Unidade 5: Normalização

1. Nesta Aula

Necessidade de Normalização
Processo de Normalização e Desnormalização
Ferramentas Case
Padronização: Trigramação e Qualificador nome descritivo
Introdução à Linguagem SQL

Metas de Compreensão

Conhecer as etapas de Normalização. Compreender o processo para colocar um projeto na forma Normal e quando desnormalizar. Conhecer algumas ferramentas case e aprender os padrões trigramação e o qualificador nome descritivo.

Apresentação

O objetivo nesta aula é tratar das formas normais do projeto e modelagem de um Banco de Dados Relacional. Você deve entender o processo de normalização e quando você deve criar um banco desnormalizado. Conhecer algumas ferramentas case o ajudará a se adaptar rápido a diferentes empresas. Também você notará que existem alguns padrões para Banco de Dados que podem lhe ajudar na modelagem.
Para entender estes conceitos, esta unidade está organizada da seguinte forma:

- a seção 2 apresenta as formas normais de um banco de dados;
- a seção 3 mostra a ferramentas case;
- a seção 4 apresenta os conceitos da padronização trigramação e qualificador nome descritivo;
- a seção 5 você vai iniciar a Linguagem SQL;

Não deixe de utilizar as bibliografias associadas à unidade. Bom estudo!

2. Formas Normais

A técnica de normalização de um banco de dados permite mostrar que os bancos de dados relacionais apresentam vantagens no tratamento de dados. A normalização não permite que se crie redundância de dados, evita tratar os dados como o sistema de arquivos, evita criar anomalias e que se misture assuntos. A normalização otimiza a alocação de espaço, agrupa dados de forma que cada dado seja dependente de sua chave primária, prove um meio formal de se estruturar a informação, ajuda a diluir duvidas do projeto. As técnicas de levantamento de requisito, analise e modelagem vista até esta aula permite criar uma boa estrutura de dados, mas as vezes, pode-se criar uma estrutura ruim de dados. Para reconhecer e classificar as estruturas de dados existe a normalização. Normalização é um processo para refinar e corrigir as estruturas de dados propostas que atua por meio de uma série de estágios chamados Formas Normais (FN). Os três primeiros estágios para a maioria dos projetos comerciais são suficientes, mas existe cinco formas normais disponíveis. Vale ressaltar que quanto maior a normalização menos espaço em HD o banco vai ocupar e maior será a demanda de processamento, uniões, relacionamentos etc. Embora a normalização seja importante no banco de dados, não presuma que todos os projetos devem atingir o maior nível de normalização pois você deve avaliar a demanda de desempenho e em alguns casos desnormalizar um banco de dados para uma Forma Normal (FN) mais baixa. O processo de normalização causa a simplificação de atributos de uma entidade tornando o banco de dados mais estável e de fácil manutenção.

Etapa 1: Na 1º FN você deve eliminar os grupos de repetição

Ao apresentar os dados em formato de tabela, elimine os valores nulos e separe os grupos de repetição. Para facilitar o entendimento veja a figura 01 com suas tuplas, representando o relatório de matrículas da Faculdade. Observe que existem colunas agrupadas, indicando a

existem de dados que se repetem. Neste caso a ideia é que a partir do relatório, você consiga modelar as entidades e os atributos de maneira correta.

FACULDADE DAS AMÉRICAS

Relatório de Matrículas

nome_aluno	rgm	nome_pai	nome_mae	nascimento	sexo	data_matricul	semestre	turma	disciplinas
Julia Izabel Ribeiro	25100	Luciano Alves Ribeiro	Josiane Ribeiro	25/05/1995	F	10/01/2017	1º	A	Lógica e Técnicas de Programação
									Arquitetura de Computadores
									Introdução a Inteligência Artificial
Amanda Esposito	25101	Edson Franco Esposito	Luciana Antunes Espotito	24/02/1996	F	12/01/2017	2º	A	Banco de Dados
									Sistemas Operacionais
									Redes Neurais
Roberto Rodrigo Prado	25102	Carlos Eduardo Prado	Eliana Pioto Prado	15/01/1994	M	10/01/2017	1º	A	Lógica e Técnicas de Programação
									Arquitetura de Computadores
									Introdução a Inteligência Artificial
Fernando Franchini	25103	Roberto Serafin Franchini	Daniela Andrade Franchini	10/02/1990	F	11/01/2017	1º	A	Lógica e Técnicas de Programação
									Arquitetura de Computadores
									Introdução a Inteligência Artificial
Nerson Martins	25104	Durval Pilon Martins	Clemildes Santos Martins	16/02/1992	M	09/01/2017	1º	A	Lógica e Técnicas de Programação
									Arquitetura de Computadores
									Introdução a Inteligência Artificial
Raphael Lucas Fernandes	25105	Wendel Garcia Fernandes	Andiara Rodrigues Fernandes	30/10/1993	M	18/01/2017	1º	A	Lógica e Técnicas de Programação
									Arquitetura de Computadores
									Introdução a Inteligência Artificial
Adriana Bonfieti Moraes	25106	Ewerton Pereira Moraes	Elena Sagio Moraes	05/05/1994	F	10/01/2017	1º	A	Lógica e Técnicas de Programação
									Arquitetura de Computadores
									Introdução a Inteligência Artificial
Berenice Amaral Martins	25107	Pedro Santos Pereira	Gabriela Aparecida Pereira	01/06/1995	F	09/01/2017	2º	A	Banco de Dados
									Sistemas Operacionais
									Redes Neurais
Alan Jacinto Patricio	25108	Franklin Bitencourt Patricio	Nathalia França Patricio	27/08/1996	M	15/01/2017	2º	A	Banco de Dados
									Sistemas Operacionais
									Redes Neurais

Figura 01. Exemplo de entidade não normalizada

A primeira forma normal exige que se elimine as repetições, para isso você deve dividir a tabela em duas tabelas, como indicado na figura 2. Além disso, todo o atributo da tabela deve ser atômico ou indivisível. Não é permitido atributo multivalorado, composto ou multivalorado composto. Nesta etapa o esquema fica assim:

Aluno(rgm, nome_aluno, nome_pai, nome_mae, nascimento, sexo, data_matricula, semestre, turma, id_semestre)
Disciplina(id_semestre, disciplinas)

A regra da 1º FN: se uma entidade apresentar grupos de repetição, desmembra-lo criando uma nova entidade.

Aluno

nome_aluno	rgm	nome_pai	nome_mae	nascimento	sexo	data_matricula	semestre	turma	id_semestre
Julia Izabel Ribeiro	25100	Luciano Alves Ribeiro	Josiane Ribeiro	25/05/1995	F	10/01/2017	1º	A	1
Amanda Esposito	25101	Edson Franco Esposito	Luciana Antunes Espotito	24/02/1996	F	12/01/2017	2º	A	2
Roberto Rodrigo Prado	25102	Carlos Eduardo Prado	Eliana Pioto Prado	15/01/1994	M	10/01/2017	1º	A	1
Fernando Franchini	25103	Roberto Serafin Franchini	Daniela Andrade Franchini	10/02/1990	F	11/01/2017	1º	A	1
Nerson Martins	25104	Durval Pilon Martins	Clemildes Santos Martins	16/02/1992	M	09/01/2017	1º	A	1
Raphael Lucas Fernandes	25105	Wendel Garcia Fernandes	Andiara Rodrigues Fernandes	30/10/1993	M	18/01/2017	1º	A	1
Adriana Bonfieti Moraes	25106	Ewerton Pereira Moraes	Elena Sagio Moraes	05/05/1994	F	10/01/2017	1º	A	1
Berenice Amaral Martins	25107	Pedro Santos Pereira	Gabriela Aparecida Pereira	01/06/1995	F	09/01/2017	2º	A	2
Alan Jacinto Patricio	25108	Franklin Bitencourt Patricio	Nathalia França Patricio	27/08/1996	M	15/01/2017	2º	A	2

Disciplina

id_semestre	disciplinas
1	Lógica e Técnicas de Programação
1	Arquitetura de Computadores
1	Introdução a Inteligência Artificial
2	Banco de Dados
2	Sistemas Operacionais
2	Redes Neurais

Figura 02. 1º FN sem grupos de repetição

Etapa 2: Na 2º FN deve existir apenas uma chave primaria

Para colocar as entidades na 2º FN você deve garantir que elas estejam na 1º FN e que cada coluna que não seja chave primaria dependa exclusivamente da chave primaria. O objetivo será corrigir a tabela fazendo que os assuntos sejam separados, assim se um atributo depender apenas de uma parte da chave primaria você deve desmembra-lo criando uma nova entidade em que todos os atributos dependam da chave primaria. Nesta etapa o esquema fica assim:

Aluno(rgm, nome_aluno, nome_pai, nome_mae, data_nascimento, sexo)
Matricula(rgm, data_matricula, id_turma, id_semestre)
Disciplina(id_disciplina, ds_disciplinas, id_semestre)
Semestre(id_semestre, ds_semestre)
Turma(id_turma, ds_turma)

Observe por exemplo a figura 02 a tabela Aluno apresenta como chave primária o rgm e as colunas nome_aluno, nome_pai, nome_mae, nascimento e sexo são dependentes da chave primária. As demais colunas não são dependentes e deve ser desmembrada. Quanto a segunda tabela Disciplina não apresenta chave primária então você deve acrescentar a chave primária. A figura 3 apresenta as correções para a 2º FN.

Aluno

rgm	nome_aluno	nome_pai	nome_mae	data_nascimento	sexo
25100	Julia Izabel Ribeiro	Luciano Alves Ribeiro	Josiane Ribeiro	25/05/1995	F
25101	Amanda Esposito	Edson Franco Esposito	Luciana Antunes Espotito	24/02/1996	F
25102	Roberto Rodrigo Prado	Carlos Eduardo Prado	Eliana Pioto Prado	15/01/1994	M
25103	Fernando Franchini	Roberto Serafin Franchini	Daniela Andrade Franchini	10/02/1990	F
25104	Nerson Martins	Durval Pilon Martins	Clemildes Santos Martins	16/02/1992	M
25105	Raphael Lucas Fernandes	Wendel Garcia Fernandes	Andiara Rodrigues Fernandes	30/10/1993	M
25106	Adriana Bonfieti Moraes	Ewerton Pereira Moraes	Elena Sagio Moraes	05/05/1994	F
25107	Berenice Amaral Martins	Durval Pilon Martins	Clemildes Santos Martins	01/06/1995	F
25108	Alan Jacinto Patricio	Franklin Bitencourt Patricio	Nathalia França Patricio	27/08/1996	M

Matricula

rgm	data_matricula	id_turma	id_semestre
25100	10/01/2017	1	1
25101	12/01/2017	1	2
25102	10/01/2017	1	1
25103	11/01/2017	1	1
25104	09/01/2017	1	1
25105	18/01/2017	1	1
25106	10/01/2017	1	1
25107	09/01/2017	1	2
25108	15/01/2017	1	2

Disciplina

id_disciplina	ds_disciplina	id_semestre
1	Lógica e Técnicas de Programação	1
2	Arquitetura de Computadores	1
3	Introdução a Inteligência Artificial	1
4	Banco de Dados	2
5	Sistemas Operacionais	2
6	Redes Neurais	2

Semestre

id_semestre	ds_semestre
1	1º Básico de Programação
2	2º Banco de Dados
3	3º Análise de Sistemas
4	4º Gestão de Projetos
5	5º Conclusão

Turma

id_turma	ds_turma
1	A - Noturno
2	B - Noturno
3	A - Diurno
4	B - Diurno

Figura 03. 2º FN campos dependentes da chave primária

Etapa 3: Na 3º FN identificar atributos que não são dependentes das chaves primárias

Para colocar as entidades na 3º FN você deve garantir que o esteja na 2º FN e que cada atributo seja dependente da chave primaria ou de parte dela. Se algum atributo for dependente de outro campo, estes devem ser desmembrados em uma nova entidade. O objetivo é corrigir a presença de atributos dependentes de outros campos que não sejam chave primária. Além disso se existir campos calculados, esses devem ser eliminados. Nesta etapa o esquema fica assim:

Aluno(rgm, nome_aluno, nome_pai, nome_mae, data_nascimento, sexo)
Matricula(rgm, data_matricula, id_classe)
Classe(id_classe, id_turma, id_semestre)
Disciplina(id_disciplina, ds_disciplinas, id_semestre)
Semestre(id_semestre, ds_semestre)
Turma(id_turma, ds_turma, ano_turma)

No exemplo apresentado na figura 03 a entidade matricula apresenta dos atributos que não dependem da chave primária rgm. A figura 04 apresenta a correção, dividindo a entidade em duas entidades para organizar de forma que todos os atributos dependam das chaves primárias de suas entidades, colocando o esquema na 3º FN.

Aluno

rgm	nome_aluno	nome_pai	nome_mae	data_nascimento	sexo
25100	Julia Izabel Ribeiro	Luciano Alves Ribeiro	Josiane Ribeiro	25/05/1995	F
25101	Amanda Esposito	Edson Franco Esposito	Luciana Antunes Espotito	24/02/1996	F
25102	Roberto Rodrigo Prado	Carlos Eduardo Prado	Eliana Pioto Prado	15/01/1994	M
25103	Fernando Franchini	Roberto Serafin Franchini	Daniela Andrade Franchini	10/02/1990	F
25104	Nerson Martins	Durval Pilon Martins	Clemildes Santos Martins	16/02/1992	M
25105	Raphael Lucas Fernandes	Wendel Garcia Fernandes	Andiara Rodrigues Fernandes	30/10/1993	M
25106	Adriana Bonfieti Moraes	Ewerton Pereira Moraes	Elena Sagio Moraes	05/05/1994	F
25107	Berenice Amaral Martins	Durval Pilon Martins	Clemildes Santos Martins	01/06/1995	F
25108	Alan Jacinto Patricio	Franklin Bitencourt Patricio	Nathalia França Patricio	27/08/1996	M

Matricula

rgm	data_matricula	id_classe
25100	10/01/2017	1
25101	12/01/2017	2
25102	10/01/2017	1
25103	11/01/2017	1
25104	09/01/2017	1
25105	18/01/2017	1
25106	10/01/2017	1
25107	09/01/2017	2
25108	15/01/2017	2

Disciplina

id_disciplina	ds_disciplina	id_semestre
1	Lógica e Técnicas de Programação	1
2	Arquitetura de Computadores	1
3	Introdução a Inteligência Artificial	1
4	Banco de Dados	2
5	Sistemas Operacionais	2
6	Redes Neurais	2

Semestre

id_semestre	ds_semestre
1	1º Básico de Programação
2	2º Banco de Dados
3	3º Análise de Sistemas
4	4º Gestão de Projetos
5	5º Conclusão

Turma

id_turma	ds_turma	ano_turma
1	A - Noturno	2016
2	B - Noturno	2017
3	A - Diurno	2016
4	B - Diurno	2017

Classe

id_classe	id_turma	id_semestre
1	1	1
2	1	2

Figura 04. 3º FN atributos dependendo da chave primária

Demais Formas Normais

Para a maioria das análises de requisitos e projetos de decomposição as 3 etapas iniciais são suficientes para obter um banco de dados rápido e eficiente. Na literatura existe outras formas normais, como a forma normal de Boyce/Cood ou BCNF, a 4º FN e 5º FN. A 4º FN é empregada se após a 3º FN existir redundância, como a utilização de atributos de valores múltiplos para a mesma chave primária. A 5º FN é um caso muito raro de ocorrer, como por exemplo chegar a 4º FN com uma entidade com 3 atributos multivalorados. A forma normal BCNF aperfeiçoa a 3º FN a medida que faz análise das chaves candidatas, impondo que só exista atributos dependentes da chave primária ou da chave candidata. Se existir atributos fora desta especificação deve ser separado em uma nova entidade.

Desnormalização

A implementação ideal de um banco de dados relacional exige que todas as entidades estejam pelo menos na 3º FN. Um bom banco de dados não apresenta redundâncias desnecessárias que podem causar anomalias. Embora normalizar torne o armazenamento mais eficiente, visto que os dados ficam no menor tamanho possível, exige maior quantidade de processamento nas consultas à base de dados. Assim, ao projetar um banco de dados você deve também analisar a necessidade de velocidade de processamento e pensar na questão da performance x demanda de usuários. O problema encontrado com as FN é que ao aplica-las notamos que a quantidade de entidades geradas aumenta e para gerar informações uteis aos usuários será necessário unir os dados que estão espalhados em diversas tabelas. A junção de um grande número de tabelas vai exigir processamento do servidor. O número de operações de entrada e saída aumenta e com isso aumenta o tempo para o servidor responder uma solicitação. A maioria dos SGBDs comerciais é capaz de tratar um grande número de junções, mas em alguns casos a quantidade de acessos simultâneos ao servidor pode gerar um atraso significativo no tempo de resposta. Tenha bom senso ao avaliar até que ponto se deve normalizar um banco, reduzindo o tamanho e a possibilidade de anomalias, ou desnormalizar aumentando o desempenho e podendo apresentar redundância de dados e anomalias em resultados. Um bom projeto de banco de dados não pode ter uma visão única de performance, pois os dados serão utilizados ao longo do tempo e é bem provável que sejam mais perenes que uma aplicação. Tenha em mente que a desnormalização deve ser utilizada com cuidado sempre com uma explicação de necessidade de tal ação.

3. Ferramentas Case

A ferramenta do tipo CASE - *Computer Aided Software Engineering*, ou seja, ferramentas de engenharia de software auxiliada por computador são projetadas para lhe auxiliar no processo de engenharia de software. Existem no mercado diversas ferramentas com essa finalidade, com foco amplo ou especifico para a construção auxiliar o engenheiro de software. Nesta aula

você deve focar apenas as ferramentas que podem auxiliar na área de Banco de Dados. Algumas ferramentas de interesse são DBDesigner, Erwin, Oracle Designer, brModelo e Workbench. É claro que essas são apenas algumas das ferramentas existentes, mas já conseguem transmitir o conceito de ferramenta CASE e permitir explorar mais ferramentas existente.
As principais características que se espera de ferramentas CASE para banco de dados é que elas suportem criar diagramas DER, suportem escrever comandos SQL, *forward engineer*, *reverse engineer* e acompanhamento da documentação. *Forward engineer* é quando à partir de um DER você consegue criar automaticamente o modelo físico no banco de dados. Este recurso permite que os modelos conceitual e lógico sejam criados de forma automática com a possibilidade de edição em cada etapa. *Reverse engineer* é quando à partir de um banco de dados físico, já implementado, a ferramenta consegue criar para você o diagrama DER automaticamente. Normalmente este recurso é utilizado quando um banco de dados não apresenta em sua documentação o diagrama quer por ser parte de software legado ou de um processo sem controle. A documentação deve apresentar suporte para o dicionário de dados e permitir acompanhar a evolução do banco de dados. As vantagens das ferramentas CASE são: maior velocidade no desenvolvimento de projetos, melhor qualidade dos processos, melhor documentação visto que integra DER, banco de dados físico e documentação e uso de interface gráfica. Com exceção do CA Erwin todos as demais ferramentas indicadas na tabela 01 são gratuitas. A tabela 1 apresenta os sites onde você pode encontrar as ferramentas para baixar. Vale lembrar que Workbench é compatível com o MySQL e não cria facilidade para ser utilizado com o Oracle. O DBDesigner permite a compatibilidade com o Oracle, enquanto Erwin, brModelo e Oracle Developer DataModeler são compatíveis com o SGBD Oracle.

Tabela 01. Ferramentas Case e endereços para download

Nome	Site
Erwin	http://erwin.com/products/data-modeler/
DBDesigner	https://dbdesigner.br.uptodown.com/windows
brModelo	http://sis4.com/brModelo/brModelo/download.html
Oracle Developer DataModeler	http://www.oracle.com/technetwork/developer-tools/datamodeler/downloads/datamodeler-087275.html
Workbench	https://dev.mysql.com/downloads/workbench/

4. Padronização no SGBD

Não existe um consenso sobre como se deve criar o projeto físico do SGBD nem sobre quais regras de nomenclatura devem ser adotadas para tabelas e colunas. A padronização de nomes, campos, tabelas é importante pois facilitam o entendimento e a velocidade de desenvolvimento de analistas e programadores. Quando você está modelando e trabalhando com um banco de dados pequeno, pode parecer perda de tempo discutir e definir uma política de padronização, mas quando o número de tabelas é grande as coisas mudam e pode se tornar muito difícil extrair informações de um banco não padronizado. Existe dois padrões que inspiram muitos DBA a criar suas regras que são variações para cada SGBD. A trigramação e o qualificador nome descritivo.
A trigramação é uma cadeia de caracteres, normalmente constituída por três letras da entidade ou pela escolha das três letras mais significativas de uma entidade. Essas três letras são utilizadas como parte da definição do atributo da mesma entidade. Para exemplificar a figura 05 apresenta a entidade Cliente e ela convertida para trigramação.

Figura 05. Trigramação

Como regra geral da trigramação nunca utilize artigo como ‘de’, ‘das’ ou ‘com’ ou pronomes em uma entidade ou atributo. Não utilize caracteres especiais, acentos ou cedilha. Todas as palavras compostas devem ser separadas por ‘_’ underscore como pessoa_juridica ou

pessoa_fisica. O nome da entidade deve ser sempre no singular e deve expressar de forma clara a finalidade da entidade. Na trigramação ao criar abreviaturas pode ocorrer de ao escolher as três primeiras letras de uma entidade encontrar outra entidade que apresente as três letras iniciais idênticas, neste caso você deve manter duas letras idênticas e escolher uma terceira que permita diferencias as entidades. Se o modelo for complexo e tiver a necessidade de se criar módulos o nome do módulo será abreviado e atribuído a todas as tabelas que lhe pertence. Toda a chave primária terá a trigramação seguido da palavra id.

Qualificador nome descritivo

O qualificador nome descritivo tem como base o padrão ISO/IEC 11179-5. A regra utiliza basicamente algumas tabelas com símbolos e todas as entidades e atributos do DER devem de alguma forma ser representados por um destes símbolos. Por exemplo imagine que queremos converter a entidade Cliente do DER para o padrão qualificador nome descritivo. Você deverá percorrer para cada atributo da entidade a tabela 02 e escolher o símbolo mais adequado. A figura 06 apresenta como fica a representação da entidade no modelo.

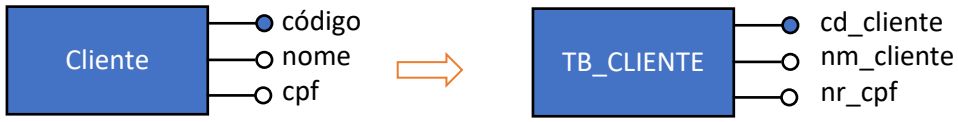


Figura 06. Qualificador nome descritivo

A tabela 02 apresenta todos os qualificadores definidos com seus respectivos significados. Deve converter a representação de um atributo do DER para um dos qualificadores apresentado na tabela 02 para cada atributo de uma entidade.

Tabela 02. Qualificador para Atributos

Qualificador	Significado
`cd_+[nome da coluna]`	Código
`no/nm_+[nome da coluna]`	Nome
`nr_+[nome da coluna]`	Número
`vl_+[nome da coluna]`	Valor
`qt_+[nome da coluna]`	Quantidade
`tx_+[nome da coluna]`	Taxa ou percentual
`ds_+[nome da coluna]`	Descrição
`sg_+[nome da coluna]`	Sigla
`dt_+[nome da coluna]`	Data
`hr_+[nome da coluna]`	Hora
`id/ie_+[nome da coluna]`	Identificador
`im_+[nome da coluna]`	Imagem
`st_+[nome da coluna]`	Situação ou status
`cg_+[nome da coluna]`	Coordenadas Geográficas
`au_+[nome da coluna]`	Auditoria

Quanto ao nome da própria entidade, você deve achar um qualificador mais adequado utilizando a tabela 03 que apresenta os qualificadores que podem ser utilizados e seus significados.

Você sabia
O DATASUS definiu uma metodologia para a administração de banco de dados que tem como base alguns estudos feitos sobre a norma ISO/IEC 11179-5. Você pode ler a norma criada e utilizar o resumo para adequar a norma para o seu banco de dados. Acesse:
http://datasus.saude.gov.br/estrutura-mad/norma-mad-menu e
http://datasus.saude.gov.br/images/MAD/files/DAAED_MAD_GuiaModelagemDados-221.pdf

---

Tabela 03. Qualificador para Entidades

Qualificador	Significado
tb_+[nome da tabela]	Tabela do software
rl_+[nm_tb_pai]_[nm tb_filho]	Tabela de Relacionamento
au_+[nome da tabela]	Tabela de Auditoria
tm_+[nome da tabela]	Tabela Temporária
th_+[nome da tabela]	Tabela de Histórico
ta_+[nome da tabela]	Tabela Auxiliar
bk_+[nome da tabela]	Tabela de Backup
rt_+[nome da tabela]	Tabela Relacionamento Ternário

É importante lembrar que como não existe um consenso, você está livre para criar seu padrão para o banco de dados. Ele deve permitir que o banco cresça e suporte ter milhares de tabelas, sendo fácil de usar, evitando ambiguidades, permitindo acessar o dado de maneira correta.

5. Introdução à Linguagem SQL

SQL é a linguagem de interface para os SGBDs relacionais, assim, todos os usuários que querem interagir com o banco de dados deve fornecer comandos escritos nela. A linguagem SQL foi desenvolvida no início da década de 70 pela IBM Research no laboratório em San Jose com o nome SEQUEL, tendo seu nome modificado mais tarde. SQL significa Structured Query Language ou linguagem estruturada de consulta. O SQL permite a criação de banco de dados, executar tarefas de gerenciamento, consultar, transformar dados, executar funções e gatilhos. Além disso deve ser padronizado para que o usuário não tenha que reaprender o básico quando passar de um SGBD para outro, assim, em 1987 o American National Standard Institute ANSI publicou o padrão SQL. A linguagem foi concebida para ser usada segundo o modelo de Codd e com o tempo ganhou outras funcionalidades.
O nascimento do SQL se deu devido a necessidade de trabalhar de forma fácil e flexível com junções, produto cartesiano, diferença, intersecção, projeção e união de dados. Praticamente todos os fornecedores de SGBD relacionais oferecem o SQL como interface. Suas funções se enquadram em duas categorias: Linguagem de definição de dados (DDL) e Linguagem de manipulação de dados (DML).
A DDL inclui comandos para criar objetos de banco de dados como tabelas, índices e visualizações, bem como comandos para definir direitos de acesso. Alguns comandos de definição de dados que veremos nesta aula são apresentados na tabela 4.

Tabela 04. Alguns comandos DDL

Comando	Descrição
CREATE SCHEMA AUTHORIZATION	Cria um esquema de banco de dados
CREATE TABLE	Cria uma nova tabela
NOT NULL	A coluna não poderá ter valores nulos
UNIQUE	O valor será único na coluna
PRIMARY KEY	Define a coluna como chave primária
FOREING KEY	Define a coluna como chave estrangeira
DEFAULT	Define um valor padrão para a coluna
CHECK	Valida os dados de um atributo
CREATE INDEX	Cria um índice para uma tabela
CREATE VIEW	Cria um subconjunto dinâmico linhas e colunas a partir de uma ou mais tabelas
ALTER TABLE	Modifica a definição de uma tabela
CREATE TABLE AS	Cria um tabela com base em uma consulta
DROP TABLE	Exclui uma tabela e seus dados
DROP INDEX	Exclui um índice
DROP VIEW	Exclui uma visualização

Você vai gostar de saber que aprender o SQL é relativamente fácil. Você terá de aprender aproximadamente 100 instruções. É um vocabulário simples e que por ser padrão ANSI/ISO significa aprendendo SQL você conseguirá migrar para diferentes SGBDRs. Mas cada

fabricante adicionou recursos que dificultam a migração de uma aplicação que foi desenvolvida em um SGBDR para outro. Assim, você notará que aprender SQL é igual a falar um idioma, dependendo da região, surgi expressões e dialetos que podem dificultar a comunicação em alguns momentos. Mas, para este caso, uma boa consulta no manual ou na internet pode sanar rapidamente as dificuldades.
Antes de iniciar a utilização do SQL, você deve conhecer a ferramenta SQL Developer que foi instalada na aula anterior. A figura 07 apresenta a interface do SQL Developer.

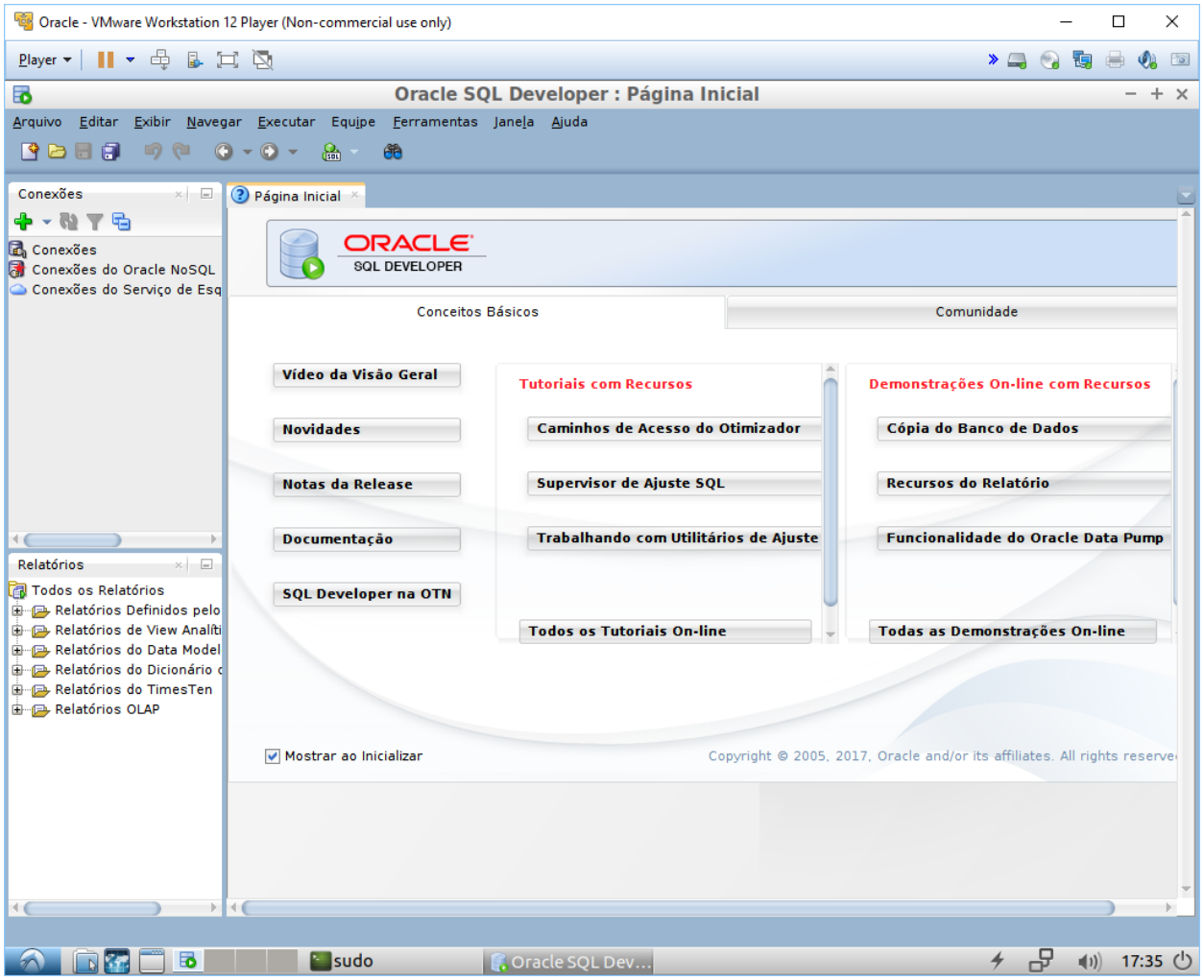


Figura 07. Tela do SQL Developer

Selecione Arquivo, Novo, Camada do Banco de Dados e Botão Ok ou se preferir pode utilizar o atalho <Ctrl><N> pressionando as teclas simultaneamente. Ainda existe a possibilidade de executar o mesmo comando clicando sobre o + verde que aparece na interface gráfica na janela conexões. Se optar por esta ação selecione Nova Conexão e uma janela semelhante a apresentada na figura 08 deverá aparecer.
Você deve criar uma conexão para o DBA do SGBD e uma segunda conexão para utilizar nesta aula. Observe que a conexão DBA não deve ser utilizada como sua conexão particular, pois você deve deixar este super usuário para criar outros usuários, definir políticas de acesso, tuning de banco de dados e outras atividades administrativas do banco. Por isso, nesta aula você irá criar duas conexões a DBA e Aluno.
Para criar o usuário DBA no campo nome digite “DBA – Oracle”, no campo nome do usuário digite “system” e em senha digite “manager” conforme definimos no processo de instalação. Selecione o campo “Salvar Senha” e selecione o botão Testar. Se aparecer no Status: “Sucesso” selecione o botão Conectar.

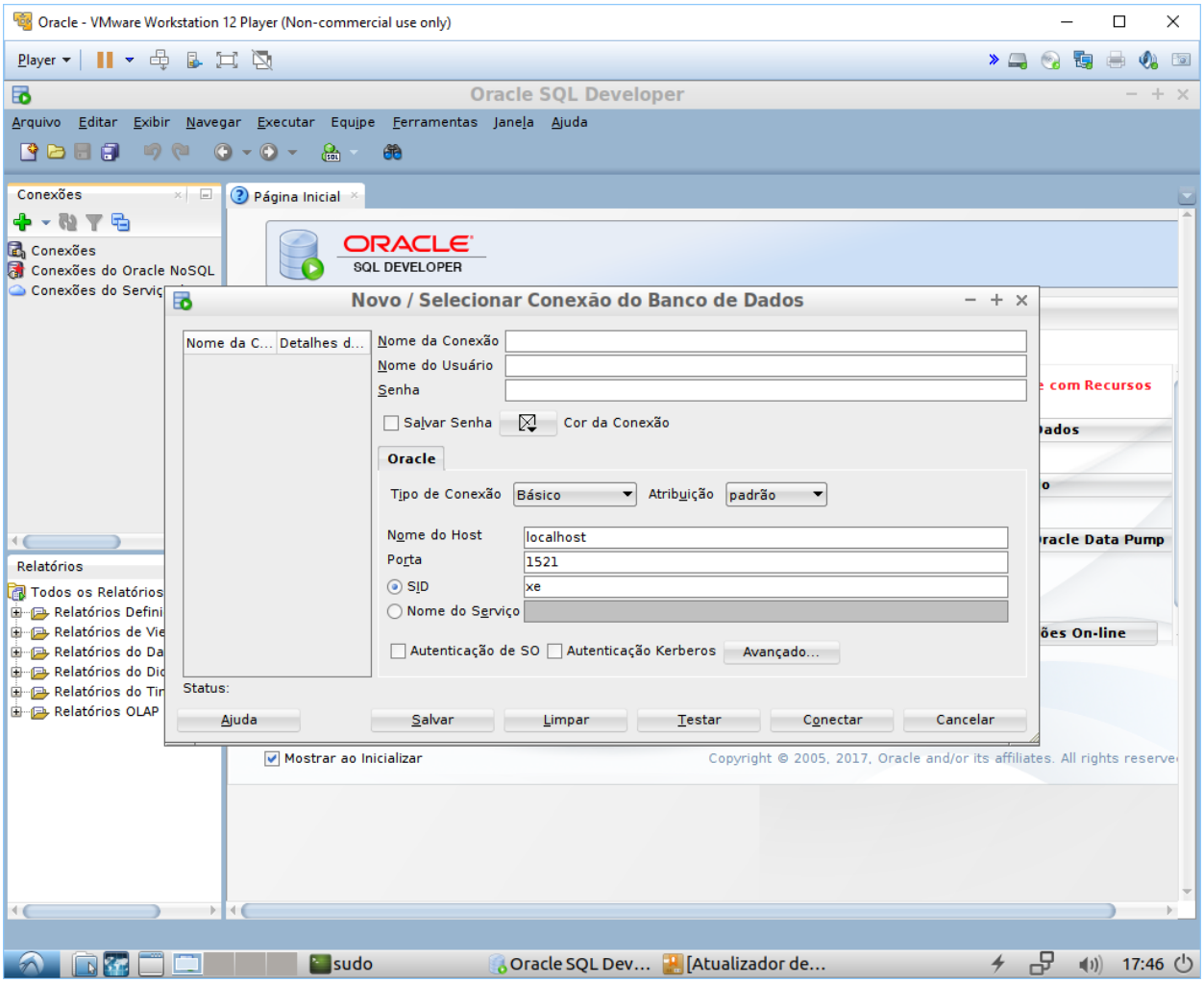


Figura 08. Nova Conexão do SQL Developer com o SGBD Oracle

Observe que a nova conexão foi criada. Selecione o + que aparece antes da Conexão DBA – Oracle e a interface deve ficar semelhante a indicada na figura 09.

Observe que na janela a direita uma nova aba foi criada com o nome da conexão e um espaço para enviarmos comandos SQL para o SGBD. Para criar um usuário no Oracle você deve utilizar a instrução CREATE USER. De maneira simplificada o comando tem a seguinte sintaxe:

```
CREATE USER nome_do_usuario IDENTIFIELD BY senha_usuario
DEFAULT TABLESPACE nome_tablespace
TEMPORARY TABLESPACE tablespace_temporaria;
```

Onde

nome_do_usuario: é o nome do usuário que você quer criar
senha_usuario: é a senha que este usuário terá
nome_tablespace: onde os objetos que do usuário serão armazenados
tablespace_temporaria: onde são armazenados os dados temporários do usuário, como por exemplo uma tabela temporária.

Então digite o seguinte comando:

```
CREATE USER aluno IDENTIFIELD BY aluno
DEFAULT TABLESPACE users
TEMPORARY TABLESPACE temp;
```

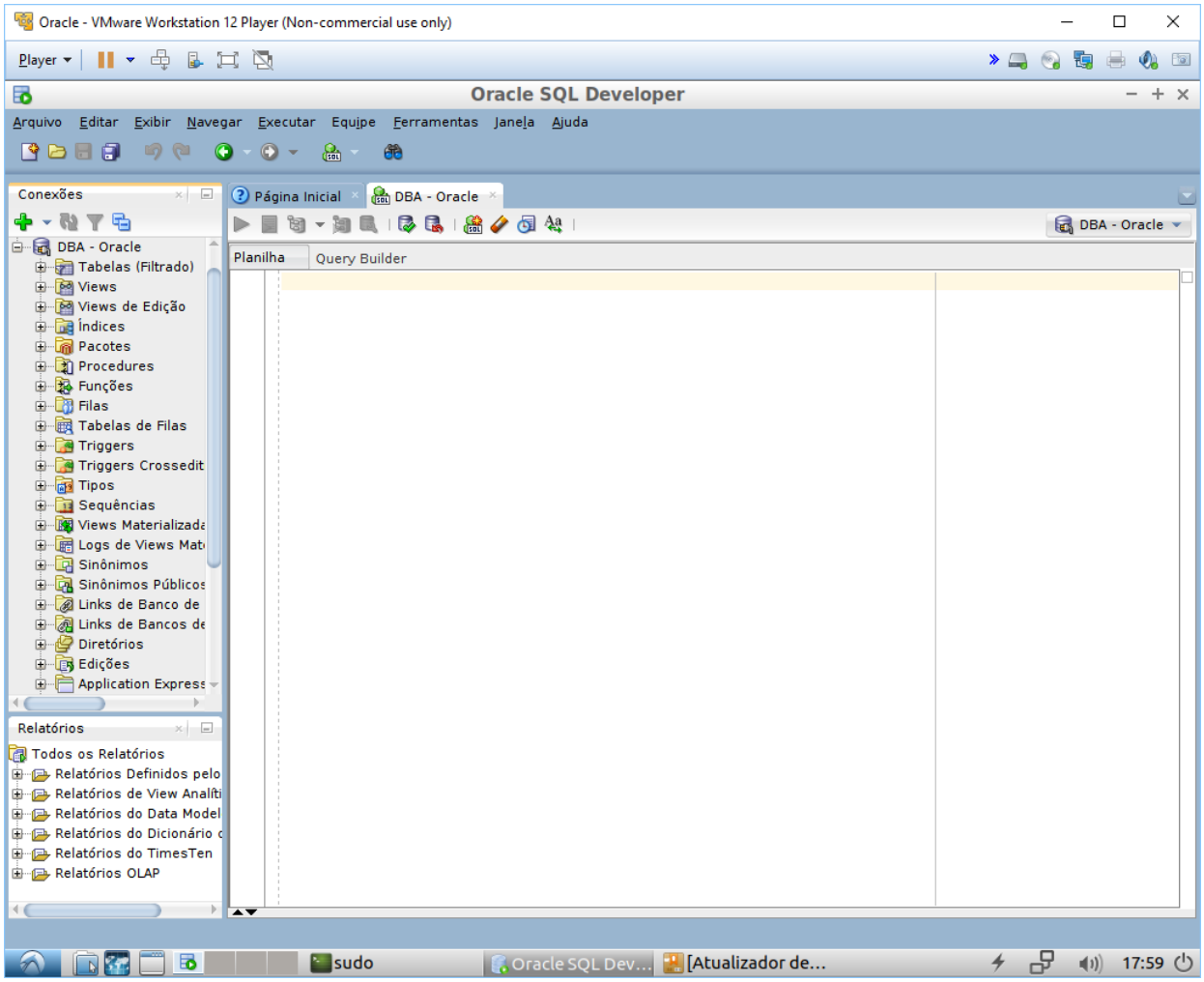


Figura 09. Conexão DBA – Oracle criada

Para executar o comando você pode pressionar as teclas <Ctrl><Enter> estando o cursor sobre a linha de comando ou selecionar o ícone play abaixo do título da aba. Veja a figura 10 o ícone de execução dos comandos SQL.


Figura 10. Ícone que executa uma linha de comando SQL

Estas duas opções executam apenas o comando em que o cursor se encontra. Para executar todos os comandos digitados na interface, ou seja, um script você pode pressionar a tecla <F5> ou selecionar o ícone apresentado na figura 11.


Figura 11. Ícone que executa todas as linhas do script

Neste momento como só existe um comando escrito em três linhas, você pode optar por qualquer um dos métodos, desde que se limite a apenas uma execução. Você notará que a janela principal será dívida e a Saída do Script se tornará visível com a mensagem "`User ALUNO criado`".
Após obter sucesso na primeira execução, se você tentar novamente, irá ser notificado de um erro. Isso se dá porque o SGBD teve sucesso na primeira execução criando o usuário e ao tentar executar o comando pela segunda vez o SGBD nota que o usuário já existe, sendo impossível criar um novo usuário com o mesmo nome, pois esta ação criaria redundância. Assim o SGBD lhe informa que existe um conflito no nome ou um erro de atribuição. Para dar privilegio para o usuário aluno de criar uma sessão de conexão com o banco de dados, criar tabelas, criar visão de dados digite:

```
GRANT create session, create table, create view, dba TO aluno;
```

Execute somente esta linha de comando com <Ctrl><Enter> e observe se na Saída do Script apareceu a mensagem "`Grant bem-sucedido`". Se você conseguiu executar ambos os comandos com sucesso você pode criar uma nova conexão para o usuário aluno. Para criar a conexão do usuário Aluno selecione Arquivo, Novo, Camada do Banco de Dados e Botão Ok ou se preferir pode utilizar o atalho <Ctrl><N> pressionando as teclas simultaneamente. É possível executar o mesmo comando clicando sobre o + verde que aparece na interface gráfica na janela conexões. Selecione Nova Conexão e uma janela semelhante a apresentada na figura 08 deverá aparecer novamente. No campo nome digite "Aluno", no campo nome do usuário digite "aluno" e em senha digite "aluno". Selecione o campo "Salvar Senha" e selecione o botão Testar. Se aparecer no Status: "Sucesso" selecione o botão Conectar. Observe que a nova conexão foi criada, selecione o + que aparece antes da Conexão Aluno e a interface deve ficar semelhante a indicada na figura 12.

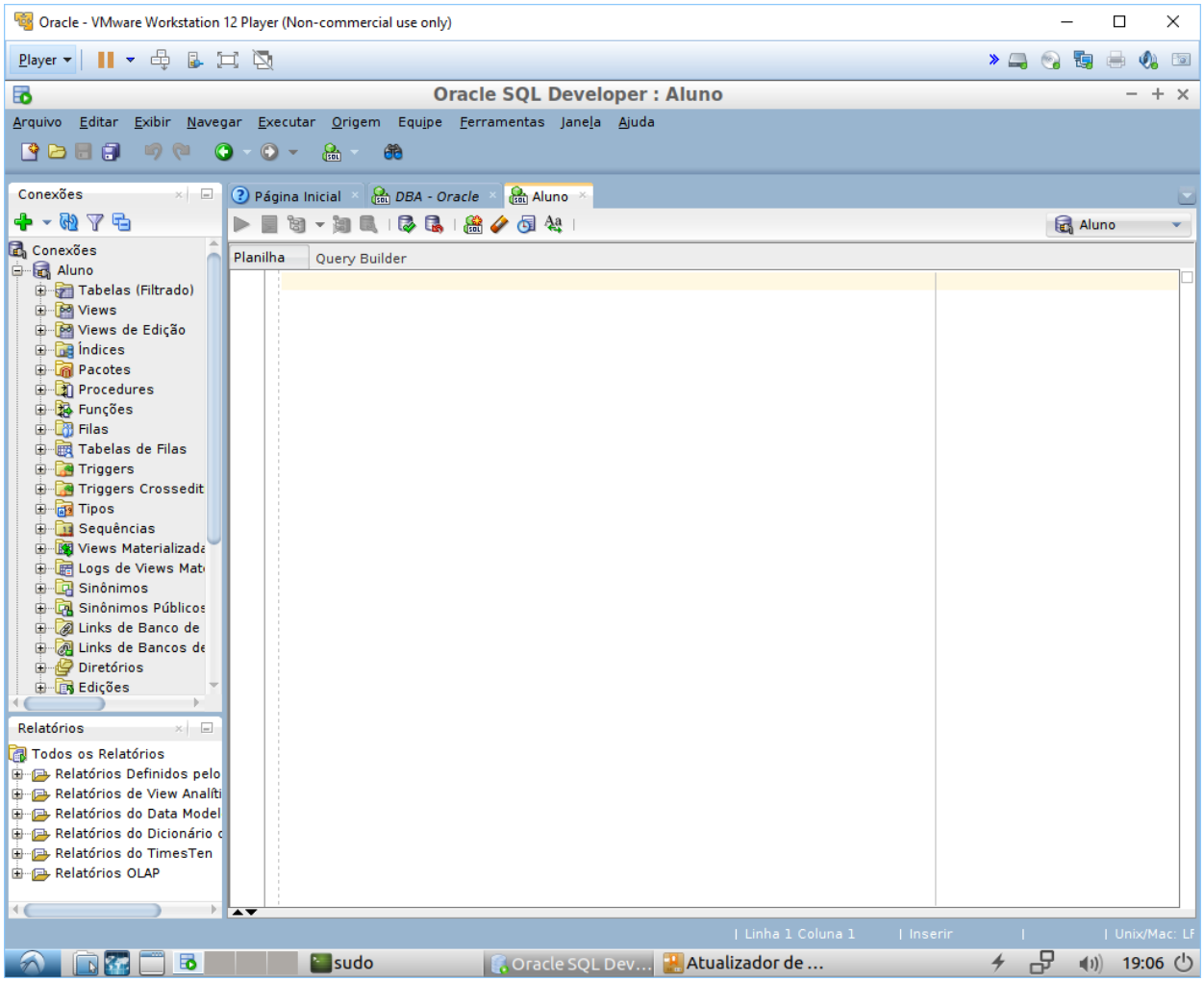


Figura 12. Conexão Aluno criada

Esquema de Banco de Dados

No SQL esquema é um grupo de objetos de banco de dados, que podem ser tabelas e índices, relacionados entre si. Na maioria das vezes o esquema pode ser atribuído a uma aplicação ou usuário. O SGBD pode manter vários esquemas que pertencem a grupos de usuários ou aplicações. Os esquemas são uteis pois agrupam as tabelas por proprietários e aplicam o primeiro nível de segurança, ou seja, o usuário só pode ver as tabelas que lhe pertence. Para comando que cria um esquema é:

```
CREATE SCHEMA AUTHORIZATION nome_usuario;
```

Então você deve digitar na nova conexão:

```
CREATE SCHEMA AUTHORIZATION aluno;
```

A maioria dos SGBDRs empresariais dá suporte a esse comando, mas eles atribuem o comando automaticamente quando você cria o usuário pela interface gráfica. Neste momento você está criando manualmente e vale ressaltar que este comando só funciona se você estiver conectado com o usuário que terá o esquema criado, assim, não tente criar um esquema para o usuário DBA e nem faze-lo definir o esquema do usuário aluno. Execute o comando como apresentado na figura 13.

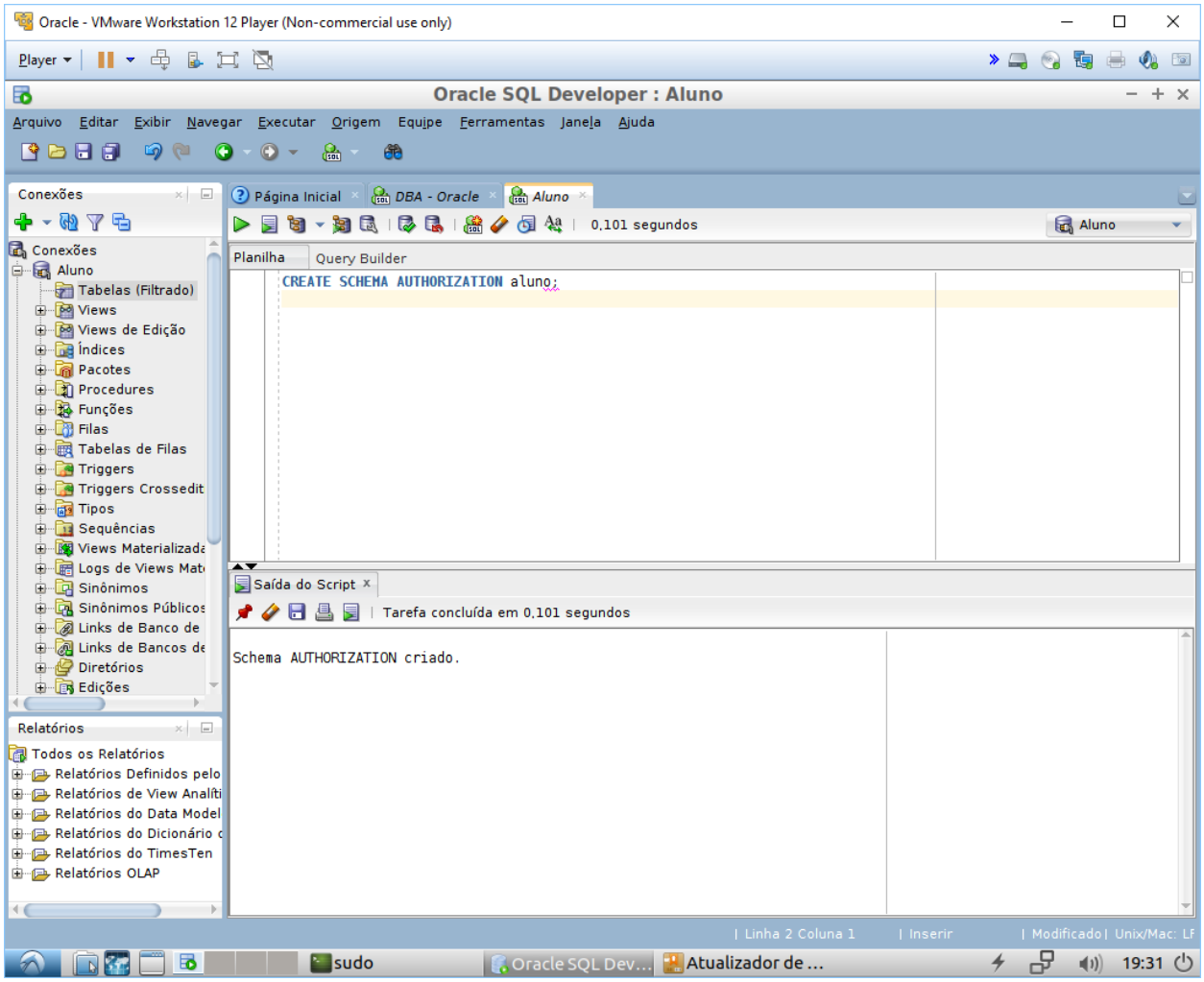


Figura 13. Criando esquema no usuário aluno

Após a criação do esquema de banco de dados, você poderá definir as estruturas das tabelas que você deseja criar. Para tanto, veja os tipos de dados que você pode utilizar na construção ou parametrização de dados no Oracle.

VARCHAR2 OU VARCHAR

São dados de caracteres com comprimento variável, podendo atingir o tamanho máximo de 4.000 bytes em uma tabela e 32.767 bytes no PL/SQL. O varchar2 é variável pois somente utiliza espaço ocupado não fazendo reserva de alocação. É possível utilizar varchar ou varchar2 no SQL. A Oracle manteve o tipo varchar para ser compatível com outros SGBDs, contudo está opção não é recomendada visto que ela pretende utilizar tal sintaxe em suas novas versões à medida que o SQL evolui.

CHAR

São dados de caracteres com comprimento fixo de até 255 caracteres ou 2.000 bytes. Ao contrário do varchar2 o char sempre contém o tamanho máximo atribuído a ele, independentemente do dado atribuído, sua alocação sempre será os dados mais os espaços

em branco. Seu uso só é vantajoso quando se terá uma cadeia de caracteres obrigatório como por exemplo as 3 letras da placa de um carro ou o sexo de um usuário.

NUMBER

São dados numéricos com sinal e pode apresentar ponto decimal sendo NUMBER(n,d) onde n é o número total de dígitos e d a quantidade de números do total que irão compor o decimal. Assim o n deve ser sempre maior que o d, pois o n é o total de números que compõem o tipo e d defini quantos números pertencerão a parte decimal do número. Por exemplo a declaração NUMBER(6,2) indica que é possível armazenar 123,56 pois temos o total de 5 números e o símbolo decimal. Também é possível armazenas -43,15 observe que o sinal negativo será computado no tamanho máximo atribuído. O tipo NUMBER apresenta alguns subtipos: DECIMAL, DEC, DOUBLE PRECISION, NUMERIC, INTEGER, INT, SMALLINT, FLOAT. Como varchar os subtipos criados visam tornar compatível e facilitar que usuários de outros SGBD migrem para o Oracle, assim a maioria dos casos esses tipos são alias ou apelidos para o tipo NUMBER. Assim se você não está migrando o SGBD recomenda-se utilizar o NUMBER para os dados numéricos.

DATA

São dados no formato de data e hora. O nome deveria ser DATATIME pois a hora está sempre presente independentemente de você utiliza-la ou não. Se você não for utilizar o campo hora o sistema atribui 00:00 A.M. automaticamente. Permite data entre 1 de janeiro de 4712 AC até 31 de dezembro de 9999 DC. O calendário interno tem em conta as alterações de calendário como a passagem do calendário Juliano para Gregoriano em 1582, onde foram eliminados 10 dias.

TIMESTAMP

São dados no formato de data, hora com segundos fracionados. É similar ao tipo data mas acrescido dos segundos fracionados. Existem muitas variações deste tipo de dados como WITH TIMEZONE e WITH LOCAL TIMEZONE.

BOOLEAN

São dados que armazenam 1 para os valores true ou 0 para valores false.

LONG

São dados de caracteres de comprimento variável até 2 gigabytes. Fornecido para compatibilidade com versões anteriores. No PL/SQL pode atingir o tamanho máximo de 32.760 bytes. Observe que isso é menor do que a largura máxima de uma coluna LONG em uma tabela.

RAW

São dados brutos que podem ser armazenados em uma coluna. O comprimento máximo é de 2.000 bytes.

LONG RAW

São dados brutos parecido com o tipo RAW que podem ser armazenados até o tamanho máximo de 2 GB. Existe algumas restrições para LONG RAW e LONG, como uma função não poder retornar o tipo LONG, não poder ser indexado, não poder ser clausula de restrição como WHERE.
Por apresentar limitações a Oracle criou o tipo LOB para resolver as limitações impostas pelo tipo LONG.

CLOB

São dados do tipo caractere com acentuações no tamanho de 1 até o tamanho máximo de 4 GB. O número de colunas CLOB por tabela é limitado somente pelo número máximo de colunas por tabela.

NCLOB

Similar a CLOB com suporte a Unicode. Conjuntos de caracteres de largura fixa e largura variável são suportados. Armazena dados de conjunto de caracteres nacionais.

BLOB

Armazenam dados não estruturados como: som, imagem e dados binários. O tamanho máximo é de 4 GB, podendo atingir 128 GB de armazenamento. Os objetos podem ser pensados como bitstreams sem semântica. Os objetos possuem suporte transacional completo. Você pode fazer alterações através do SQL com os pacotes DBMS_LOB ou Oracle Call Interface (OCI). As manipulações de valor podem ser feitas e revertidas. No entanto, você não pode salvar uma localização em uma sessão e depois usá-la em outra sessão.

BFILE

Permite o acesso de arquivos binários que são armazenados no sistema de arquivos fora do banco de dados. Um atributo BFILE armazena o local do arquivo binário no sistema do servidor. O localizador mantém o nome do diretório e o nome do arquivo. Você pode alterar o nome do arquivo e o caminho sem afetar a tabela. Os arquivos binários LOB não participam de transações e não são recuperáveis. Em vez disso, o sistema operacional fornece integridade e o armazenamento do arquivo. O DBA deve garantir que o arquivo externo existe e que os processos Oracle possuem permissões de leitura no sistema operacional no arquivo. O tipo de dados permite o suporte de somente leitura nos arquivos binários grandes. Você não pode modificar ou replicar esse arquivo. Oracle fornece APIs para acessar os dados dos arquivos. As interfaces primárias que você usa para acessar os dados do arquivo são os pacotes DBMS_LOB e Oracle Call Interface (OCI).

Resumo dos tipos de dados

Os tipos de dados mais comuns que você vai utilizar no Oracle é apresentado na tabela 05 sendo que estes devem nortear o processo de definição dos atributos ao criar as tabelas.

Tabela 05. Tipos de Dados Padrão

Tipo de Dados	Descrição
`VARCHAR2(n)`	Campo do tipo caractere com tamanho variável
`CHAR(n)`	Campo caractere fixo com tamanho máximo de 255 caracteres
`DATE`	Campo de data e hora dia, mês, ano, hora, minuto e segundo
`NUMBER(n,d)`	Campo numérico onde n é total de dígitos e d as casas decimais
`BLOG`	Campo para som, imagem e dados binários

Leitura Indicada
Para mais informações sobre os tipos de dados que você pode utilizar nos atributos consulte a matéria intitulada Referencias da Linguagem SQL para Banco de Dados disponível em: http://docs.oracle.com/cd/B28359_01/server.111/b28286/sql_elements001.htm#SQLRF0021 e Tipos de Dados do Oracle 8 a Oracle 11g disponível em: https://ss64.com/ora/syntax-datatypes.html

---

Quando se define o tipo de dados do atributo, você deve pensar em sua utilização. Será que este dado será utilizado em algum processamento ou apenas a nível de informação? Vai sofrer cálculos? Para ilustrar imagine uma aplicação imobiliária, um atributo será o número de banheiros de uma casa. Este atributo pode ser number(2) ou varchar2(2). Se optar por maior velocidade e menor espaço de armazenamento você vai atribuir varchar2, pois não ocorrem operações aritméticas sobre o número de banheiros. Se quiser classificar as casas pelo

número de banheiros você vai optar por number para efetuar comparações e criar índices. Assim vale refletir sobre a utilização do atributo para definir o tipo de maneira mais adequada. Os SGBDRs comerciais dão suporte a muito mais tipos de dados dos que apresentados na tabela 05 no entanto esta aula foi concebida para apresentar os tipos básicos do SQL.

Criando Tabelas

Agora você está pronto para criar a primeira tabela no SGBD com a ajuda do comando CREATE TABLE apresentado a seguir.

```
CREATE TABLE nome_da_tabela (
    coluna_1     tipo_de_dados  [restrição] [,
    coluna_2     tipo_de_dados  [restrição] [,
    PRIMARY KEY  (coluna_1      [, coluna_2]) ] [,
    FOREIGN KEY  (coluna_1      [, coluna_2]) REFERENCES
                 nome_da_tabela] [,
    CONSTRAINT nome_restrição ]
);
```

Para facilitar a leitura do SQL, a maioria dos DBA e programadores utilizam uma linha para definir uma coluna e tabulação para alinhar características e restrições dos atributos. Outra característica é que você pode padronizar seu código mantendo por convenção o nome das tabelas em maiúsculo e atributos em minúsculo.
Para exemplificar imagine que você deve criar as tabelas representadas no modelo DER apresentado na figura 14.

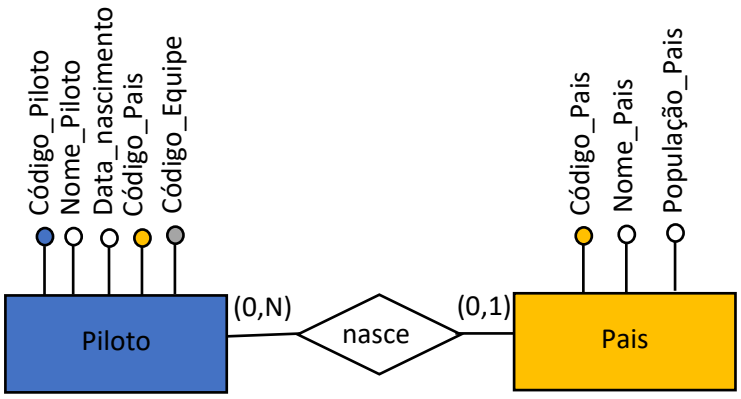


Figura 14. Modelo Conceitual das entidades Piloto e País.

A primeira tarefa é converter o Modelo Conceitual em Modelo Lógico. Nesta etapa você pode aproveitar e remover acentuação, cedilha e adotar um dos padrões estudados na sessão 4 desta aula. O modelo lógico no formato texto ficaria assim:

```
TB_PILOTO(id_piloto, nm_piloto, dt_nascimento, id_pais, id_equipe)
TB_PAIS(id_pais, nm_pais, nr_poulacao)
```

O modelo gráfico fica similar ao apresentado na figura 15.

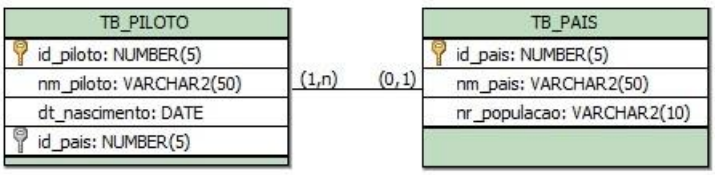


Figura 15. Modelo Lógico padronizado com qualificado nome descritivo

O código SQL para criar a tabela Piloto fica assim:

```
CREATE TABLE TB_PILOTO (
id_piloto       NUMBER(5)       PRIMARY KEY,
nm_piloto       VARCHAR2(50),
dt_nascimento   DATE,
id_pais         NUMBER(5)
);
```

Para executar a instrução você deve apertar <Ctrl><Enter> simultaneamente ou utilizar o ícone play. A instrução cria a tabela TB_PILOTO com id_piloto como chave primaria, a coluna nm_piloto com 50 espaços, a coluna dt_nascimento para armazenar uma data válida e uma coluna que será definida como chave estrangeira mais adiante. Para criar a tabela TB_PAIS você pode utilizar um comando similar ao anterior. Você notará que o SQL permite ao desenvolvedor variações de métodos, por isso a seguir será apresentado uma outra forma de indicar a chave primária. Para criar novamente a tabela é necessário antes apagar a existente. Utilize:

```
DROP TABLE TB_PILOTO;
```

Após executar o comando DROP TABLE você poderá executar novamente o comando CREATE TABLE só que neste caso estamos especificando uma mensagem caso ocorra a violação da chave primaria.

```
CREATE TABLE TB_PILOTO (
id_piloto     NUMBER(5) CONSTRAINT tb_piloto_id_piloto_pk PRIMARY KEY,
nm_piloto     VARCHAR2(50),
dt_nascimento DATE,
id_pais       NUMBER(5)
);
```

Restrições ou CONSTRAINT são regras básicas estabelecidas para o preenchimento de uma ou mais colunas da tabela. Existe uma padronização para as CONSTRAINT. Observe que após o comando CONSTRAINT é atribuída uma mensagem que será utilizada pelo SGBD para dar um feedback amigável para o usuário, caso ocorra uma tentativa violar a regra de restrição. No exemplo visto você está definindo uma regra de restrição de chave primária. O padrão que utilizado foi declarar: nome_da_tabela + nome_do_campo + sigla. A sigla utilizada foi pk que é uma abreviação de PRIMARY KEY ou chave primaria. Você tem toda a liberdade de criar mensagens otimizadas, desde que não ultrapassem 30 caracteres. As restrições e são definidas ao final da especificação de cada coluna ou ao final do comando, antes do ponto-e-vírgula. Entre as restrições mais comuns encontram-se: chaves primárias (pk), chaves únicas (un), chaves estrangeiras (fk), identificadores de campos obrigatórios (nn), condições para valores permitidos para determinado campo (ck).
Você pode verificar se a tabela foi criada com as especificações que solicitou. Para verificar utilize o comando DESCRIBE ou DESC que vai mostrar ou exibir as estruturas da tabela ou visão que você verá nas próximas aulas. Digite:

```
DESCRIBE TB_PILOTO;
```

Você pode apagar novamente a tabela, para criar novamente de outra forma. Nesta aula você deve testar as diferentes formas que o SQL permite você criar tabelas, isso vai lhe ajudar a definir o formato que mais lhe agrada bem como facilitar quando for contratado a se ajustar rapidamente a política adotada pela empresa. É comum a empresa optar por um padrão e solicitar que todos que utilizam o SGBD codifiquem de maneira uniforme. Desta forma DROP a tabela TB_PILOTO e crie novamente com o seguinte comando:

```
DROP TABLE TB_PILOTO;

CREATE TABLE TB_PILOTO (
id_piloto     NUMBER(5),
nm_piloto     VARCHAR2(50),
dt_nascimento DATE,
```

```
id_pais       NUMBER(5),
CONSTRAINT tb_piloto_id_piloto_pk PRIMARY KEY (id_piloto)
);
```

A forma apresentada por último é considera por muitos DBAs como a mais organizada, pois define as restrições no final do comando. Como você pode perceber espera-se mais de uma restrição em cada tabela e se você quiser tornar o SGBD mais amigável deverá definir cada restrição com CONSTRAINT. Se você não definir uma mensagem de erro o SGBD deve criar uma automática como SYS_C007082 onde 007082 identifica a restrição. Caso uma violação ocorra o SGBD irá informar apenas a violação. Assim DROP novamente a tabela e crie com o seguinte comando:

```
DROP TABLE TB_PILOTO;

CREATE TABLE TB_PAIS (
id_pais          NUMBER(5),
nm_pais          VARCHAR2(50),
nr_populacao     VARCHAR2(10),
CONSTRAINT tb_pais_id_pais_pk PRIMARY KEY (id_pais),
CONSTRAINT tb_pais_nm_pais_nn CHECK (nm_pais IS NOT NULL)
);

CREATE TABLE TB_PILOTO (
id_piloto     NUMBER(5),
nm_piloto     VARCHAR2(50),
dt_nascimento DATE,
id_pais       NUMBER(5),
CONSTRAINT tb_piloto_id_piloto_pk PRIMARY KEY (id_piloto),
CONSTRAINT tb_piloto_id_pais_fk   FOREIGN KEY (id_pais)
          REFERENCES TB_PAIS (id_pais)
);
```

Observe que a tabela TB_PILOTO apresenta o atributo id_pais que chave estrangeira. Para ser chave estrangeira em uma tabela o atributo deve ser chave primária em outra tabela e o relacionamento entre as entidades deve ser 1:N. Neste caso, o atributo id_pais é chave primária na TB_PAIS e pode ser declarado a CONTRAINT. Observe que tb_piloto_id_pais_fk é o nome da restrição seguido da declaração de qual campo é chave estrangeira pelo comando FOREIGN KEY (id_pais). A declaração REFERENCES indica qual tabela será consultada para validar a chave estrangeira, no exemplo TB_PAIS e dentro dos parênteses indicamos qual a coluna será consultada. Lembre-se que deve ser a chave primária da tabela, ou seja, a chave primária de TB_PAIS é id_pais.
Neste tipo de CONSTRAINT podem ser criados relacionamentos que utilizem mais de uma coluna como por exemplo a chave primária composta. Para definir o nome da CONSTRAINT continue a utilizar o padrão nome_tabela+nome_do_atributo+tipo_constraint. Observe algumas regras:

1. caso o tipo de dados da coluna na tabela inicial e na tabela referenciada sejam diferentes, será apresentado um erro;
2. se a tabela referenciada não possua chave primária a foreign key será estabelecida sobre a chave primária da tabela referenciada;

O uso de chaves estrangeiras garante que não existirão tuplas ou linhas órfãs nas tabelas-filhas (tabelas que possuem dados que devem estar cadastrado previamente em outra tabela, denominada tabela mãe).
Se você tentar apagar a TB_PAIS com o comando DROP TABLE TB_PAIS observara uma mensagem de erro. O SGBD agora sabe que a tabela TB_PAIS é consultada e tem vínculo devido a chave estrangeira com a TB_PILOTO de forma que para apagar a TB_PAIS você deve primeiro apagar a TB_PILOTO e depois a TB_PAIS. Experimente:

```
DROP TABLE TB_PILOTO;
DROP TABLE TB_PAIS;
```

Agora você vai experimentar definir um campo como único. Para isso, você pode utilizar o script que estamos trabalhando. Você vai definir uma ou mais colunas que não podem ter valor repetido em mais de uma linha da tabela. Por exemplo, não existem dois países com o mesmo nome, mas este campo não é a chave primária da TB_PAIS. Então experimente:

```
CREATE TABLE TB_PAIS (
id_pais          NUMBER(5),
nm_pais          VARCHAR2(50) UNIQUE,
nr_populacao     VARCHAR2(10),
CONSTRAINT tb_pais_id_pais_pk PRIMARY KEY (id_pais),
CONSTRAINT tb_pais_nm_pais_nn CHECK (nm_pais IS NOT NULL)
);
```

Ou pode declarar a CONSTRAINT

```
CREATE TABLE TB_PAIS (
id_pais          NUMBER(5),
nm_pais          VARCHAR2(50),
nr_populacao     VARCHAR2(10),
CONSTRAINT tb_pais_id_pais_pk PRIMARY KEY (id_pais),
CONSTRAINT tb_pais_nm_pais_nn CHECK (nm_pais IS NOT NULL),
CONSTRAINT tb_pais_nm_pais_un UNIQUE (nm_pais)
);
```

Outra definição que você pode atribuir a um campo é definir um valor padrão que se o usuário não informar ao inserir um registro o SGBD insere automaticamente. Para isso utilizamos o comando DEFAULT. Para exemplificar se um piloto for cadastrado sem o campo dt_nascimento o comando a seguir irá fazer que o SGBD pegue a data e hora atual do servidor e preenche o campo.

```
CREATE TABLE TB_PILOTO (
id_piloto      NUMBER(5),
nm_piloto      VARCHAR2(50),
dt_nascimento  DATE DEFAULT SYSDATE,
id_pais        NUMBER(5),
CONSTRAINT tb_piloto_id_piloto_pk PRIMARY KEY (id_piloto),
CONSTRAINT tb_piloto_id_pais_fk   FOREIGN KEY (id_pais)
           REFERENCES TB_PAIS (id_pais)
);
```

Outra restrição que você pode definir é o sexo do piloto. Imaginando inicialmente que serão classificados em dois gêneros: M para masculino e F para feminino. Você pode apagar a tabela TB_PILOTO e cria-la com o comando:

```
CREATE TABLE TB_PILOTO (
id_piloto      NUMBER(5),
nm_piloto      VARCHAR2(50),
dt_nascimento  DATE DEFAULT SYSDATE,
id_pais        NUMBER(5),
ie_sexo        CHAR(1),
CONSTRAINT tb_piloto_ie_sexo_ck   CHECK (ie_sexo IN ('M','F')),
CONSTRAINT tb_piloto_id_piloto_pk PRIMARY KEY (id_piloto),
CONSTRAINT tb_piloto_id_pais_fk   FOREIGN KEY (id_pais)
           REFERENCES TB_PAIS (id_pais)
);
```

A CONSTRAINT CHECK define um conjunto de valores permitidos ou uma condição para a inserção de dados em uma determinada coluna. Neste caso o comando IN cria uma lista onde qualquer elemento da lista pode ser inserido. Você pode ainda determinar uma expressão ou regra de validação em um CHECK. A sintaxe da CONTRAINT CHECK e expressa:

```
CONSTRAINT nome_da_constraint CHECK (nome_da_coluna condição)
```

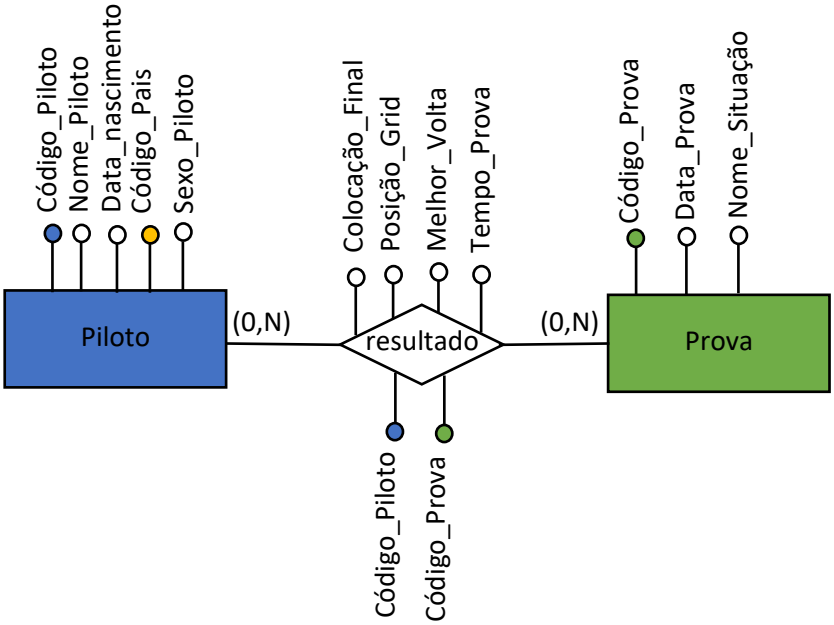
Onde:

nome_da_constraint é a mensagem que será exibida caso a restrição seja violada
nome_da_coluna é a coluna que será validade e testada
condição é a expressão SQL que deve ser validada

Por exemplo:

```
CREATE TABLE TB_PILOTO (
id_piloto      NUMBER(5),
nm_piloto      VARCHAR2(50),
dt_nascimento  DATE DEFAULT SYSDATE,
id_pais        NUMBER(5),
ie_sexo        CHAR(1),
CONSTRAINT tb_piloto_ie_sexo_ck    CHECK (ie_sexo IN ('M','F')),
CONSTRAINT tb_piloto_id_piloto_pk PRIMARY KEY (id_piloto),
CONSTRAINT tb_piloto_id_pais_fk    FOREIGN KEY (id_pais)
           REFERENCES TB_PAIS (id_pais),
CONSTRAINT tb_piloto_nm_piloto_ck CHECK (nm_piloto = UPPER(nm_piloto))
);
```

Neste caso a CONTRAINT tb_piloto_nm_piloto_ck vai validade e só irá aceitar se os nomes forem cadastrados com todos os caracteres maiúsculos. Caso o usuário tente inserir ou alterar o nome de um piloto com caracteres minúsculos o SGBD irá apontar a violação da regra e a transação será recusada.
Agora imagine que você deseja acrescentar as seguintes tabelas:



Temos o SQL:

```
CREATE TABLE TB_PROVA (
id_prova       NUMBER(5),
dt_prova       DATE,
st_prova       VARCHAR2(20),
CONSTRAINT tb_prova_id_prova_pk PRIMARY KEY (id_prova)
);
```

O comando é simples e fácil de implementar. O que é importante destacar neste exemplo é a próxima tabela que será uma tabela de um relacionamento N:N. Neste caso, a chave primária será composta por duas colunas, sendo que estas colunas são chave estrangeiras.

```
CREATE TABLE TB_RESULTADO (
id_piloto             NUMBER(5),
id_prova              NUMBER(5),
nr_colocacao_final    NUMBER(2),
nr_posicao_grid       NUMBER(2),
hr_tempo_prova        TIMESTAMP,
nr_melhor_volta       TIMESTAMP,
CONSTRAINT tb_resultado_pk            PRIMARY KEY (id_piloto,id_prova),
CONSTRAINT tb_resultado_id_piloto_fk  FOREIGN KEY (id_piloto)
           REFERENCES TB_PILOTO (id_piloto),
CONSTRAINT tb_resultado_id_prova_fk   FOREIGN KEY (id_prova)
           REFERENCES TB_PROVA  (id_prova)
);
```

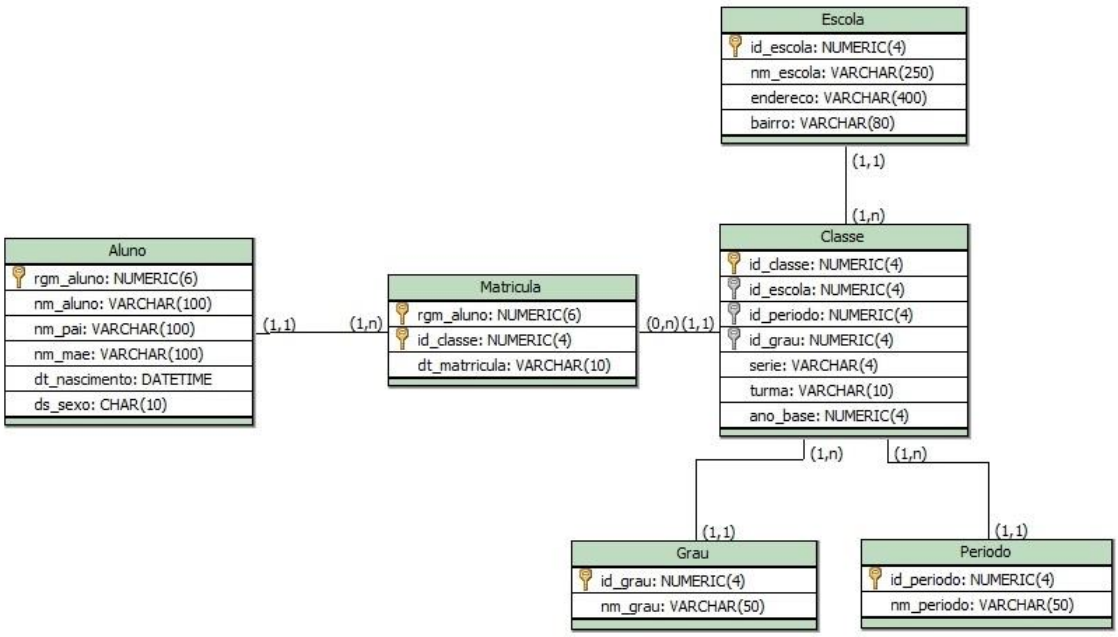
Observe como a chave primaria foi declarada com duas colunas, está é uma chave primaria composta. As chaves primarias podem ter inúmeras colunas, sendo definidas no momento do projeto.

Síntese

Nesta quinta aula, você conheceu as principais etapas na normalização de um banco de dados relacional e pode compreender quais os custos e benefícios de normalizar um banco de dados. Você pode conhecer dois modelos de padronização e como as ferramentas case podem lhe ajudar. Por fim, você inicia o SQL, criando sua conexão, usuário, aprendendo como representar os tipos de dados no SQL e criando suas primeiras tabelas.

Atividade

1. Com base no esquema diagramático de Engenharia da Informação defina o script SQL



Glossário

Normalização – processo para refinar e corrigir as estruturas de dados
Formas Normais – estágios que atuam para a Normalização do banco de dados
Ferramenta CASE - ferramentas de engenharia de software que são projetadas para lhe auxiliar
Forward engineer - à partir de um DER consegue-se criar automaticamente o modelo físico
Reverse engineer - à partir de um banco de dados físico, já implementado, a ferramenta consegue criar o diagrama DER automaticamente

Trigramação - cadeia de caracteres, normalmente constituída por três letras da entidade
Qualificador nome descritivo - todas as entidades e atributos do DER devem de alguma forma ser representados por símbolos contido na tabelas de símbolos
SQL - Structured Query Language ou linguagem estruturada de consulta
ANSI - American National Standard Institute
DDL - Linguagem de definição de dados
DML - Linguagem de manipulação de dados
Script – conjunto de comandos sql dispostos em uma lógica sequencial
Esquema de banco de dados - grupo de objetos relacionados entre si