

Essencial Database

Linguagem SQL - Comandos DDL

Professora: Lúcia Contente Mós

Tutor: Marcelo Estruc

Introdução

É através da linguagem SQL que interagimos com o SGBD. Essa linguagem é composta de categorias de comandos. São elas DDL (comandos de definição de estruturas), DML (comandos de manipulação de dados), DCL (comandos de controle) e DQL (comandos de consulta).

Conhecer os comandos da linguagem SQL é fundamental para acessar o SGBD, manipular estruturas de armazenamento e seus dados.

Objetivos da aula

- Acessar o ambiente do SGBD Oracle Live;
- Entender e Desenvolver comandos da linguagem SQL;
- Implementar Comandos DDL (Data Definition Language) Create, Alter e Drop Criação, Alteração e Exclusão de Tabelas;
- Implementar de Restrições de Integridade.

Resumo

Acesso ao SGBD

O SGBD que iremos utilizar para realizar as práticas dos comandos será o Oracle. A empresa Oracle mantém um ambiente chamado de Oracle Live, que permanece em nuvem. Torna-se uma excelente opção para o aprendizado, pois para acessar o SGBD e executar os comandos, basta somente ter uma conexão com a Internet. Vamos ver o passo-a-passo, para acessar o ambiente Oracle live.

Digite no seu navegador o nome do ambiente Oracle live, conforme figura abaixo.

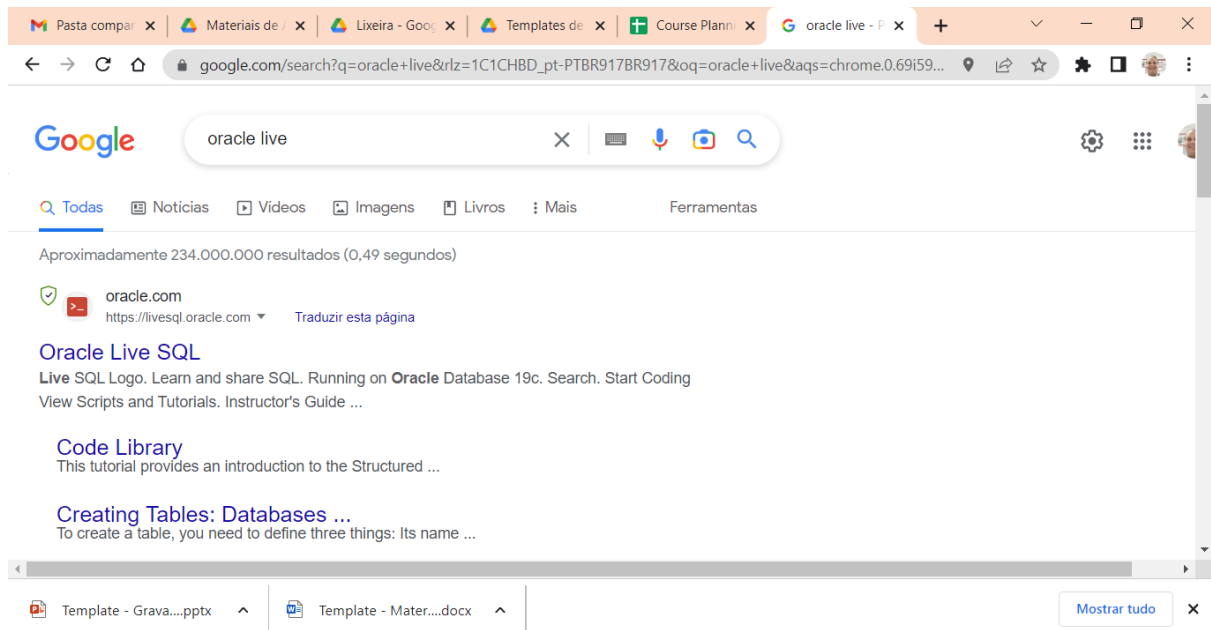


Figura 1 - Acesso ao Oracle Live - Fonte: Autoral, 2023.

Clique no botão Start Coding (procure manter o idioma inglês no seu navegador), pois todos os comandos estão em inglês. Conforme figura abaixo.

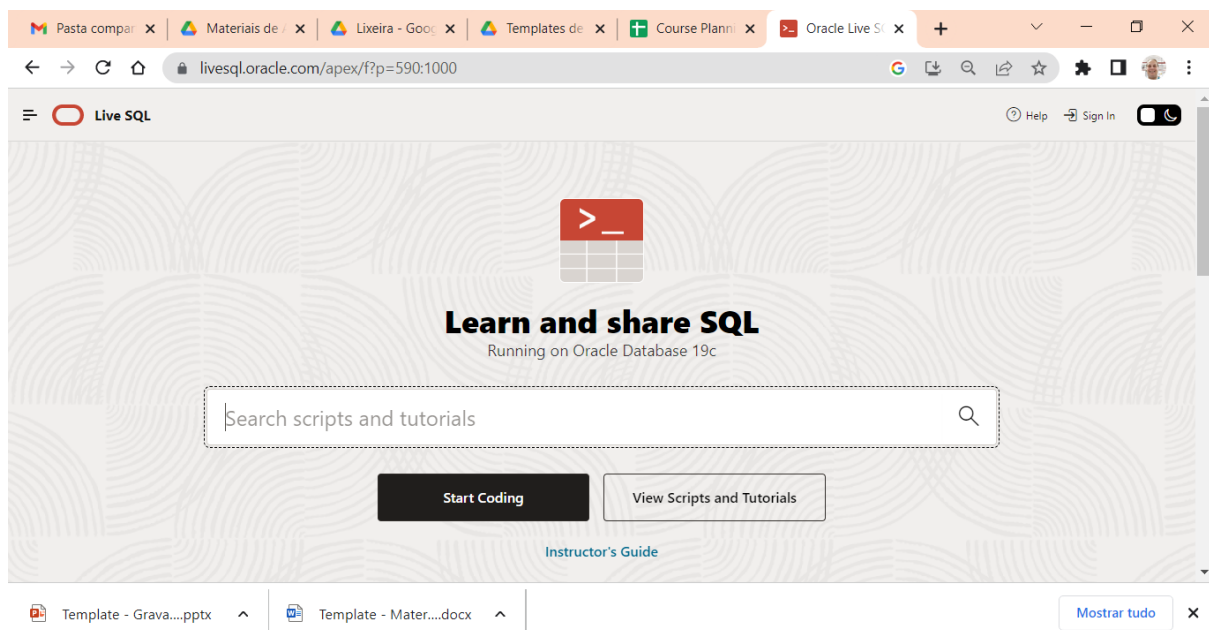


Figura 2 - Acesso ao Oracle Live - Fonte: autoral, 2023.

Depois, é necessário criar uma conta na Oracle, para ter acesso ao ambiente. Para tal, clique no botão Criar conta, conforme figura abaixo.

https://profile.oracle.com/myprofile/account/create-account.jspx

Figura 3 - Criar Conta - Fonte: Autoral, 2023.

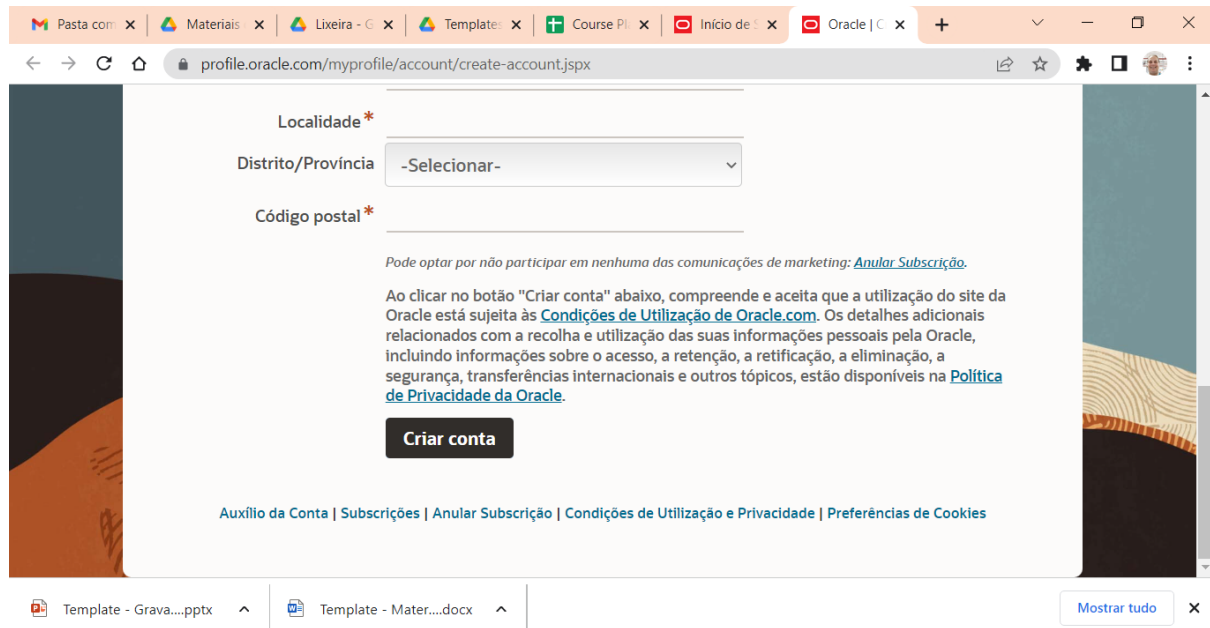
Preencha com os seus dados, todos os campos marcados com *, pois são de preenchimento obrigatório. Atenção para o email e a senha, pois esses dados serão usados para acessar o ambiente.

O seu endereço de e-mail é o seu nome de utilizador.

As senhas têm de incluir letras minúsculas e maiúsculas, pelo menos 1 algarismo e carácter especial, um mínimo de 8 caracteres e não podem coincidir com o seu e-mail nem incluí-lo.

Figura 4 - Preenchendo com os seus dados - Fonte: Autoral, 2023.

Depois de colocar todos os dados solicitados, clique no botão criar conta, conforme figura abaixo.



The screenshot shows a web browser window with the URL `profile.oracle.com/myprofile/account/create-account.jspx`. The page is titled "Criar conta" (Create account) and features a form with the following fields:

- Localidade *** (Location): A text input field.
- Distrito/Provincia** (District/Province): A dropdown menu with the placeholder text "-Selecionar-" (Select).
- Código postal *** (Postal code): A text input field.

Below the form, there is a link: [Pode optar por não participar em nenhuma das comunicações de marketing: Anular Subscrição.](#)

A paragraph of text follows: "Ao clicar no botão 'Criar conta' abaixo, compreende e aceita que a utilização do site da Oracle está sujeita às [Condições de Utilização de Oracle.com](#). Os detalhes adicionais relacionados com a recolha e utilização das suas informações pessoais pela Oracle, incluindo informações sobre o acesso, a retenção, a retificação, a eliminação, a segurança, transferências internacionais e outros tópicos, estão disponíveis na [Política de Privacidade da Oracle](#)."

A black button with the text "Criar conta" is positioned below the text.

At the bottom of the page, there is a footer with links: [Auxílio da Conta](#) | [Subscrições](#) | [Anular Subscrição](#) | [Condições de Utilização e Privacidade](#) | [Preferências de Cookies](#).

Figura 5 - Criar conta - Fonte: Autoral, 2023.

Pronto! Sua conta na Oracle foi criada e agora você já pode acessar o Oracle Live. A Oracle vai enviar um email para que você clique no link e confirme a senha. Depois disso, chame novamente no seu navegador o oracle live, clique no botão start coding e informe seu email e senha que você criou a conta, depois clique no botão iniciar sessão, conforme figura abaixo.

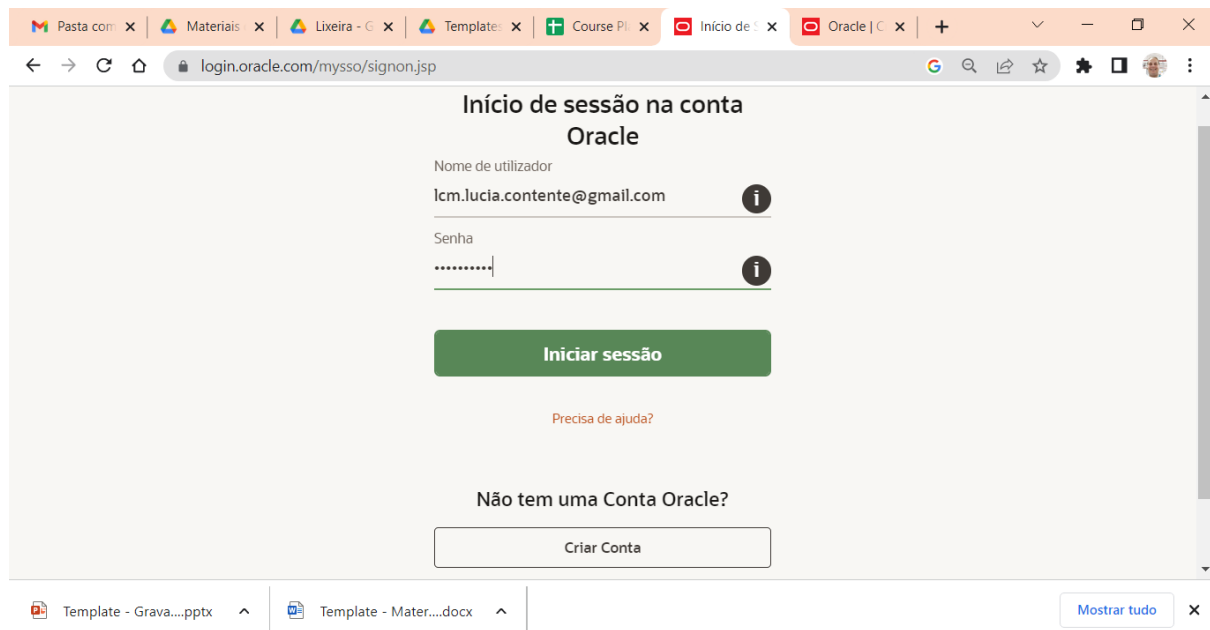


Figura 6 - Iniciar sessão - Fonte: Autoral, 2023.

Esta é a tela onde digitamos os comandos, conforme figura abaixo.

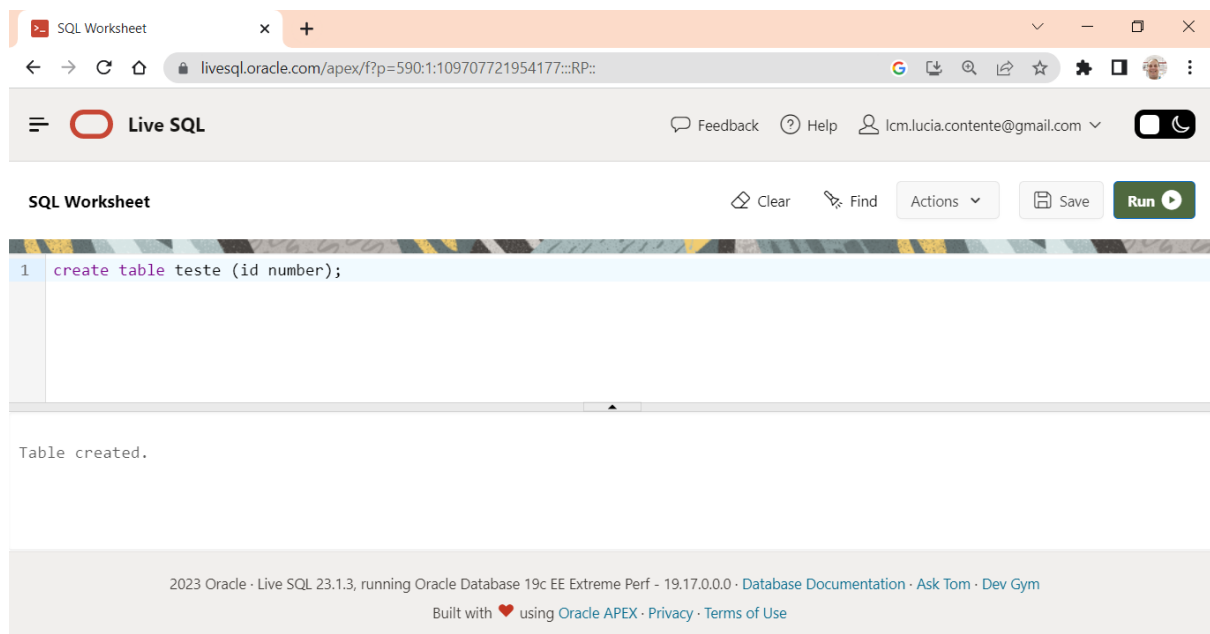


Figura 7 - Tela de Comandos do Oracle Live - Fonte: Autoral, 2023.

Clique no botão run, para executar o comando, conforme figura abaixo.

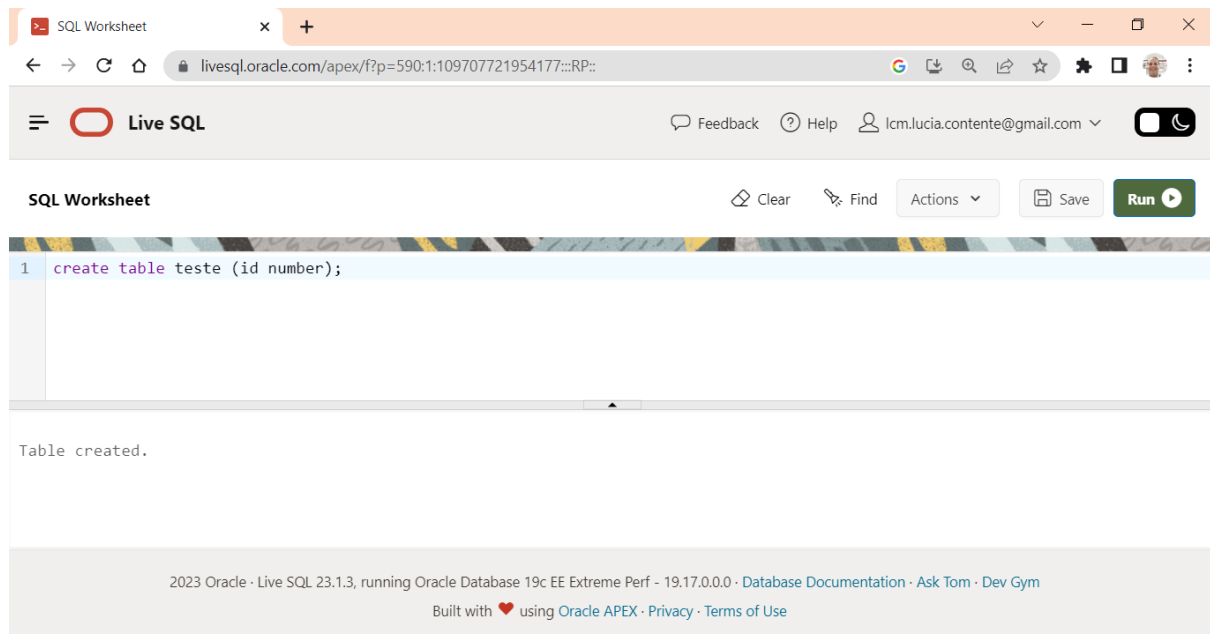


Figura 8 - Execução de comando - Fonte: Autoral, 2023.

DDL – DATA DEFINITION LANGUAGE – TRABALHANDO COM TABELAS

A DDL é a parte da SQL usada para criar e manipular objetos que se relacionam aos dados, especialmente tabelas. Com a DDL você pode criar uma tabela, configurar suas colunas e todas as restrições de integridade, por isso antes de começarmos a escrever nossos primeiros comandos temos que entender bem a linguagem SQL e as restrições de integridade envolvidas.

A DDL está intimamente ligada também a modelagem física do banco de dados, pois é nessa parte do projeto onde os modelos lógicos são derivados para um banco de dados real, ou seja, se tornam físicos (saem do papel).

Vamos começar entendendo a sintaxe do comando que é capaz de criar uma tabela, o CREATE TABLE, que funciona da seguinte forma:

```
CREATE TABLE nome da tabela (  
    nome_da_coluna1 tipo-de-dados restrições,  
    nome_da_coluna2 tipo-de-dados restrições,  
    ...  
);
```

Figura 9 - Criação de Tabela - Fonte: Autoral, 2023.

Para entender melhor este comando, vamos utilizá-lo em um modelo real. A modelagem conceitual pode ser vista na figura 1, onde estamos criando a base de dados de um pequeno jogo, onde cada personagem é de um tipo e possui um ou mais poderes (somente esta parte do jogo). A figura 2 mostra a derivação do respectivo contexto para a modelagem lógica.

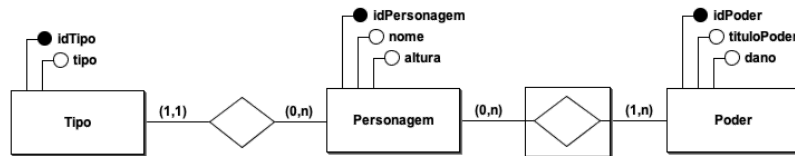


Figura 10 - Modelagem conceitual de exemplo. Fonte: autoral, 2023.

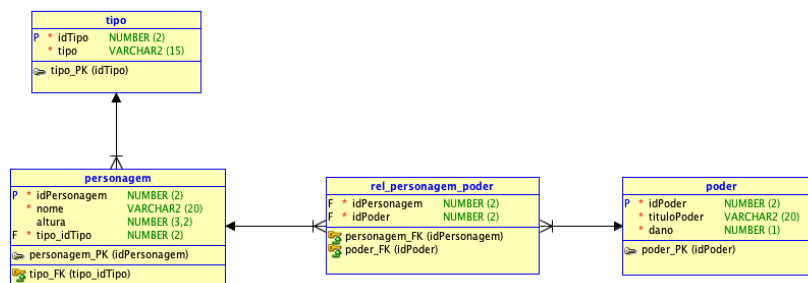


Figura 11 - Derivação do exemplo para modelagem lógica. Fonte: autoral, 2023.

Vamos começar a implementação física deste cenário pela tabela mais simples, ou seja, a que possui menos restrições e não depende de nenhuma outra tabela para existir. Neste caso temos as tabelas “tipo” e “poder”. A tabela “personagem”, por exemplo, possui uma chave estrangeira para “tipo” e, por isso, não poderíamos começar por ela, a não ser que façamos a inclusão da restrição referencial depois, alterando-a.

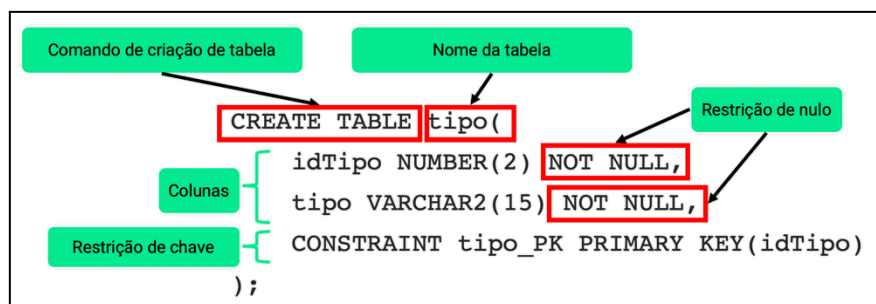


Figura 12 - Criação de Tabela com constraints - Fonte: Autoral, 2023.

Repare que assim como na modelagem lógica, a restrição da chave primária recebe um nome, nesse caso “tipo_pk”. Isso ocorre nas restrições de chave e referencial. Ao se incluir a restrição de chave é necessário informar qual é a coluna que recebe essa restrição, neste caso a coluna “idTipo”. Seguindo a mesma lógica, criamos as próximas tabelas.

```
CREATE TABLE poder(  
    idPoder NUMBER(2) NOT NULL,  
    tituloPoder VARCHAR2(20) NOT NULL,  
    dano NUMBER(1) NOT NULL,  
    CONSTRAINT poder_PK PRIMARY KEY(idPoder)  
);
```

Figura 13 - Criação da tabela Poder - Fonte: GOYA, 2013.

```
CREATE TABLE personagem(  
    idPersonagem NUMBER(2),  
    nome VARCHAR2(20) NOT NULL,  
    altura NUMBER(3,2) NULL,  
    tipo_idTipo NUMBER(2) NOT NULL,  
    CONSTRAINT personagem_PK PRIMARY KEY(idPersonagem),  
    CONSTRAINT tipo_FK FOREIGN KEY(tipo_idTipo) REFERENCES tipo(idTipo)  
);
```

Figura 14 - Criação da tabela personagem - Fonte: GOYA, 2013.

Repare na criação da tabela “personagem” que há a restrição referencial sendo aplicada com o acréscimo da CONSTRAINT do tipo FOREIGN KEY. Lembre-se que toda restrição, no Oracle, deve ter um nome e, no caso da chave estrangeira, é preciso informar qual coluna local a recebe e qual o “alvo”, ou seja, a tabela e sua respectiva coluna de chave primária.

Por fim, vamos criar a tabela de relacionamento entre “personagem” e “poder”, pois como a relação é N:N (cada personagem pode possuir N poderes e cada poder pode estar associado a N personagens), temos que as relacionar através de uma terceira tabela, conforme:


```
CREATE TABLE rel_personagem_poder(
    idPersonagem NUMBER(2) NOT NULL,
    idPoder NUMBER(2) NOT NULL,
    CONSTRAINT personagem_FK FOREIGN KEY (idPersonagem)
        REFERENCES PERSONAGEM(idPersonagem),
    CONSTRAINT poder_FK FOREIGN KEY (idPoder)
        REFERENCES PODER(idPoder)
);
```

Figura 15 - Criação de Tabela de Relacionamento - Fonte: PUGA, 2013.

Repare nas quebras de linhas e a indentação (afastamento da margem) são diferentes para cada comando. Isso quer dizer que não importa se as linhas estão quebradas ou não para a execução do script, mas aqui são descritas assim para maior legibilidade. Essa é, inclusive, uma forte recomendação da comunidade internacional e de qualquer manual de bancos de dados. Repare, ainda, que todos os comandos foram escritos com letras maiúsculas e isso também poderia ter sido diferente, ou seja cada comando poderia ter sido escrito completamente em letras minúsculas, mas por boas práticas de programação escrevemos as palavras reservadas da SQL em letras maiúsculas e o nome dos objetos criados por nós mesmos em letras minúsculas. Por fim, e não menos importante, repare que ao final de cada comando há um “;” que especifica sua finalização.

Se você precisar excluir uma tabela, deve-se utilizar o comando “DROP TABLE nome_da_tabela;”, por exemplo “DROP TABLE tipo;” e a tabela tipo será eliminada do banco de dados. Deve-se utilizar este comando com muito cuidado.

Para se alterar dados de uma tabela, utiliza-se o comando “ALTER TABLE nome_da_tabela MODIFY (modificações);”. A palavra reservada “MODIFY” pode ser substituída por “ADD” se você está adicionando algo na tabela. Por exemplo, se você precisa alterar que a altura do personagem não pode ser nula, deve-se executar o comando “ALTER TABLE personagem MODIFY (altura NOT NULL);” e se você precisar acrescentar uma nova coluna chamada “idade” (que refere-se à idade da personagem), o comando “ALTER TABLE personagem ADD idade NUMBER(2) NULL;” deve ser executado.

Na criação de tabelas, usa-se um comando DDL. Para criar uma tabela é necessário que o usuário tenha privilégio e uma área para armazenamento. A sintaxe simplificada para criação de tabelas é:

Sintaxe: CREATE TABLE [esquema.]tabela

(nome da coluna *tipo do dado* [DEFAULT *expr*]

[*constraint da coluna*],

...,

[*constraint da tabela*]);

onde:

<i>Esquema:</i>	é o nome do proprietário da tabela, quando omitido a tabela é criada no esquema do usuário corrente
<i>Tabela:</i>	é o nome da tabela
<i>DEFAULT expr:</i>	especifica um valor default que será utilizado quando um dado for omitido na inserção
<i>Coluna:</i>	é o nome da coluna
<i>tipo de dados:</i>	é o tipo de dados e o comprimento da coluna
<i>Constraint:</i>	Esta cláusula é opcional e especifica as restrições para a coluna ou para a tabela, quando o nome da constraint é omitido o Oracle assume uma identificação

Convenções para Nomeação de Tabelas e Colunas:

- Deve começar com uma letra
- Pode ter de 1 a 30 caracteres (depende do SGBD)
- Deve conter somente A–Z, a–z, 0–9, _, \$ e #
- Não deve duplicar o nome de outro objeto de propriedade do mesmo usuário
- Não deve ser uma palavra reservada pelo Oracle Server

Tipos de Dados:

Tipo de Dados	Descrição
VARCHAR2(<i>tamanho</i>)	Dados de caractere de comprimento variável
CHAR(<i>tamanho</i>)	Dados de caractere de comprimento fixo
NUMBER(<i>p,s</i>)	Dados numéricos de comprimento variável
DATE	Valores de data e hora
LONG	Dados de caractere de comprimento variável até 2 gigabytes
CLOB	Dados de caractere de um byte de até 4 gigabytes
RAW e LONG RAW	Dados binários brutos
BLOB	Dados binários de até 4 gigabytes
BFILE	Dados binários armazenados em um arquivo externo de até 4 gigabytes

Tabela 1 - Tipos de Dados - Fonte: PUGA, 2013.

No exemplo abaixo está sendo criada a tabela DEPT, com três colunas — chamadas, DEPTNO, DNAME e LOC.

```
SQL>CREATE TABLE dept
2   (deptno NUMBER(2),
3   dname      VARCHAR2(14),
4   loc       VARCHAR2(13));
Table created.
```

A instrução Describe é utilizada para exibir a estrutura de uma tabela.

```
SQL> DESCRIBE dept
```

```
Name          Null?   Type
```

DEPTNO NUMBER(2)
DNAME VARCHAR2(14)
LOC VARCHAR2(13)

RESTRIÇÕES DE INTEGRIDADE

São elas chaves primárias, chaves estrangeiras, colunas de dados únicos, não nulos etc., especialmente na modelagem lógica. Quando uma restrição é “violada” (não é atendida) é muito mais fácil encontrarmos a origem do erro. Os bancos de dados chamam as restrições com seu termo em inglês, ou seja, “CONSTRAINT”.

As restrições de integridade servem para impedir que dados inválidos sejam inseridos numa determinada coluna e são garantidas pelo próprio SGBD (Sistema Gerenciador de Banco de Dados), nós apenas as configuramos na hora de criar os objetos (através da DDL). Podemos, inclusive, dizer que as restrições (CONSTRAINTS) impõem regras nas colunas. Essas restrições são divididas em seis grupos, sendo:

- Restrição de integridade de **domínio**
- Restrição de integridade de **chave**
- Restrição de integridade **nula**
- Restrição de integridade **referencial**
- Restrição de integridade **unicidade**
- Restrição de integridade **de valor padrão**

A integridade de domínio especifica qual **tipo de valor** um determinado atributo pode admitir, ou seja, é são os tipos de dados (datatypes). Por exemplo, colunas configuradas como número inteiro, número de ponto flutuante, texto, data com hora, somente data, etc. Se um dado do tipo texto for inserido em uma coluna que aceita apenas valores numéricos, a restrição de integridade de domínio será violada. A integridade de domínio pode ir além, podendo **checar** se um valor inserido está dentro de uma especificação. Para isso, utiliza-se a restrição

“CHECK”. Por exemplo, o estado no cadastro de um endereço, deve ser uma das 27 opções; o sexo de uma pessoa etc.

A integridade de vazio especifica se o valor de um atributo **pode ou não ser vazio**, ou seja, nulo. São os famosos NULL (permite que um dado seja vazio) e NOT NULL (não permite que um dado não seja especificado naquela coluna na hora de inserir uma linha). A integridade de vazio é também associada à cardinalidade mínima nos relacionamentos (0, a FK pode ser nula. 1, a FK não pode ser nula).

Já a integridade de chave (ou como alguns autores chamam, integridade de chave primária) garante que os valores das chaves primárias sejam sempre únicos e não nulos, pela própria definição de chave primária. Por exemplo, se tentarmos inserir uma linha cuja chave primária já exista, haverá violação da restrição de chave primária.

A integridade referencial é a integridade das FKs (chaves estrangeiras – *Foreign Keys*). Ela garante que o valor que aparecer nos atributos de uma chave estrangeira deve sempre aparecer na chave primária da tabela referenciada, ou seja, garante que todo valor de uma FK seja o mesmo de uma PK, a qual a coluna em questão aponta. Assim sendo, sempre que uma coluna é definida como FK, o SGBD deve garantir que ela sempre aponte para uma PK. Na hora que estamos criando as colunas FKs devemos dizer qual é a tabela alvo e sua respectiva chave primária.

A integridade de unicidade é aquela restrição que permite, ou não, que o valor se repita. No banco de dados utilizamos a palavra reservada UNIQUE quando queremos que seus dados sejam únicos, como por exemplo, o RG de uma pessoa, o e-mail de um usuário, os dados de uma impressão digital etc. Não confunda a restrição de unicidade com a de chave primária, pois os valores dessa coluna não são necessariamente identificadores da tabela.

Por fim e não menos importante temos a integridade de valor padrão associada à palavra reservada DEFAULT. Essa restrição permite que se um valor nulo for inserido em uma coluna (ou não foi informado, por exemplo) um valor padrão (pré-definido) é colocado em seu lugar. Por exemplo, se o país não foi informado numa tabela de endereços incluir “Brasil”. Essa restrição está fortemente ligada à restrição de nulo.

É muito importante reforçar que todas as restrições de integridade são garantidas pelo próprio SGBD. Uma restrição que não se encaixa em nenhuma das

citadas aqui é chamada de restrição semântica e deve ser garantida pelo software que utiliza o banco de dados. Por exemplo, a cardinalidade máxima de um relacionamento ou, ainda como exemplo real, imagine que temos um relacionamento entre funcionários e departamento, onde um funcionário de recursos humanos não poderia assumir o cargo de analista de sistemas. Essa é uma restrição que deve ser garantida pela própria aplicação.

Conteúdo bônus

Como pode-se observar, são muitos comandos que compõem a linguagem SQL. Por isso, é sempre bom ter acesso a um guia rápido de comandos, uma espécie de “guia de bolso” dos principais comandos da linguagem SQL e em que situação usá-los. O objetivo desse conteúdo extra, é fornecer esse tipo de guia, para facilitar as buscas pelos comandos.

Comandos	Exemplo
Criação de Tabela	<pre>CREATE TABLE dept1(deptno NUMBER(2), dname VARCHAR2(14), loc VARCHAR2(10));</pre>
Visualizar a estrutura da tabela	<pre>describe dept1;</pre>
Adicionar um campo a tabela	<pre>alter table dept1 add funcao varchar2(30);</pre>
Adicionar uma constraint de chave primária	<pre>alter table dept1 add constraint pk_deptno primary key (deptno);</pre>

Implementar uma tabela associativa	<pre>CREATE TABLE PROFESSOR(COD_PROF NUMBER(2), NOME_PROF VARCHAR2(50)); CREATE TABLE DISCIPLINA(COD_DISC NUMBER(3), NOME_DISC VARCHAR2(50)); CREATE TABLE PROF_DISC(COD_PROF NUMBER(2), COD_DISC NUMBER(3)); ALTER TABLE PROFESSOR ADD CONSTRAINT PK_COD_PROF PRIMARY KEY (COD_PROF); ALTER TABLE DISCIPLINA ADD CONSTRAINT PK_COD_DISC PRIMARY KEY (COD_DISC);</pre>
Adição de uma chave primária composta	<pre>ALTER TABLE PROF_DISC ADD CONSTRAINT PK_CODS PRIMARY KEY(COD_PROF,COD_DISC);</pre>
Adição de uma chave estrangeira	<pre>ALTER TABLE PROF_DISC ADD CONSTRAINT FK_COD_PROF FOREIGN KEY(COD_PROF) REFERENCES PROFESSOR(COD_PROF); ALTER TABLE PROF_DISC</pre>

	ADD CONSTRAINT FK_COD_DISC FOREIGN KEY(COD_DISC) REFERENCES DISCIPLINA(COD_DISC);
--	--

Tabela 2 - Principais Comandos - Fonte: Autor, 2023.

Referências Bibliográficas

DATE, C. J. **Introdução a sistemas de banco de dados**. Rio de Janeiro. Ed. Campus, 1991.

CHEN, Peter. **Modelagem de dados: a abordagem entidade-relacionamento para projeto lógico**. São Paulo: Makron Books, 1990.

MEDEIROS, L. F. **Banco de dados, princípios e práticas**, 1ª. ed., Ed. Intersaberes, 2013.

PUGA, S., França E., GOYA M., **Banco de dados: Implementação em SQL, PL/SQL e Oracle 11g**, Ed. Pearson, 2013.

ELMASRI R., NAVATHE, S. **Sistemas de Banco de Dados**, 4ª ed. Ed. Pearson, 2005.