

Banco de Dados II DDL

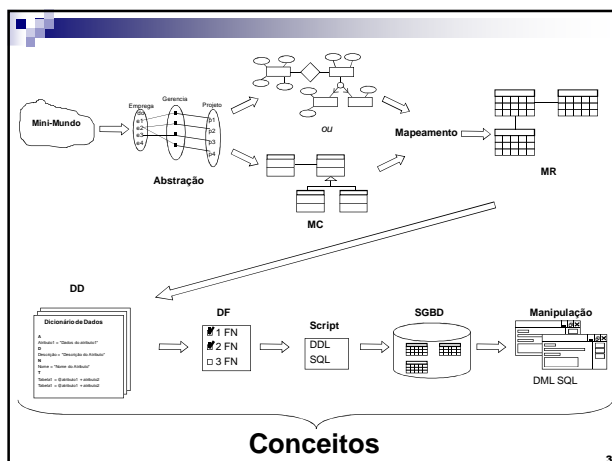
Osmar de Oliveira Braz Junior

1

Objetivos

- Conhecer a sintaxe do SQL DDL.
- Utilizar um terminal para especificar um esquema em um SGBD.
- Converter um MR para um SGBD utilizando SQL DDL.

2



3

6. Linguagem de Definição de Dados

- Para a definição dos esquemas lógico ou físico pode-se utilizar uma linguagem chamada **DDL** (Data Definition Language - Linguagem de Definição de Dados).
- O SGBD possui um compilador **DDL** que permite a execução das declarações para identificar as descrições dos esquemas e para armazená-las em tabelas que constituem um arquivo especial chamado dicionário de dados ou diretório de dados.

4

6.3. Tipos de Domínios em SQL

- **char(n)** –cadeia de caracteres de tamanho fixo, com o tamanho n definido pelo usuário. (**character**).
- **varchar(n)** –cadeia de caracteres de tamanho variável, com o tamanho n máximo definido pelo usuário (**character varying**).
- **int** – é um inteiro (um subconjunto finito de inteiros que depende do equipamento)(**integer**).
- **smallint** é um inteiro pequeno (um subconjunto do domínio dos tipos inteiros que depende do equipamento).
- **numeric(p,d)** – é um número de ponto fixo cuja precisão é definida pelo usuário. O número consiste de p dígitos (mais o sinal), e d dos p dígitos estão à direita da vírgula decimal.
- **real, double precision** - são números de ponto flutuante e ponto flutuante de precisão dupla cuja precisão depende do equipamento.
- **float(n)** - é um número de ponto flutuante com precisão definida pelo usuário em pelo menos n dígitos.
- **date** - é uma data de calendário contendo um ano (com quatro dígitos), mês e dia do mês.
- **time** – a hora do dia, em horas, minutos e segundos.
- **timestamp** - uma combinação de **date** e **time**.
- **clob** - objetos grandes para dados de caractere.
- **blob** - objetos grandes para dados binários. As letras lob significam "Large Object".

5

6.3. Tipos de Domínios em SQL-Oracle

- **Char(n)**- Tamanho Fixo, pode conter uma sequência de 1 a 255 bytes alfanuméricos;
- **Varchar2(n)**- Tamanho Variável, pode conter uma sequência de 1 a 4000 bytes - alfanuméricos.
- **Clob** - pode conter uma sequência até 4 Gigabytes - alfanuméricos.
- **Long**- Tamanho Variável até 2 Gigabytes alfanuméricos nota : só pode existir uma coluna long em cada tabela
- **Number(p,s)** - Numérico com sinal e ponto decimal, sendo precisão de 1 a 38 dígitos
- **Raw** - Binário Variável até 255 bytes
- **Long Raw** - Binário Variável até 2 gigabytes – imagem
- **Date** - Data c/ hora, minuto e segundo
- **Blob** – dado binário até 4 Gigabyte
- **BFile** – dado binário externo até 4 Gigabyte

6

6.3. Tipos de Domínios em SQL-Oracle

■ Estes tipos são convertidos:

- `int` = `NUMBER(38,0)`
- `numeric(12,2)` = `NUMBER(12,2)`
- `varchar(10)` = `VARCHAR2(10)`

7

6.4. Definição de Esquema em SQL

■ Definimos uma relação SQL usando o comando `create table`:

```
CREATE TABLE r (A1 D1,
                A2 D2,
                ...,
                An Dn,
                <regras de integridade 1>,
                <regras de integridade 2>,
                ...,
                <regras de integridade n>)
```

- em que `r` é o nome da relação, cada `Ai` é o nome de um atributo no esquema da relação `r` e `Di` é o tipo de domínio dos valores no domínio dos atributos `Ai`. É possível definir valores default para os atributos, esta especificação é feita depois da especificação do domínio do atributo pela para **default** e o valor desejado.

8

6.4. Definição de Esquema em SQL

■ As principais regras de Integridade englobam:

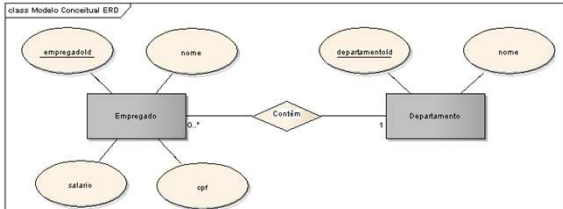
```
PRIMARY KEY(Aj1, Aj2, ..., Ajm)
CHECK (P)
FOREIGN KEY (Aj1, Aj2, ..., Ajm) REFERENCES t
```

- A especificação **PRIMARY KEY** diz que os atributos `Aj1, Aj2, ..., Ajm` formam a chave primária da relação.
- A cláusula **CHECK** especifica um predicado `P` que precisa ser satisfeito por todas as tuplas em uma relação.
- A cláusula **FOREIGN KEY** inclui a relação dos atributos que constituem a chave estrangeira (`Aj1, Aj2, ..., Ajm`) quanto o nome da relação à qual a chave estrangeira faz referência(`t`).

9

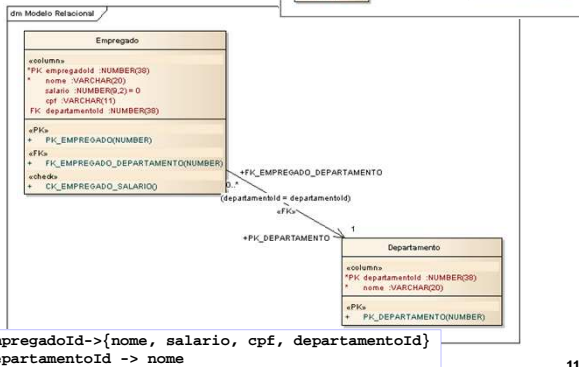
Exemplo

- Uma empresa deseja manter um cadastro de seus empregados com os dados nome, salario e cpf, é criado identificador para diferenciar os empregado. Os empregados estão organizados por departamento.



10

Exemplo



11

Exemplo

```
CREATE TABLE departamento(
    departamentoId INT,
    nome VARCHAR(20),
    PRIMARY KEY(departamentoId)
);

CREATE TABLE empregado(
    empregadoId INT,
    nome VARCHAR(20),
    salario NUMERIC(9,2),
    cpf VARCHAR(11),
    departamentoId INT,
    PRIMARY KEY(empregadoId),
    FOREIGN KEY(departamentoId)
        REFERENCES departamento,
    CHECK (salario >= 0)
);
```

Para uma melhor legibilidade defina somente um atributo ou constraint por linha.

12

6.4. Definição de Esquema em SQL

6.4.1. Esvaziando/Remoção

■ Esvaziar uma Tabela

```
TRUNCATE TABLE <Nome_Tabela>;  
ou  
DELETE FROM <Nome_Tabela>;  
COMMIT;
```

■ Remover Tabelas

```
DROP TABLE <Nome_Tabela>;
```

13

6.4. Definição de Esquema em SQL

6.4.2. Manipulando Atributos

■ Adicionar Atributos

O comando **alter table** permite que o usuário faça a inclusão de novos atributos em uma tabela. A forma geral para o comando **alter table** é a seguinte:

```
ALTER TABLE <Nome_Tabela> ADD  
    <Nome_Atributo> <Tipo_Atributo>;
```

■ Exemplo:

```
ALTER TABLE empregado ADD  
    cidade VARCHAR(30);
```

14

6.4. Definição de Esquema em SQL

6.4.2. Manipulando Atributos

■ Alterar Atributos

O comando **alter table** também permite a alteração de características de atributos de uma tabela. A forma geral para o comando **alter table** é a seguinte:

```
ALTER TABLE <Nome_Tabela> MODIFY  
    <Nome_Coluna> <Tipo_Atributo>;
```

■ Exemplo:

```
ALTER TABLE empregado MODIFY cidade  
    VARCHAR(50);
```

15

6.4. Definição de Esquema em SQL

6.4.2. Manipulando Atributos

■ Renomear Atributos

O comando **alter table** também permite a alteração do nome de um atributo de uma tabela. A forma geral para o comando **alter table** é a seguinte:

```
ALTER TABLE <Nome_Tabela>  
    RENAME COLUMN <Nome_Origem>  
    TO <Nome_Destino>;
```

■ Exemplo:

```
ALTER TABLE empregado  
    RENAME COLUMN cidade TO cidades;
```

16

6.4. Definição de Esquema em SQL

6.4.2. Manipulando Atributos

■ Excluir Atributos

O comando **alter table** também permite a exclusão de atributos de uma tabela. A forma geral para o comando **alter table** é a seguinte:

```
ALTER TABLE <Nome_Tabela>  
    DROP COLUMN <Nome_Atributo>;
```

■ Exemplo:

```
ALTER TABLE empregado  
    DROP COLUMN cidades;
```

17

6.4. Definição de Esquema em SQL

6.4.3. Comandos Adicionais

■ Renomear Tabelas(visões,seqüências ou sinônimos)

```
RENAME <Nome_Origem> TO <Nome_Destino>;
```

■ Listar as tabelas que o usuário é dono

```
SELECT Table_Name FROM USER_TABLES;  
ou  
SELECT Table_Name FROM CAT;
```

■ Listar as tabelas que o usuário tem acesso

```
SELECT Table_Name FROM ALL_TABLES;
```

■ Mostrar a Estrutura de uma Tabela

```
Desc <Nome_Tabela>;
```

■ Mostrar todos os Dados de uma Tabela

```
SELECT * FROM <Nome_Tabela>;
```

18

6.5. Integridade (Constraints)

- No SQL todas as regras de integridade de dados e entidade são definidos por objetos chamados **CONSTRAINT**. Que podem ser definidos quando da criação da tabela ou posteriori via comando **ALTER TABLE**.
- Os constraints suportados são :
 - NOT NULL
 - UNIQUE
 - PRIMARY KEY
 - FOREIGN KEY
 - CHECK

19

6.5. Integridade (Constraints)

6.5.1. Definição de Constraints

- Quando o constraint for definido sem nome, o oracle define um nome para o mesmo - **sys_c00n** - onde n é um número seqüencial crescente.

```
INSERT INTO departamento values(1,'RH');
INSERT INTO departamento values(1,'RH');
```

- Repare na mensagem!

20

6.5. Integridade (Constraints)

6.5.1. Definição de Constraints

- Formas de Definição:

- CONSTRAINTS IN-LINE

Exemplo:

```
CREATE TABLE Cliente
( cpf_cliente VARCHAR(11) Constraint PK_cliente PRIMARY KEY,
  .....
  ..... );
```

- CONSTRAINTS OUT-OF-LINE

Exemplo:

```
CREATE TABLE Cliente
cpf_cliente VARCHAR(11),
.....
.....
CONSTRAINT PK_cliente PRIMARY KEY(cpf_cliente));
```

21

6.5. Integridade (Constraints)

6.5.2. Constraints

- NOT NULL CONSTRAINT

A especificação **NOT NULL** proíbe a inserção de um valor nulo para esse atributo. A inserção de um nulo em um atributo declarado para ser **NOT NULL** gera um erro.

- Exemplo de tabela que todos os campos devem ser preenchidos:

```
CREATE TABLE Cliente (
  cpf_cliente VARCHAR(11) CONSTRAINT NN_cliente_cpf NOT NULL,
  nome_cliente VARCHAR(15) CONSTRAINT NN_cliente_nome NOT NULL,
  rua_cliente VARCHAR(20) CONSTRAINT NN_cliente_rua NOT NULL,
  nome_cidade VARCHAR(15) CONSTRAINT NN_cliente_cidade NOT NULL,
  rg_cliente VARCHAR(7) CONSTRAINT NN_cliente_rg NOT NULL,
  data_nascimento DATE CONSTRAINT NN_cliente_data_nascimento NOT
  NULL);
```

22

Exemplo NOT NULL

```
CREATE TABLE departamento(
  departamentoId INT,
  nome VARCHAR(20) CONSTRAINT NN_DEPARTAMENTO_NOME NOT NULL,
  PRIMARY KEY(departamentoId)
);
```

```
CREATE TABLE empregado(
  empregadoId INT,
  nome VARCHAR(20) CONSTRAINT NN_EMPREGADO_NOME NOT NULL,
  salario NUMERIC(9,2),
  cpf VARCHAR(11),
  departamentoId INT,
  PRIMARY KEY(empregadoId),
  FOREIGN KEY(departamentoId) REFERENCES departamento,
  CHECK (salario >= 0)
);
```

23

6.5. Integridade (Constraints)

6.5.2. Constraints

- UNIQUE CONSTRAINT

Atributos de uma declaração que seja **unique** (isto é, atributos de uma chave candidata) têm a garantia de unicidade deste valor para todas as tuplas. Mesmo a tabela já possuindo chave primária pode-se garantir a unicidade do atributo.

- Exemplo:

```
CREATE TABLE Cliente (
  cpf_cliente VARCHAR(11),
  nome_cliente VARCHAR(15)
  CONSTRAINT NN_cliente_nome NOT NULL,
  rua_cliente VARCHAR(20),
  nome_cidade VARCHAR(15),
  rg_cliente VARCHAR(7),
  data_nascimento DATE,
  Constraint UK_cliente_rg UNIQUE (rg_cliente));
```

24

Exemplo UNIQUE

```
CREATE TABLE departamento(
    departamentoId INT,
    nome VARCHAR(20) CONSTRAINT NN_DEPARTAMENTO_NOME NOT NULL,
    PRIMARY KEY(departamentoId)
);

CREATE TABLE empregado(
    empregadoId INT,
    nome VARCHAR(20) CONSTRAINT NN_EMPREGADO_NOME NOT NULL,
    salario NUMERIC(9,2),
    cpf VARCHAR(11),
    departamentoId INT,
    PRIMARY KEY(empregadoId),
    CONSTRAINT UK_EMPREGADO_CPF UNIQUE(cpf),
    FOREIGN KEY(departamentoId) REFERENCES departamento,
    CHECK (salario >= 0)
);
```

25

6.5. Integridade (Constraints)

6.5.2. Constraints

■ PRIMARY KEY CONSTRAINT

Valor único que identifica cada linha da tabela.

■ Exemplo:

```
CREATE TABLE Cliente (
    cpf_cliente VARCHAR(11),
    nome_cliente VARCHAR(15) CONSTRAINT NN_cliente_nome NOT NULL,
    rua_cliente VARCHAR(20),
    nome_cidade VARCHAR(15),
    rg_cliente VARCHAR(7),
    data_nascimento DATE,
    Constraint UK_cliente_rg UNIQUE (rg_cliente),
    Constraint PK_cliente PRIMARY KEY (cpf_cliente));
```

26

Exemplo PK

```
CREATE TABLE departamento(
    departamentoId INT,
    nome VARCHAR(20) CONSTRAINT NN_DEPARTAMENTO_NOME NOT NULL,
    CONSTRAINT PK_DEPARTAMENTO PRIMARY KEY(departamentoId)
);

CREATE TABLE empregado(
    empregadoId INT,
    nome VARCHAR(20) CONSTRAINT NN_EMPREGADO_NOME NOT NULL,
    salario NUMERIC(9,2),
    cpf VARCHAR(11),
    departamentoId INT,
    CONSTRAINT PK_EMPREGADO PRIMARY KEY(empregadoId),
    CONSTRAINT UK_EMPREGADO_CPF UNIQUE(cpf),
    FOREIGN KEY(departamentoId) REFERENCES departamento,
    CHECK (salario >= 0)
);
```

27

6.5. Integridade (Constraints)

6.5.2. Constraints

■ FOREIGN KEY CONSTRAINT

- Deve estar associada a uma primary key ou unique definida anteriormente.
- Pode assumir valor nulo ou igual ao da chave referenciada.
- Não existe limite para um número de foreign keys.
- Garante a consistência com a primary key referenciada.
- Pode fazer referência a própria tabela.
- Não pode ser criada para views, synonyms e remote table

■ Exemplo:

```
CONSTRAINT FK_conta_cliente
    FOREIGN KEY (cpf_cliente)
    REFERENCES cliente (cpf_cliente)
```

OU

```
CONSTRAINT FK_conta_cliente
    FOREIGN KEY (cpf_cliente)
    REFERENCES cliente
```

28

6.5. Integridade (Constraints)

6.5.2. Constraints

■ FOREIGN KEY CONSTRAINT

■ Exemplo in-line:

```
CREATE TABLE Conta (
    ...
    cpf_cliente VARCHAR(11) CONSTRAINT FK_conta_cliente
        REFERENCES cliente,
    ...
);
```

29

Exemplo FK

```
CREATE TABLE departamento(
    departamentoId INT,
    nome VARCHAR(20) CONSTRAINT NN_DEPARTAMENTO_NOME NOT NULL,
    CONSTRAINT PK_DEPARTAMENTO PRIMARY KEY(departamentoId)
);
```

```
CREATE TABLE empregado(
    empregadoId INT,
    nome VARCHAR(20) CONSTRAINT NN_EMPREGADO_NOME NOT NULL,
    salario NUMERIC(9,2),
    cpf VARCHAR(11),
    departamentoId INT,
    CONSTRAINT PK_EMPREGADO PRIMARY KEY(empregadoId),
    CONSTRAINT UK_EMPREGADO_CPF UNIQUE(cpf),
    CONSTRAINT FK_EMPREGADO_DEPARTAMENTO
        FOREIGN KEY(departamentoId) REFERENCES departamento,
    CHECK (salario >= 0)
);
```

30

6.5. Integridade (Constraints)

6.5.2. Constraints

■ CHECK CONSTRAINT

As validações de colunas são feitas utilizando o CHECK CONSTRAINT.

■ Exemplo:

```
CREATE TABLE cliente (
    nome_cliente VARCHAR(15) NOT NULL,
    rua_cliente VARCHAR(20) NOT NULL,
    nome_cidade VARCHAR(15),
    estado_civil VARCHAR(20),
    idade_cliente INT,
    data_nascimento DATE,
    CONSTRAINT PK_cliente PRIMARY KEY (nome_cliente),
    CONSTRAINT CK_cliente_estado_civil
        CHECK (estado_civil IN ('solteiro', 'casado', 'viúvo')),
    CONSTRAINT CK_cliente_idade
        CHECK (idade_cliente BETWEEN 1 AND 250));
```

31

Exemplo CK

```
CREATE TABLE departamento(
    departamentoId INT,
    nome VARCHAR(20) CONSTRAINT NN_DEPARTAMENTO_NOME NOT NULL,
    CONSTRAINT PK_DEPARTAMENTO PRIMARY KEY(departamentoId)
);
```

```
CREATE TABLE empregado(
    empregadoId INT,
    nome VARCHAR(20) CONSTRAINT NN_EMPREGADO_NOME NOT NULL,
    salario NUMERIC(9,2),
    cpf VARCHAR(11),
    departamentoId INT,
    CONSTRAINT PK_EMPREGADO PRIMARY KEY(empregadoId),
    CONSTRAINT UK_EMPREGADO_CPF UNIQUE(cpf),
    CONSTRAINT FK_EMPREGADO_DEPARTAMENTO
        FOREIGN KEY(departamentoId) REFERENCES departamento,
    CONSTRAINT CK_EMPREGADO_SALARIO CHECK (salario >= 0)
);
```

32

6.5. Integridade (Constraints)

6.5.2. Constraints

■ DEFAULT SPECIFICATION

Podemos atribuir valores **DEFAULT** para colunas, visando facilitar a inserção de dados

■ Exemplo:

```
CREATE TABLE cliente (
    nome_cliente VARCHAR(15) NOT NULL,
    rua_cliente VARCHAR(20) NOT NULL,
    nome_cidade VARCHAR(15),
    estado_civil VARCHAR(20) DEFAULT 'solteiro',
    idade_cliente INT,
    data_nascimento DATE DEFAULT SYSDATE,
    CONSTRAINT PK_cliente PRIMARY KEY (nome_cliente),
    CONSTRAINT CK_cliente_estado_civil
        CHECK (estado_civil IN ('solteiro', 'casado', 'viúvo')),
    CONSTRAINT CK_cliente_idade
        CHECK (idade_cliente BETWEEN 1 AND 250));
```

33

Exemplo DEFAULT

```
CREATE TABLE departamento(
    departamentoId INT,
    nome VARCHAR(20) CONSTRAINT NN_DEPARTAMENTO_NOME NOT NULL,
    CONSTRAINT PK_DEPARTAMENTO PRIMARY KEY(departamentoId)
);
```

```
CREATE TABLE empregado(
    empregadoId INT,
    nome VARCHAR(20) CONSTRAINT NN_EMPREGADO_NOME NOT NULL,
    salario NUMERIC(9,2) DEFAULT 0,
    cpf VARCHAR(11),
    departamentoId INT,
    CONSTRAINT PK_EMPREGADO PRIMARY KEY(empregadoId),
    CONSTRAINT UK_EMPREGADO_CPF UNIQUE(cpf),
    CONSTRAINT FK_EMPREGADO_DEPARTAMENTO
        FOREIGN KEY(departamentoId) REFERENCES departamento,
    CONSTRAINT CK_EMPREGADO_SALARIO CHECK (salario >= 0)
);
```

34

6.5. Integridade (Constraints)

6.5.3. Padronização Nomes

■ Primary Key

FK_<Nome da Tabela>

Ex.: FK_CLIENTE, FK_PRODUTO, FK_PEDIDO

■ Foreign Key

FK_<Tabela Origem>_<Tabela Destino>

Ex.: FK_PEDIDO_CLIENTE, FK_ITEM_PRODUTO

■ Check

CK_<Nome_Tabela>_<Nome_Atributo>

Ex.: CK_CLIENTE_IDADE, CK_CLIENTE_SALARIO

■ Not Null

NN_<Nome_Tabela>_<Nome_Atributo>

Ex.: NN_CLIENTE_NOME, NN_CLIENTE_CPF

■ Unique

UK_<Nome_Tabela>_<Nome_Atributo>

Ex.: UK_CLIENTE_CPF, UK_CLIENTE_RG, UK_USUARIO_LOGIN

35

Exemplo CONSTRAINT

```
CREATE TABLE departamento(
    departamentoId INT,
    nome VARCHAR(20) CONSTRAINT NN_DEPARTAMENTO_NOME NOT NULL,
    CONSTRAINT PK_DEPARTAMENTO PRIMARY KEY(departamentoId)
);
```

```
CREATE TABLE empregado(
    empregadoId INT,
    nome VARCHAR(20) CONSTRAINT NN_EMPREGADO_NOME NOT NULL,
    salario NUMERIC(9,2) DEFAULT 0,
    cpf varchar(11),
    departamentoId INT,
    CONSTRAINT PK_EMPREGADO PRIMARY KEY(empregadoId),
    CONSTRAINT UK_EMPREGADO_CPF UNIQUE(cpf),
    CONSTRAINT FK_EMPREGADO_DEPARTAMENTO FOREIGN
        KEY(departamentoId) REFERENCES departamento,
    CONSTRAINT CK_EMPREGADO_SALARIO CHECK (salario >= 0)
);
```

36

6.5. Integridade (Constraints)

6.5.4. Deleção em Cascata

- Opção a ser utilizada quando da definição do constraint foreign key, para que quando deletamos registros da tabela pai os registros da tabela filho sejam automaticamente deletados.

- Exemplo

```
CREATE TABLE depositante
(.....
  cpf_cliente VARCHAR(11)
  CONSTRAINT FK_depositante_cliente
  REFERENCES cliente ON DELETE CASCADE
  .....);
```

37

6.5. Integridade (Constraints)

6.5.5. Atualização em Cascata

- Quando atualizamos registros da tabela pai os registros da tabela filho sejam automaticamente atualizados. Pode ser utilizada com a exclusão em cascata.

- Exemplo

```
CREATE TABLE depositante
(.....
  cpf_cliente VARCHAR(11)
  CONSTRAINT FK_depositante_cliente
  REFERENCES cliente ON DELETE CASCADE
  ON UPDATE CASCADE
  .....);
```

*não está disponível para oracle

38

6.5. Integridade (Constraints)

6.5.6. Relacionamento Recursivo

- Para criar uma tabela que se auto-relaciona geralmente você precisa primeiro criar a tabela e depois adicionar chave estrangeira a tabela.
- Mas no Oracle você pode adicionar o auto-relacionamento no mesmo script de criação da tabela. Consulte a documentação do seu SGBD para mais detalhes.
- Exemplo de uma tabela com relacionamento recursivo

```
CREATE TABLE Funcionario(
  funcionarioId INT,
  nome_funcionario VARCHAR(20) ,
  cpf_funcionario VARCHAR(11),
  salario_funcionario NUMERIC(9,2),
  supervisorId INT,
  CONSTRAINT PK_funcionario PRIMARY KEY(funcionarioId));

ALTER TABLE Funcionario ADD CONSTRAINT Fk_Funcionario
FOREIGN KEY (supervisorId) REFERENCES Funcionario;
```

39

6.5. Integridade (Constraints)

6.5.7. Operações

- Adicionar Constraints
ALTER TABLE <Nome_Tabela> ADD CONSTRAINT <Nome_Constraint> <Constraint>;
- Excluir Constraint
ALTER TABLE <Nome_Tabela> DROP CONSTRAINT <Nome_Constraint>;
- Renomear Constraint
ALTER TABLE <Nome_Tabela> RENAME CONSTRAINT <Nome_Constraint> TO <Nome_Novo>;
- Ativar Constraints
ALTER TABLE <Nome_Tabela> ENABLE CONSTRAINT <Nome_Constraint>;
- Desativar Constraints
ALTER TABLE <Nome_Tabela> DISABLE CONSTRAINT <Nome_Constraint>;
- Listar Constraints
 - Listando os Nomes das Constraints
SELECT constraint_name FROM user_constraints;
 - Listando os Nomes das Constraints e suas Tabelas
SELECT constraint_name, table_name FROM user_constraints;
 - Listando os Nomes das Constraints seu tipo e suas Tabelas
SELECT constraint_name, constraint_type, table_name FROM user_constraints;

40

6.6. Dicionário de Dados

- O Dicionário de dados não é refletido somente pelos domínios dos atributos em um SGBD.
- Os comentários realizados no dicionário de dados para as tabelas e atributos podem ser armazenados no banco de dados nas suas respectivas tabelas e atributos.
- Os comentários criados de tabelas e colunas são armazenados no dicionário de dados do Oracle em:
 - ALL_COL_COMMENTS
 - USER_COL_COMMENTS
 - ALL_TAB_COMMENTS
 - USER_TAB_COMMENTS
- Utilize ALL para ver os comentários de todas as tabelas e USER para os comentários criados somente pelo usuário em suas tabelas e atributos.

41

6.6. Dicionário de Dados

6.6.1. Comentando Tabela

- Para adicionar um comentário a uma tabela.

```
COMMENT ON TABLE <Nome_Tabela> IS '<Comentario>';
```

- Onde <Nome_Tabela> é a tabela que irá receber o comentário <Comentario>.

- Exemplo:

```
COMMENT ON TABLE Funcionario IS 'Tabela que armazena os dados do funcionario';
```

42

6.6. Dicionário de Dados

6.6.2. Visualizando Comentário Tabela

- Para visualizar o comentário de uma tabela.

```
SELECT COMMENTS
FROM USER_TAB_COMMENTS
WHERE TABLE_NAME = '<Nome_Tabela>';
```

□ Onde <Nome_Tabela> é a tabela que será consultada.

- Exemplo:

```
SELECT COMMENTS
FROM USER_TAB_COMMENTS
WHERE TABLE_NAME = 'FUNCIONARIO';
```

- Exibindo o comentário e sua tabela

```
SELECT TABLE_NAME, COMMENTS
FROM USER_TAB_COMMENTS
WHERE TABLE_NAME = 'FUNCIONARIO';
```

- Exibindo o comentário de todas as tabelas

```
SELECT TABLE_NAME, COMMENTS
FROM USER_TAB_COMMENTS;
```

43

6.6. Dicionário de Dados

6.6.3. Comentando Atributos

- Para adicionar um comentário a um atributo de uma tabela.

```
COMMENT ON COLUMN <Nome_Tabela>.<Nome_Atributo> IS
'<Comentario>';
```

□ Onde <Nome_Tabela> é a tabela que possui o atributo <Nome_Atributo> que irá receber o comentário <Comentario>.

- Exemplo:

```
COMMENT ON COLUMN Funcionario.RG_FUNCIONARIO IS 'Campo
chave primaria da tabela funcionario';
```

44

6.6. Dicionário de Dados

6.6.4. Visualizando Comentário Atributos

- Para visualizar os comentários de atributos de uma tabela.

```
SELECT COMMENTS
FROM USER_COL_COMMENTS
WHERE TABLE_NAME = '<Nome_Tabela>';
```

□ Onde <Nome_Tabela> é a tabela que será consultada.

- Exemplo:

```
SELECT COMMENTS
FROM USER_COL_COMMENTS
WHERE TABLE_NAME = 'FUNCIONARIO';
```

- Exibindo o comentário e o atributo e sua tabela

```
SELECT TABLE_NAME, COLUMN_NAME, COMMENTS
FROM USER_COL_COMMENTS
WHERE TABLE_NAME = 'FUNCIONARIO';
```

45

Exemplo COMMENT

```
CREATE TABLE departamento(
    departamentoId INT,
    nome VARCHAR(20) CONSTRAINT NN_DEPARTAMENTO_NOME NOT NULL,
    CONSTRAINT FK_DEPARTAMENTO PRIMARY KEY(departamentoId)
);
COMMENT ON TABLE departamento IS 'Tabela que armazena os dados de departamento';
COMMENT ON COLUMN departamento.departamentoId IS 'Campo chave de departamento';
COMMENT ON COLUMN departamento.nome IS 'Nome do departamento';

CREATE TABLE empregado(
    empregadoId INT,
    nome VARCHAR(20) CONSTRAINT NN_EMPREGADO_NOME NOT NULL,
    salario NUMERIC(9,2),
    cpf VARCHAR(11),
    departamentoId INT,
    CONSTRAINT PK_EMPREGADO PRIMARY KEY(empregadoId),
    CONSTRAINT UK_EMPREGADO_CPF UNIQUE(cpf),
    CONSTRAINT FK_EMPREGADO_DEPARTAMENTO FOREIGN
        KEY(departamentoId) REFERENCES departamento,
    CONSTRAINT CK_EMPREGADO_SALARIO CHECK (salario >= 0)
);
COMMENT ON TABLE empregado IS 'Tabela que armazena os dados de empregado';
COMMENT ON COLUMN empregado.empregadoId IS 'Campo chave de empregado';
COMMENT ON COLUMN empregado.nome IS 'Nome do empregado';
COMMENT ON COLUMN empregado.salario IS 'Salário do empregado';
COMMENT ON COLUMN empregado.cpf IS 'CPF do empregado';
COMMENT ON COLUMN empregado.departamentoId IS 'Chave estrangeira de departamento';
```

46

Linha de Comando SQL

Comandos em Arquivos

- Salvando Comandos em Arquivo

```
Save <caminho + nome_arquivo>;
```

□ Obs. Extensão Default .SQL

- Carregando Comandos em Arquivo

```
get <caminho + nome_arquivo>;
```

OU

```
@ <caminho + nome_arquivo>;
```

* get somente carrega o conteúdo do arquivo
** @ carrega e executa o conteúdo do arquivo

47

Linha de Comando SQL

Conexão

- Conectando ao Banco

```
connect;
```

ou

```
connect usuario@stringid;
```

ou

```
connect usuario@enderenco_ip;
```

- Desconectando do Banco

```
disconnect;
```

- Saindo da Linha de Comando

```
exit;
```

- Limpando a tela

```
clear screen;
```

48

Linha de Comando SQL Usuário

■ Mostrando o Usuário Conectado

```
SHOW user;
```

■ Selecionando o Nome dos Usuários

```
SELECT Username FROM All_USERS;
```

■ Alterando a Senha do Usuário

```
ALTER USER <nome_usuario>  
IDENTIFIED BY <nova_senha>;
```

49

Bibliografia

■ Principal

- CHEN, P. *The Entity-Relationship Model - Toward a Unified View of Data*. TODS, 1976.
- DATE, C. J. *Introdução a Sistemas de Banco de Dados*. 7. ed. Rio de Janeiro: Campus, 2000. 803 p.
- ELMASRI, S. N.; NAVATHE, B. S. *Sistemas de Banco de Dados: Fundamentos e Aplicações*. 3. ed. Rio de Janeiro: Livros Técnicos e Científicos, 2002. 837 p.
- SILBERSCHATZ, A. ; KORTH, H.F. ; SUDARSHAN, S. *Sistema de Banco de Dados*. 5. ed. Rio de Janeiro: Elsevier, 2006.

■ Complementar

- BEZERRA, E. *Princípios de Análise e Projeto de Sistemas com UML*. 1. ed. Rio de Janeiro: Campus, 2002. 296 p.
- CHIAVENATO, I. *Introdução à Teoria Geral da Administração*. 7. ed. Rio de Janeiro: Campus, 2004. 634 p.
- DAUM, B. *Modelagem de Objetos de negócio com XML: abordagem com base em XML Schema*. Rio de Janeiro: Elsevier, 2004.
- DAMAS, L. *SQL: structured query language*. 1. ed. Rio de Janeiro: LTC, 2007.
- KROENKE, D.M. *Banco de Dados: Fundamentos, Projeto e Implementações*. 6. ed. Rio de Janeiro: Livros Técnicos e Científicos, 1998. 382 p.
- OZSU, M.T.; VALDURIEZ, P. *Princípios de sistemas de banco de dados distribuídos*. 1. ed. Rio de Janeiro: Campus, 2001.
- YOURDON, E. *Análise Estrutura Moderna*. 3. ed. Rio de Janeiro: Campus, 1992. 836 p.

50

Fim

51