilizando a linguagem SQL é **CREATE DATABASE**, seguido do nome do banco de dados. A sintaxe do comando de criação de banco de dados obedece a sequência:

CREATE DATABASE <NOME>;

A seguir criaremos o banco de dados TRABALHO com o comando:

CREATE DATABASE EMPRESA;

USE EMPRESA;

Agora criaremos as tabelas que estarão contidas no Banco de Dados EMPRESA.

OBS.: em ambientes gráfico de SGBD este comando é automático.

Criando Tabelas

- A primeira Tabela será a de **Departamentos**. Esta tabela conterà além dos campos também sua chave primária, suas chaves estrangeiras e também seus índices. A segunda tabela será a de **Empregados**, que também será criada.
- Não devemos esquecer de primeiramente abrirmos o Banco de Dados. Diferentemente do que ocorre em alguns aplicativos. No SQL o fato de criarmos um Banco de Dados, não significa que o banco recém criado já está preparado para utilização.

Código de criação de Tabela

- Código necessário à criação da tabela Departamentos:

CREATE TABLE departamentos

(num_departamento INTEGER NOT NULL,

nome VARCHAR(40) NOT NULL,

localizacao VARCHAR(15) NOT NULL,

PRIMARY KEY (num_departamento)

);

Criação da tabela de empregados

```
CREATE TABLE empregados
( numero      INTEGER NOT NULL,
  nome        VARCHAR(40) NOT NULL,
  funcao      VARCHAR(20) NOT NULL,
  salario     DECIMAL(10,2) NOT NULL,
  premios     INTEGER DEFAULT NULL,
  num_departamento  INTEGER NOT NULL,
  PRIMARY KEY (numero),
  FOREIGN KEY (num_departamento) REFERENCES
    departamentos (num_departamento)
);
```

Outra forma para a criação da tabela de empregados

```
CREATE TABLE empregados  
( numero      INTEGER NOT NULL PRIMARY KEY,  
  nome        VARCHAR(40) NOT NULL,  
  funcao      VARCHAR(20) NOT NULL,  
  salario     DECIMAL(10,2) NOT NULL,  
  premios     INTEGER DEFAULT NULL,  
  num_departamento  INTEGER NOT NULL  
  REFERENCES departamentos(num_departamento);  
);
```

Atenção

- A Tabela de Empregados **não** poderia ter sido criada *antes* da Tabela de Departamentos, pois contém uma referência direta àquela tabela.
- Quando declaramos que `num_departamento` é chave estrangeira, promovemos de fato a ligação do cadastro de empregados como o cadastro de departamentos.
- Ao restringirmos as exclusões, permitimos a existência de funcionários não alocados a nenhum departamento. Apesar desta prática ser contrária à tese de que devemos possuir apenas tuplas perfeitamente relacionáveis em nossas tabelas, podemos deixar esta pequena abertura, pois um usuário que excluísse inadvertidamente determinado departamento acabaria por excluir também uma grande quantidade de funcionários, que estivessem ligados a este departamento.

3.1.2 Apagar Tabelas (DROP TABLE)

Este comando elimina a definição da tabela, seus dados e referências.

Sintaxe: **DROP TABLE** < nome_tabela > ;

Para apagar tabelas usa-se o comando DROP TABLE seguido do nome da tabela. Exemplo para apagar a tabela empregado:

DROP TABLE empregados;

Se existirem chaves estrangeiras, noutras tabelas, a apontar para algum dos registros da tabela a apagar, o comando devolve um erro:

DROP TABLE departamentos;

*ORA-02449: unique/primary keys in table referenced by foreign keys

Para forçar o apagamento da tabela, anulando as restrições de integridade necessárias, o comando a executar é:

DROP TABLE departamentos **CASCADE CONSTRAINTS**;

3.1.3 Alterar Tabelas (ALTER TABLE)

- Depois de criar uma tabela é possível alterá-la. Embora não seja um comando usado frequentemente pode ser particularmente importante conseguir alterar a tabela sem ter que a destruir e voltar a criar novamente com outro comando.
- É particularmente importante se a tabela já contiver dados que não se podem perder.

3.1.3 Alterar Tabelas (ALTER TABLE) cont.

As alterações possíveis são:

- Acrescentar colunas;
- Redefinir colunas no que diz respeito ao tipo de dados, comprimento e valor por omissão;
- Acrescentar e retirar restrições;
- Ativar e desativar restrições.

Repare que não é possível remover uma coluna. Note ainda que só é possível diminuir o tamanho dos dados de uma coluna se a tabela estiver vazia.

Acréscentar Colunas

Para acrescentar uma coluna a uma tabela já existente usa-se a seguinte sintaxe:

ALTER TABLE tabela

ADD (coluna tipo_coluna [restricoes] [valor_omissao]);

- **Exemplo 1**– Acrescentar uma coluna a empregados. Definir restrições sobre a coluna bem como valores por omissão.

```
ALTER TABLE empregados
```

```
ADD (sexo CHAR(1) DEFAULT 'F' CHECK (sexo IN ('M', 'F'))
```

```
);
```

Alterar Colunas

- Para alterar colunas (tipo, comprimento dos dados, valores por omissão) usa-se o comando:

ALTER TABLE tabela

MODIFY (coluna [tipo_coluna] [valor_omissao]);

- Note que não é possível com a cláusula MODIFY coluna alterar ou remover restrições sobre essas colunas. Para isso é preciso a cláusula DROP restrição. Os parêntesis são opcionais porque a cláusula MODIFY refere-se sempre a uma e uma só coluna.

Outros exemplos

Exemplo 2 – Alterar o tipo de dados

```
ALTER TABLE empregados
```

```
MODIFY sexo CHAR(3);
```

Exemplo 3 – Adicionar a chave primária da tabela

```
ALTER TABLE departamentos
```

```
ADD primary key(num_departamento);
```

Exemplo 4 – Remover uma coluna da tabela empregado.

```
ALTER TABLE empregados
```

```
DROP sexo;
```

Vamos praticar!

