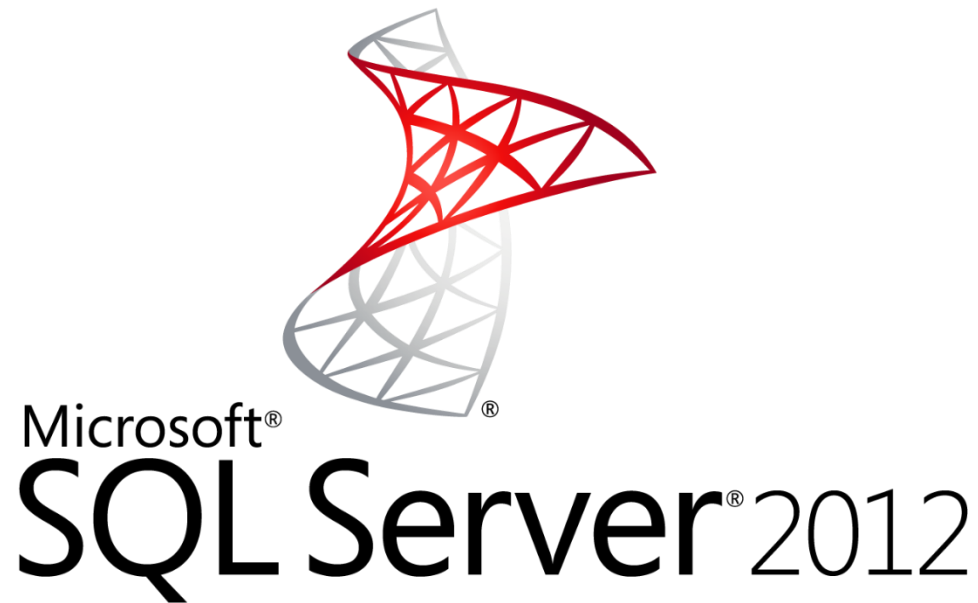




**CWI SOFTWARE**

módulo 2 | banco de dados

**Crescer 2015-2**



# 1 - Comandos SQL - DDL

André Luís Nunes

setembro/2015



# 1 - Comandos SQL - DDL

André Luís Nunes

setembro/2015

# Tópicos

- Comandos **DDL** (Data Definition Language)
  - Estrutura de tabelas (não envolve dados)
- Modelagem ER
- Criação de uma base de dados
  - Documentos relacionados:
    - *01 – Instalando*
    - *02 – Criando database*

# Linguagem SQL

- Linguagem comercial para BD relacional
  - padrão ISO desde a década de 80
    - SQL-1 (86); SQL-2 (92); SQL-3 (99)... SQL-2008
- Base
  - álgebra relacional e cálculo relacional
- Funcionalidades principais
  - definição (DDL) e manipulação (DML) de dados
  - definição de visões e autorizações de acesso
  - definição de restrições de integridade

# Bancos de dados mais utilizados

- Os bancos de dados mais usados são:

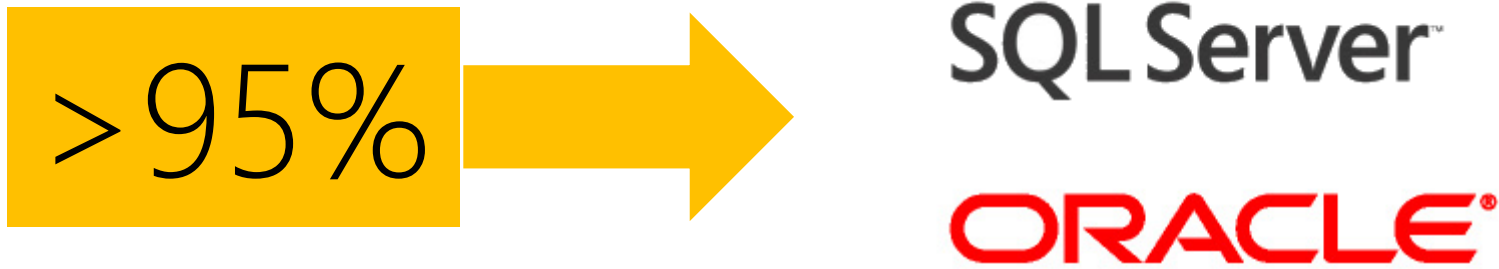
- Oracle
- SQL Server
- MySQL
- Postgre

Microsoft  
**SQL Server™**

**ORACLE®**

# Bancos de dados mais utilizados

- Os bancos de dados mais usados são:



# Banco de dados - conceitos

O termo banco de dados será utilizado para se referir a um Sistema Gerenciador de Banco de Dados (SGBD).

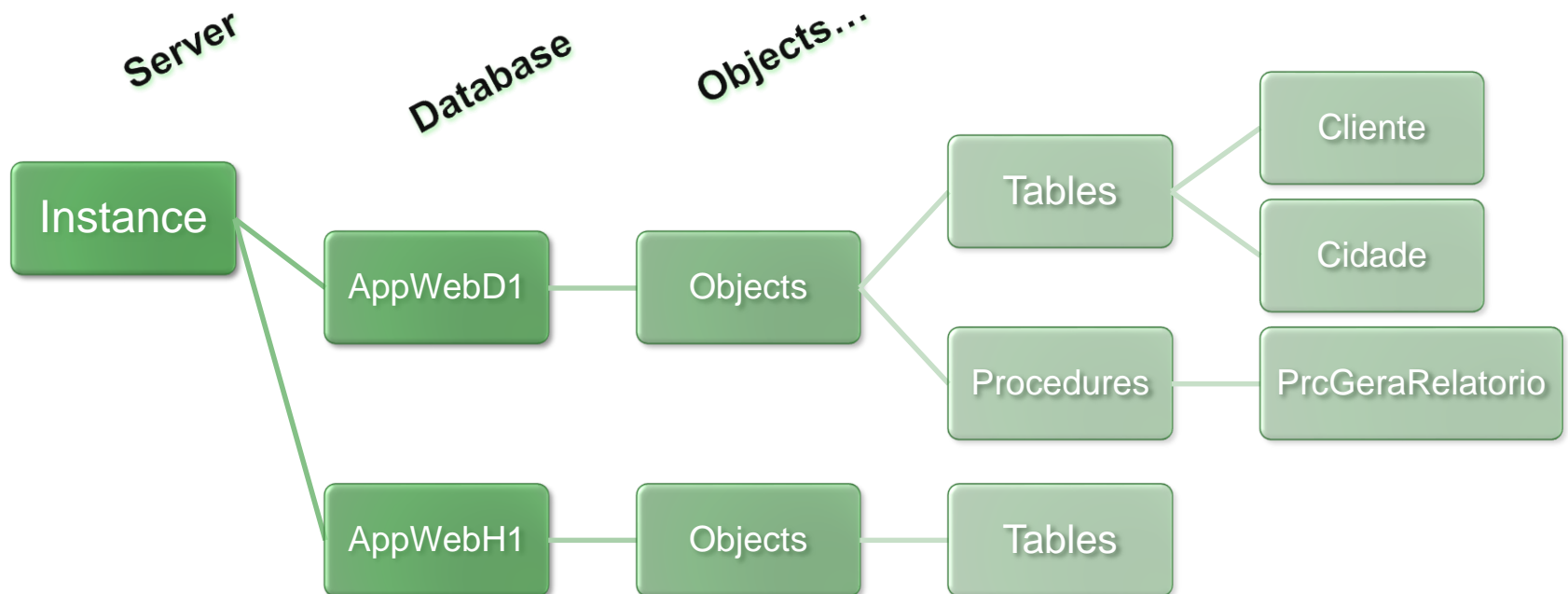
Principais características:

- ❖ Integridade dos dados;
- ❖ Visões consistentes;
- ❖ Acessos simultâneos;
- ❖ Controles de acessos;
- ❖ Transações.



# Banco de dados - arquitetura

Estrutura de organização de um SGBD SQL Server:



# Objetos de um banco de dados

- Todos os **objetos de um banco de dados** tem suas definições salvas no dicionário de dados (metadado) do SGBD.



- ❖ A maioria dos SGBDs disponibiliza uma estrutura de views (consultas) com informações sobre os objetos de cada schema/database.
- ❖ As ferramentas clients também disponibilizam consulta nessa estrutura.

# Objetos de um banco de dados

- Os principais objetos são:

- ❖ **Tables**

- Constraints: primary key, unique key, foreign key e check.

- ❖ Sequences

- ❖ Indexes

- ❖ Views

- ❖ Triggers

- ❖ Procedures

- ❖ Functions

# Objetos - tabela

- Tables (ou tabelas)

- ❑ Estrutura de armazenamento de dados constituída por colunas.

- Exemplo: resultado de uma consulta na tabela Empregado.

IDEmpregado	Nome	DataNascimento	CPF
1	Julio de Castilhos	14/12/1947	21234567895
2	Antonio Augusto Borges de Medeiros	19/03/1942	81234567891
3	Osvaldo Aranha	08/02/1958	01234567893

\*\*\* Lista sendo exibida em ordem alfabética de nome.

- Existem 2 tipos de tabelas:

- **Permanente:** os dados são salvos nos arquivos de dados (grupo de arquivos).

- **Temporária:** os dados são mantidos em memória apenas (área temporária do banco).

# Objetos – tabela > constraints

- Constraints

- ❑ São recursos que permitem estabelecer regras e condições de valores de dados para determinadas colunas de uma tabela.

- **Primary key:** permite determinar uma chave primária para a tabela, estabelece que o conjunto de colunas terá o preenchimento obrigatório e garante que não tenha mais de um registro com o mesmo valor.

- **Unique key:** permite determinar uma chave única para a tabela, estabelecendo que o conjunto de colunas não poderá possuir mais de um registro com o mesmo valor. Permite que a coluna seja opcional.

- **Foreign Key:** permite criar um relacionamento entre 2 tabelas, garantido a integridade dos dados.

- **Check:** permite especificar uma condição para determinada coluna (exemplo: colunaA > 0).

# Objetos – índices

- **Estrutura auxiliar de acesso**

- ❑ Estruturas utilizadas para facilitar o acesso a determinadas informações.
- ❑ Cria ponteiros apontando para os dados armazenados em colunas específicas.

Um exemplo clássico é utilizar um índice de um livro.

# Objetos – sequences

- **Gerador sequencial**

- ☐ Permite a geração de um número sequencial exclusivo para utilização como auto-incremento em uma coluna.
- ☐ Não fica vinculado diretamente à tabela.
  - ☐ Deve ser utilizado diretamente no comando de INSERT; ou
  - ☐ Através de um gatilho (trigger) antes do INSERT.

# Objetos – views

- **Consultas (SQL) salvas no dicionário de dados**
  - ❑ Consultas SQL salvas na base de dados. Permitindo o reaproveitamento de SQL.
  - Não cria nenhuma estrutura a para armazenamento. Irá apenas executar a consulta criada com ela.
  - O resultado da execução de uma view é online, e **não** é uma foto (não confunda).



# Objetos – triggers

- **gatilhos**

- ❑ Gatilhos relacionados a eventos (insert/update/delete) de uma tabela.
- São usados geralmente para criar auditoria sobre operações nas tabelas. Permitindo identificar quem executou determinada operação, quando e quais foram as alterações.

# Objetos – procedures

- **Procedimento armazenado**

- ❑ Blocos de instruções salvos no banco de dados. Permitem o uso de parâmetros de entrada e saída.
- São usados geralmente para geração de relatórios, procedimentos de atualizações, captura de dados, etc.

# Objetos – functions

- **Função armazenada**

- ❑ Estrutura de instruções armazenadas no banco de dados que sempre retornarão um valor.
- São usados geralmente para retornar uma descrição, código ou executar um cálculo.

# Tipos de comandos

## → **DDL:** Data Definition Language

→ Comandos que permitem definir a estrutura de objetos.

## → **DML:** Data Manipulation Language

→ Comandos que permitem alterar os dados de uma tabela.

→ Inserir, alterar, excluir e consultar registros.

## → **DCL:** Data Control Language

→ Comandos que permitem conceder ou revogar permissões à determinados objetos de uma base dados.

# Comandos - DDL - Create

Comandos que definem a estrutura de um objeto, não utilizam transação (não precisam de COMMIT para confirmar).

→ CREATE

→ ALTER

→ DROP

```
Create table <TableName>
(
  <ColumnName> <datatype> (<length>,<scale>) <DefaultValue> <NotNull>,
  <ColumnName> <datatype> (<length>,<scale>) <DefaultValue> <NotNull>,
  <Constraints>
);
```

# Comandos - DDL - Create > Table

Exemplo de criação de uma tabela:

Nome da Coluna	Tipo	Tamanho	Nulo
IDCidade	Numérico inteiro	5	não
Nome	Caractere	30	não
UF	Caractere	2	não

```
Create table Cidade
(
  IDCidade      int          NOT NULL,
  Nome          varchar(30)  NOT NULL,
  UF            varchar(2)   NOT NULL
);
```

Neste exemplo não foi definida a chave primária.

# Comandos - DDL - Create > Table

Exemplo de criação de outra tabela (cliente)

```
Create table Cliente
(
  IDCliente    int          NOT NULL,
  Nome         varchar(30)  NOT NULL,
  Endereco     varchar(35) ,
  Bairro       varchar(35) ,
  IDCidade     int
);
```

O SQL Server não é case-sensitive, porém permite que seja configurado para tal. Tanto para nomes de estrutura quanto dados, tudo dependerá do Collation utilizado na criação da base de dados.

## Comandos - DDL - Create > Table

Autoincremento: definindo valor sequencial para a chave-primária.

A opção **IDENTITY** permite definir uma coluna como autoincremental.

É possível definir um valor inicial e um intervalo (default é 1,1):

**identity (<inicial>, <intervalo>**

```
Create table Cliente
(
  IDCliente    int IDENTITY NOT NULL,
  Nome         varchar(30)  NOT NULL,
  Endereco     varchar(35) ,
  Bairro       varchar(35) ,
  IDCidade     int
);
```

Ao definir essa propriedade à coluna não será possível especificar o valor para a coluna no momento de inserir um novo registro (somente habilitando um modo de inserção sem identity)



# Comandos - DDL > Table (constraints) 1/3

Constraint <nome> <constraint\_type> (<columns>)

- » **Chave Primária (primary key):** permite identificar um registro único na tabela. Normalmente uma coluna numérica, sequencial.
- » **Chave Única (unique):** permite evitar valores (ou combinações) duplicados. Diferente da PK, a coluna não precisa ser obrigatória. Normalmente utilizado em colunas do tipo nome, descrição.

```
Create table Cidade
(
  IDCidade      int          NOT NULL,
  Nome          varchar(30)  NOT NULL,
  UF            varchar(2)   NOT NULL,
  constraint PK_Cidade      Primary Key (IDCidade),
  constraint UK_Cidade_Nome Unique (Nome)
);
```

## Comandos - DDL > Table (constraints) 2/3

Constraint <nome> <constraint\_type> (<columns>)

- » Condição (check): permite definir uma condição para a coluna restringindo valores válidos.

```
Create table Empregado
(
  IDEmpregado int          NOT NULL,
  Nome        varchar(50)  NOT NULL,
  Sexo        char(1)      NOT NULL,
  constraint PK_Empregado  Primary Key (IDEmpregado),
  constraint CC_Empregado_Sexo Check (Sexo in ('F', 'M'))
);
```

## Comandos - DDL > Table (constraints) 2/3

```
Constraint <nome> <constraint_type> (<columnFK>)  
References <table_origem> (<ColumnPK>)
```

- » Chave estrangeira (foreign key): permite criar uma referência para outra tabela, garantindo assim a integridade dos dados.

```
Create table Empregado  
(  
  IDEmpregado int NOT NULL,  
  Nome varchar(50) NOT NULL,  
  Sexo char(1) NOT NULL,  
  IDCidade int NOT NULL,  
  constraint PK_Empregado Primary Key (IDEmpregado),  
  constraint CC_Empregado_Sexo Check (Sexo in ('F', 'M')),  
  constraint FK_Empregado_Cidade Foreign Key (IDCidade)  
    References Cidade (IDCidade)  
);
```



# Comandos - DDL - Alter

Comandos que ALTERAM a estrutura de um objeto:

→ CREATE

→ ALTER

→ DROP

```
Alter table Cidade AddCodigoIBGE Integer;
```

```
Alter table Cidade Add  
Constraint PK_Cidade primary key (IDCidade);
```

## Tipos de alterações mais comuns:

- » Adicionar ou remover: colunas, chave-primária, chave-estrangeira, chave-única, condição (check);
- » Alterar nome e tipo da coluna;
- » Alterar obrigatoriedade da coluna;
- » Alterar valores padrões (default).

# Comandos - DDL - Alter

Comandos que ALTERAM a estrutura de um objeto:

Exemplos:

Alterações de colunas também são permitidas:

```
Alter table Cidade alter column IBGE decimal(8);
```

Renomeando uma coluna

```
exec sp_RENAME 'Cidade.CodigoIBGE' , 'Cod_IBGE', 'COLUMN'
```

Sintaxe para renomear colunas:

```
exec sp_RENAME 'TableName.[OldColumnName]' , '[NewColumnName]', 'COLUMN'
```

Eliminar constraints e colunas também deve ser utilizado o comando ALTER:

```
Alter table Cidade DROP Constraint PK_Cidade;
```

# Comandos - DDL - Drop

Comandos que ELIMINAM a estrutura de um objeto:

→ CREATE

→ ALTER

→ DROP

```
Drop table <table_name>;
```

```
Drop table Cidade;
```

» Deve ser respeitado a relação de integridades (constraints).

# Comandos - Tabela

## Resumo

Formada por colunas com nomes e tipos definidos;

Pode ter 4 tipos de 'regras' (constraints):

- **Primary key**: identificação de um registro (apenas 1 por tabela), ex.: ID
- **Unique key**: identificação de um valor único, ex: CPF
- **Foreign key**: identificação de um relacionamento com outra tabela;
- **Check**: definição de restrição de valores para uma determinada coluna.

Deve ter **PRIMARY KEY** sempre;

# Diagrama Entidade Relacionamento (DER)

- O diagrama ER é equivalente a uma planta de uma construção.



- Assim como na construção civil, nem sempre a planta é construída antes da casa, e nem sempre o modelo está atualizado.

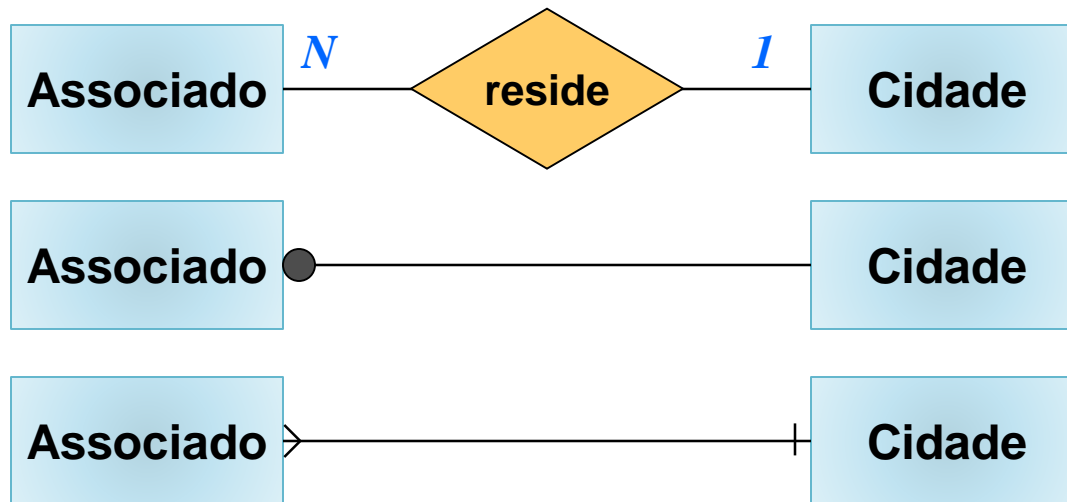


# Diagrama Entidade Relacionamento (DER)

- Itens de um diagrama Entidade-Relacionamento (ER):
  - ENTIDADE: representa normalmente uma tabela no banco de dados.
  - ATRIBUTOS: representa as colunas da tabela.
  - RELACIONAMENTO: identifica a relação entre as entidades.
  - CARDINALIDADE: indica a participação e obrigatoriedade do relacionamento.

# Modelagem de dados (DER)

- Cardinalidade máxima



Notação N pra 1.

Notação tradicional.

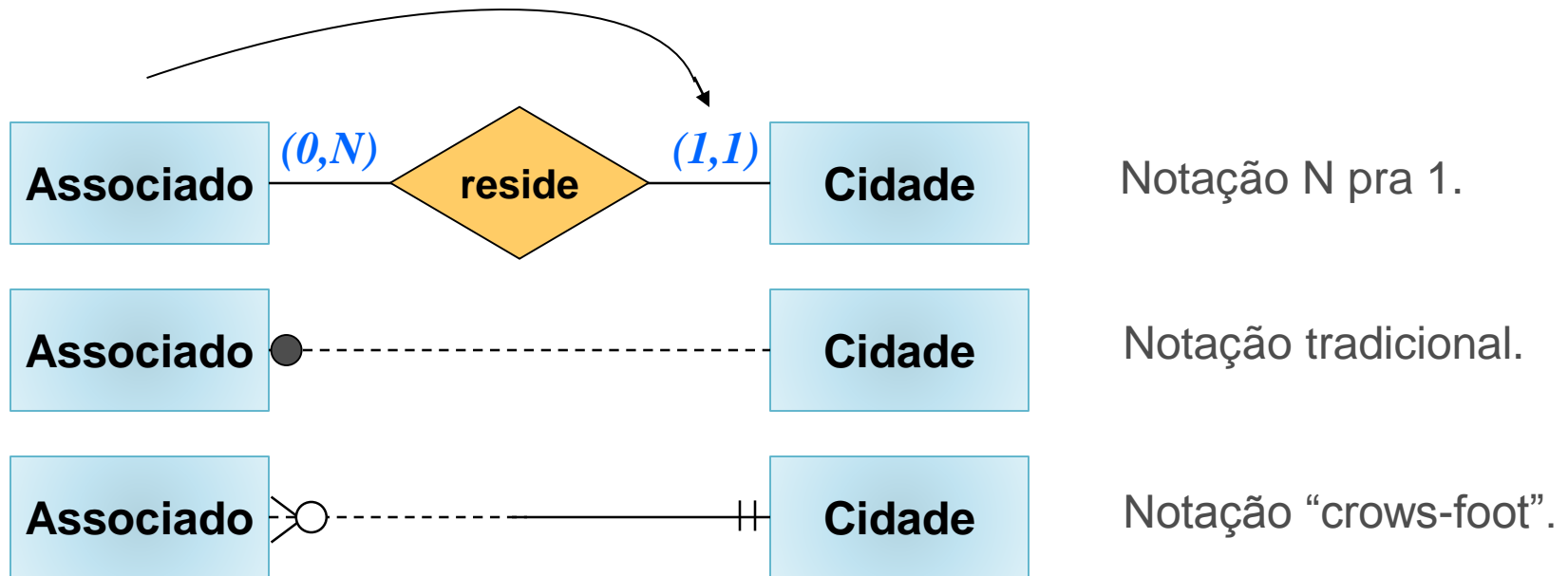
Notação “crows-foot”.

*Um associado reside no máximo em 1 cidade.*

*Uma cidade tem até N associados residindo nela.*

# Modelagem de dados (DER)

- Cardinalidade mínima

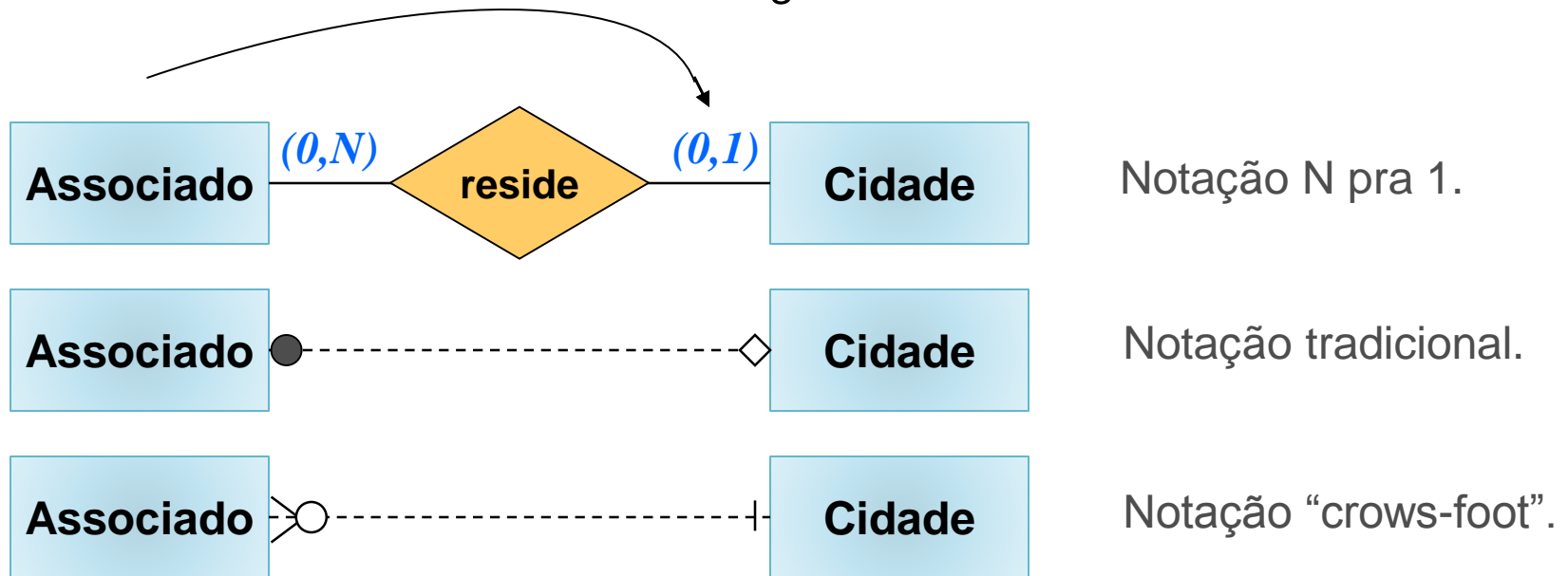


Um associado **obrigatoriamente reside** no máximo em 1 cidade.

Uma cidade **pode ter** até N associados residindo nela.

# Modelagem de dados (DER)

- Cardinalidade mínima, alterando obrigatoriedade:

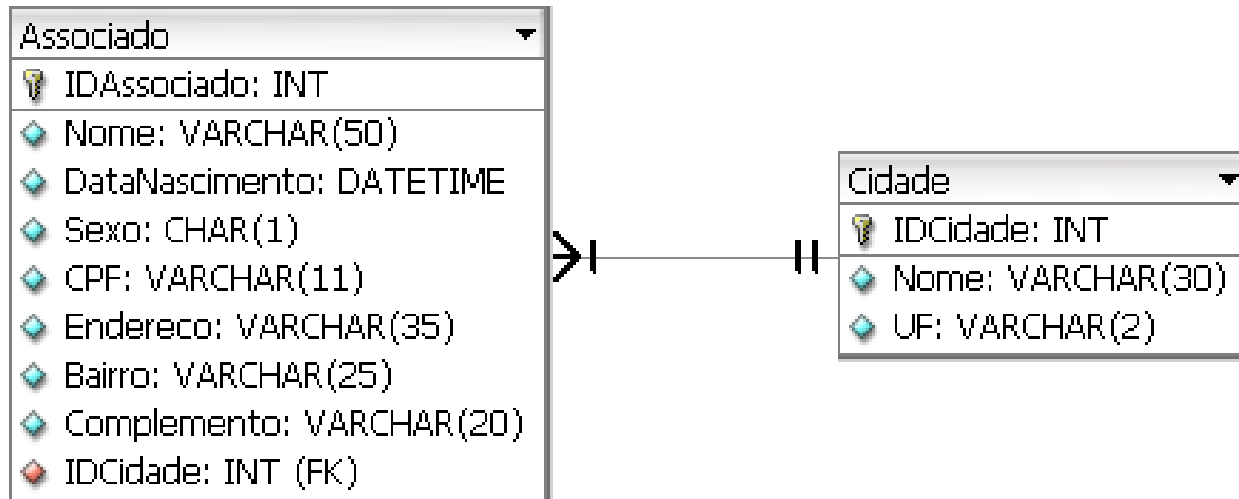


Um associado **opcionalmente reside** no máximo em 1 cidade.

Uma cidade **pode ter** até N associados residindo nela.

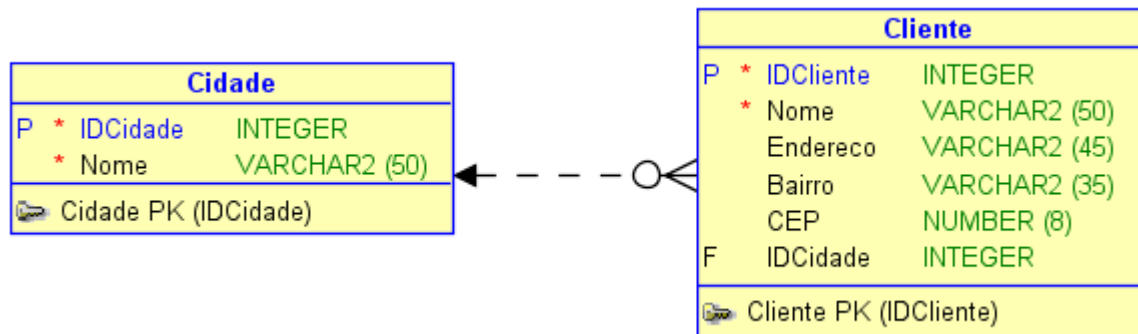
# Modelagem de dados (DER)

- Exemplo da relação entre um Associado e Cidade:



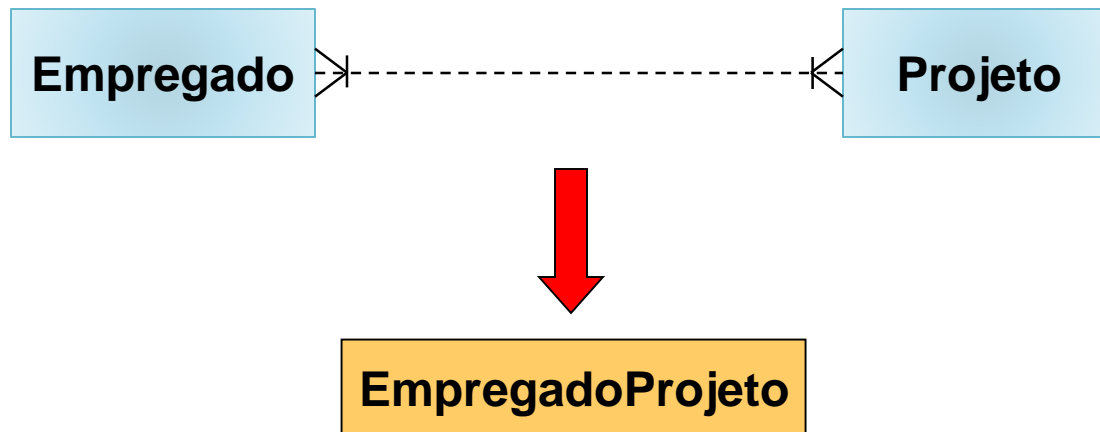
# Modelagem de dados (DER)

- Exemplo da relação entre Cidade e Cliente:



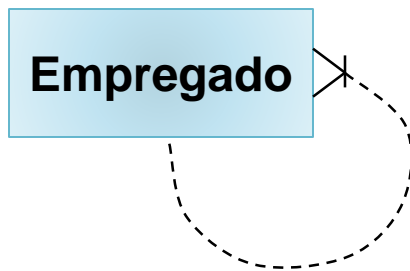
# Modelagem de dados (DER)

- Exemplo de um diagrama, utilizando notação "Crows foot", onde o relacionamento irá gerar uma nova tabela.



# Modelagem de dados (DER)

- Exemplo de uma tabela com auto-relacionamento:



A tabela terá uma coluna opcional, que fará referência para a própria tabela, porém outra coluna e consequentemente outro registro.





# exercícios

**Quais tipos de objetos são utilizados, quando se deseja:**

- 1) armazenar uma lista/cadastro de fornecedores?
- 2) consultar (todos os dias) apenas os clientes ativos?
- 3) consultar (cfe ID) a data do último pagamento de um duplicata?
- 4) efetuar o pagamento de uma mensalidade?
- 5) gerar um sequencial para o ID de uma tabela?
- 6) registrar o histórico de alterações do salário de um associado?
- 7) adicionar uma nova informação ao associado: grau de instrução?
- 8) determinar que a coluna Situacao da tabela Pedido possa ter apenas 5 valores (A, C, S, T, E) ?
- 9) determinar que o nome de uma cidade não poderá ser repetido?
- 10) adicionar a informação sobre a cidade onde o associado trabalha?

# Comandos DDL

## Laboratório

Crie uma base de dados (CursoSQL) para executar os laboratórios.



**mãos à obra**

**Criando as tabelas do script Lab1.sql**

- Cidade
- Associado

# André Nunes

## Núcleo de Tecnologia

andrenunes@cwí.com.br

(51) 3081.3622