

Competências, Habilidades e Bases Tecnológicas da disciplina de Banco de Dados I.

I.3 BANCO DE DADOS I					
Função: Modelagem e implementação de bancos de dados					
Classificação: Planejamento					
Atribuições e Responsabilidades					
Modelar bancos de dados relacionais. Implementar modelos de bancos de dados relacionais.					
Valores e Atitudes					
Estimular a organização. Estimular o interesse na resolução de situações-problema. Promover ações que considerem o respeito às normas estabelecidas.					
Competências			Habilidades		
1. Desenvolver modelo de banco de dados relacional.			1.1 Pesquisar as necessidades de informações do sistema. 1.2 Normalizar tabelas de bancos de dados. 1.3 Correlacionar tabelas.		
2. Implementar modelo com a utilização de sistema gerenciador de bancos de dados relacionais.			2.1 Criar objetos no banco de dados com a utilização de linguagem de definição de dados.		
Bases Tecnológicas					
Conceitos de bancos de dados <ul style="list-style-type: none">Modelo relacional;Sistemas gerenciadores de bancos de dados (SGBDR);Arquitetura cliente/servidor;Linguagem estruturada de consulta (SQL).					
Bancos de dados relacionais <ul style="list-style-type: none">Entidades, atributos e relacionamentos;Integridade referencial.					
Modelagem E-R e Normalização <ul style="list-style-type: none">Ferramenta CASE para criação de Diagramas E-R;Tipos de dados e NULL;Formas normais (1FN, 2FN e 3FN);Desnormalização;Especialização e generalização.					
Linguagem de definição de dados (DDL) com SGBDR <ul style="list-style-type: none">Criação de banco de dados;Criação de objetos (tabelas, colunas, chaves e índices);Alteração e exclusão de objetos.					
Carga horária (horas-aula)					
Teórica	00	Prática Profissional	60	Total	60 Horas-aula
Teórica (2,5)	00	Prática Profissional (2,5)	50	Total (2,5)	50 Horas-aula

Observações a considerar:

1-) Comunicação de alunos com alunos e professores:

- Microsoft Teams;
- Criação de grupo nas redes sociais também é interessante.

2-) Uso de celulares:

Para o bom andamento das aulas, recomendo que utilizem os celulares em *vibracall*, para não atrapalhar o andamento da aula.

3-) Material das aulas:

A disciplina trabalha com **NOTAS DE AULA** que são disponibilizadas ao final de cada aula.

4-) Prazos de trabalhos e atividades:

Toda atividade solicitada terá uma data limite de entrega, de forma alguma tal data será postergada ou seja, se não for entregue até a data limite a mesma receberá menção I, isso tanto para atividades entregues de forma impressa, digital ou no Microsoft Teams.

5-) Qualidade do material de atividades:

- Impressas ou manuscritas:
Muita atenção na qualidade do que será entregue, atividades sem grampear, faltando nome e número de componentes, rasgadas, amassadas, com rebarba de folha de caderno e etc. serão desconsiderados por mim.
- Digitais:
Ao enviarem atividades para o e-mail, ou inseridas no Microsoft Teams, **SEMPRE** deverá ter o nome da atividade que está sendo enviada, e no corpo do e-mail ou da atividade deverá ter o(s) nome(s) do(s) integrante(s), sem estar desta forma a atividade será **DESCONSIDERADA**.

6-) Menções e critérios de avaliação:

Na ETEC os senhores serão avaliados por MENÇÃO, onde temos:

- MB - Muito Bom;
- B - Bom;
- R - Regular;
- I - Insatisfatório.

Cada trimestres poderemos ter as seguintes formas de avaliação:

- Avaliação teórica;
- Avaliação prática (a partir do 2º trimestre, e as turmas do 2º módulo em diante);
- Seminários;
- Trabalhos teóricos e/ou práticos;
- Assiduidade;
- Outras que se fizerem necessário.

Caso o aluno tenha alguma menção I em algum dos trimestres será aplicada uma recuperação, que poderá ser em forma de trabalho prático ou teórico.

1. Recomendações bibliográficas:

CASTRO, Eduardo. Modelagem Lógica de Dados: construção básica e simplificada. São Paulo. Ciência Moderna, Agosto 2012.

MARTELLI, Richard; FILHO, Ozeas; CABRAL, Alex. Modelagem e banco de dados. São Paulo. Senac, 2017.

PEREIRA, Paloma. Introdução a banco de dados. São Paulo. Senac, 2020.

NACIONAL, Senac. Modelagem de dados. Rio de Janeiro. Senac, 2014.

2. Introdução: Dados, Informação e conhecimento

Dados, informação e conhecimento, lidamos com esses conceitos o tempo todo, seja em casa, nas empresas, escolas, igreja etc. Ouvimos muitos termos relacionados como processamento de dados, sistemas de informação, gestão de conhecimento, arquitetura da informação, coleta de dados, base de conhecimentos, entre outros. Mas qual a diferença entre Dados, Informação e Conhecimento?

2.1. Dados

Dados são códigos que constituem a matéria prima da informação, ou seja, é a informação não tratada. Os dados representam um ou mais significados que isoladamente não podem transmitir uma mensagem ou representar algum conhecimento.

Em uma pesquisa eleitoral por exemplo, são coletados dados, isto é, cada participante da pesquisa fornece suas opiniões e escolhas sobre determinados candidatos, mas essas opiniões não significam muita coisa no âmbito da eleição. Só depois de ser integrada com as demais opiniões é que teremos algo significativo.

2.2. Informações

Informações são dados tratados. O resultado do processamento de dados são as informações. As informações têm significado, podem ser tomadas decisões ou fazer afirmações considerando as informações.

No exemplo da pesquisa eleitoral, os pesquisadores retêm dados dos entrevistados, mas quando inseridos nos sistemas e processados produzem informações e essas informações diz que tem mais chance de ser eleito, entre outras.

Desta forma podemos dizer que as informações é o conjunto de dados que foram processados, seja por meio eletrônico, mecânico ou manual e que produziu um resultado com significado.

2.3. Conhecimento

O conhecimento vai além de informações, pois ele além de ter um significado tem uma aplicação.

Conhecimento é o ato ou efeito de abstrair ideia ou noção de alguma coisa, como por exemplo: conhecimento das leis; conhecimento de um fato (obter informação); conhecimento de um documento; termo de recibo ou nota em que se declara o aceite de um produto ou serviço; saber, instrução ou cabedal científico (homem com grande conhecimento).

As informações são valiosas, mas o conhecimento constitui um saber. Produz ideias e experiências que as informações por si só não serão capaz de mostrar. Se informação é dado trabalhado, então conhecimento é informação trabalhada.

3. O que é um banco de dados?

Bancos de dados, (ou bases de dados), são conjuntos de dados com uma estrutura regular que organizam informação. Um banco de dados normalmente agrupa informações utilizadas para um mesmo fim.

Um banco de dados é usualmente mantido e acessado por meio de um software conhecido como Sistema Gerenciador de Banco de Dados (SGBD). Muitas vezes o termo banco de dados é usado como sinônimo de SGBD.

O modelo de dados mais adotado hoje em dia é o modelo relacional, onde as estruturas têm a forma de tabelas, compostas por linhas e colunas.

Resumindo, um banco de dados é uma coleção de dados relacionados. Entende-se por dado, todo elemento relevante que pode ser armazenado e que apresenta algum significado implícito dentro do contexto ao qual ele se aplica. Por exemplo, num sistema bancário, uma pessoa é identificada pelo seu CPF (cliente). Em um sistema escolar a pessoa é identificada pelo seu número de matrícula (aluno). Além disso, os dados que serão armazenados em cada situação podem diferir consideravelmente.

4. Modelo Relacional

O modelo relacional é uma teoria matemática desenvolvida por Edgar Frank Codd para descrever como as bases de dados devem funcionar. Embora esta teoria seja a base para o software de bases de dados relacionais, muito poucos sistemas de gestão de bases de dados seguem o modelo de forma restrita, e todos têm funcionalidades que violam a teoria, desta forma variando a complexidade e o poder. A discussão se esses bancos de dados merecem ser chamados de relacional ficou esgotada com tempo, com a evolução dos bancos existentes.

De acordo com a arquitetura ANSI / SPARC em três níveis, os Bancos de dados relacionais consistem de três componentes:

- uma coleção de estruturas de dados, formalmente chamadas de relações, ou informalmente tabelas, compondo o nível conceitual;
- uma coleção dos operadores, a álgebra e o cálculo relacionais, que constituem a base da linguagem SQL; e

• uma coleção de restrições da integridade, definindo o conjunto consistente de estados de base de dados e de alterações de estados. As restrições de integridade podem ser de quatro tipos:

- domínio (ou tipo de dados);
- atributo;
- relações;
- restrições de base de dados.

De acordo com o Princípio de Informação: toda informação tem de ser representada como dados; qualquer tipo de atributo representa relações entre conjuntos de dados.

Nos bancos de dados relacionais os relacionamentos entre as tabelas não são codificados explicitamente na sua definição. Em vez disso, se fazem implicitamente pela presença de atributos chave. As bases de dados relacionais permitem aos utilizadores (incluindo programadores) escreverem consultas (queries), reorganizando e utilizando os dados de forma flexível e não necessariamente antecipada pelos projetistas originais. Esta flexibilidade é especialmente importante em bases de dados que podem ser utilizadas durante décadas, tornando as bases de dados relacionais muito populares no meio comercial.

Um dos pontos fortes do modelo relacional de banco de dados é a possibilidade de definição de um conjunto de restrições de integridade. Estas definem os conjuntos de estados e mudanças de estado consistentes do banco de dados, determinando os valores que podem e os que não podem ser armazenados.

5. Aplicações de bancos de dados

Bancos de dados são usados em muitas aplicações, enquanto atravessando virtualmente a gama inteira de software de computador. Bancos de dados são o método preferido de armazenamento para aplicações multiusuárias grandes onde a coordenação entre muitos usuários é necessária. Até mesmo usuários individuais os acham conveniente, entretanto, e muitos programas de correio eletrônico e os organizadores pessoais estão baseado em tecnologia de banco de dados standard.

6. Aplicativo de Banco de Dados

Um Aplicativo de Banco de dados é um tipo de software exclusivo para gerenciar um banco de dados. Aplicativos de banco de dados abrangem uma vasta variedade de necessidades e objetivos, de pequenas ferramentas como uma agenda, até complexos sistemas empresariais para desempenhar tarefas como a contabilidade.

O termo "Aplicativo de Banco de dados" usualmente se refere a softwares que oferecem uma interface para o banco de dados. O software que gerencia os dados é geralmente chamado de sistema gerenciador de banco de dados (SGBD) ou (se for embarcado) de "*database engine*".

Exemplos de aplicativos de banco de dados são *Firebird*, *Microsoft Access*, *dBASE*, *FileMaker*, *MySQL*, *PostgreSQL*, *Microsoft SQL Server*, *Informix* e *Oracle*.

7. Exemplo prático de análise

A seguir teremos o exemplo da descrição de um banco de dados acadêmico, onde é especificada somente a necessidade para este problema, e neste caso, é o que analisaremos em nossa aula agora.

7.1. Descrição do Banco de dados Acadêmico

O banco de dados descrito representado abaixo é de um pequeno sistema escolar, onde existem basicamente dois componentes que são os alunos matriculados na instituição, bem como as notas obtidas por eles em todas as avaliações realizadas durante um período escolar.

Uma vez definido o escopo da aplicação, ou seja, o seu propósito, o próximo passo é identificar os elementos que a constituem, e por consequência definir todos os dados relevantes para cada item existente. Esses elementos são comumente chamados de entidades, e que por questões de facilidade, são representadas por tabelas.

Matrícula	Nome	Série	Turma	Telefone	Data de Nascimento
1	José da Silva	Oitava	1	(31) 3666-9090	05-10-1986
2	Ana Maria	Sétima	1	(21) 1234-4567	12-11-1983
3	Paulo Simon	Quinta	1	(31) 8890-7654	17-04-1984
4	Carla Beatriz	Sexta	1	(45) 9946-8989	30-07-1979
5	Ana Paula	Oitava	2	(62) 7878-0909	22-01-1980
6	Joana Prado	Quinta	2	(35) 8878-0099	06-05-1986

Tabela 1: Definição de ENTIDADE alunos

Matrícula	Data do Teste	Ponto
1	25-03-2004	5.5
2	25-03-2004	6
3	25-03-2004	8
4	25-03-2004	10
5	25-03-2004	7.8
6	25-03-2004	4.6
1	18-05-2004	7.2
2	18-05-2004	9.5
6	18-05-2004	10

Tabela 2: Definição de PONTUAÇÕES

A segunda entidade identificada no problema são as pontuações obtidas por cada aluno. Vale ressaltar que durante um período letivo poderão existir várias avaliações, geralmente em datas diferentes, onde deverão ser armazenados os resultados de todos os alunos para cada um destes testes. A tabela 2 descreve a entidade pontuações, que serve para o propósito exposto anteriormente.

Cada entidade é representada por uma tabela, sendo que neste universo de discussão ou modelo, existem apenas duas tabelas e um relacionamento entre elas, já que cada entidade aluno está ligada à entidade pontuação. Em aplicações mais complexas, poderão existir inúmeras tabelas e relacionamentos de forma a permitir a representação do problema abordado.

Atributos definem uma entidade

Cada entidade, no exemplo alunos e pontuações, é representada por uma tabela, que por sua vez são constituídas de linhas e colunas. Cada coluna representa um fragmento de dado e o conjunto de todas as colunas constitui a entidade propriamente dita. No contexto de banco de dados cada coluna é chamada de atributo e uma entidade será formada por um ou vários atributos.

Um atributo define uma característica da entidade, no exemplo aluno é constituído de seis atributos, que são o número de matrícula, nome, a série que está cursando, a sua turma, o seu telefone residencial e a data de nascimento. O atributo matrícula possui um papel importante no modelo servindo como identificador único para cada aluno. Em um banco de dados caso ocorram registros com valores idênticos não será possível determinar um contexto que os identifiquem unicamente. Por isso deve existir uma chave ou atributo que identifique unicamente cada registro. Ao observar a Tabela 1, percebe-se que não há dois alunos cadastrados com o mesmo número de matrícula. Portanto, este é o atributo chave da entidade, utilizado para a pesquisa de um registro nesta tabela.

A entidade pontuações necessita identificar o aluno, a data da avaliação e a pontuação atingida pelo aluno. Neste caso, como cada aluno é identificado unicamente pela sua matrícula, este atributo será inserido na tabela de pontuações para permitir associar o aluno à nota registrada, conforme visto na Tabela 2.

Percebe-se que cada atributo possui um conjunto de valores válidos e aceitáveis, que é definido como domínio do atributo. Todas informações vistas na tabela são textuais, isto é, sequências de letras e números, mas é notório que o conjunto de dados contido em cada coluna é diferente umas das outras. No caso de matrícula do aluno, o domínio dos dados é o conjunto dos números inteiros positivos, já que para cada aluno é atribuído um código numérico que denota a ordem em que este foi matriculado na escola. Ou seja, o texto contido nesta coluna é formado por uma combinação de números, portanto não existem letras.

8. Modelagem de dados: modelo conceitual, modelo lógico e físico

A modelagem de dados é uma técnica usada para a especificação das regras de negócios e as estruturas de dados de um banco de dados. Ela faz parte do ciclo de desenvolvimento de um sistema de informação e é de vital importância para o bom resultado do projeto. Modelar dados consiste em desenhar o sistema de informações, concentrando-se nas entidades lógicas e nas dependências lógicas entre essas entidades.

Modelagem de dados ou modelagem de banco de dados envolve uma série de aplicações teóricas e práticas, visando construir um modelo de dados consistente, não redundante e perfeitamente aplicável em qualquer SGBD moderno.

A modelagem de dados está dividida em:

8.1. Modelo conceitual

A modelagem conceitual baseia-se no mais alto nível e deve ser usada para envolver o cliente, pois o foco aqui é discutir os aspectos do negócio do cliente e não da tecnologia. Os exemplos de modelagem de dados vistos pelo modelo conceitual são mais fáceis de compreender, já que não há limitações ou aplicação de tecnologia específica.

A arquitetura desta abstração se dá em três níveis. O mais externo, o nível de visões do usuário, descreve partes do banco que serão visualizadas pelos usuários. No nível intermediário, tem-se o nível conceitual (ou lógico), que descreve quais os dados estão armazenados e seus relacionamentos. Finalmente, no nível mais baixo, está o nível físico, descrevendo a forma como os dados estão realmente armazenados.

O termo negócio refere-se ao problema em questão que se deseja realizar uma modelagem. A intenção de armazenar informações de alunos numa escola, por exemplo, sugere que o negócio seja “Registro acadêmico”. Num outro exemplo, o negócio “Instituição financeira” poderá ser modelado tendo dados de correntistas e saldos.

Razões para a criação do modelo conceitual:

- Descreve exatamente as informações necessárias ao negócio. Para a modelagem, todas as regras do negócio deverão ser conhecidas e, cabe ao projetista, traduzi-las em informações relevantes ao banco;
- Facilita a discussão, seja entre o projetista e o usuário ou entre o projetista e sua equipe de trabalho;
- Ajuda a prevenir erros do futuro sistema;
- Uma forma de documentar o sistema ideal. Um sistema é ideal quando todas suas informações estão modeladas de acordo com certas condições;
- É a base para o projeto físico do banco de dados.

A abordagem utilizada aqui será a representação de dados no modelo relacional, utilizando-se, para tal, o Modelo de Entidade-Relacionamento (MER). O MER é um modelo baseado na percepção do mundo real, que consiste em um conjunto de objetos básicos chamados de entidades e nos relacionamentos entre esses objetos.

Foi proposto por Peter Chen, em 1976, como uma ferramenta de projeto de banco de dados. O MER apresenta como contribuições um maior grau de independência de dados que os modelos convencionais (de redes e hierárquico) e uma unificação de representação destes modelos, através do formalismo gráfico do Diagrama de Entidade-Relacionamento (DER).

São características do MER:

- Modela regras de negócio e não a implementação. A modelagem é dos dados requeridos para o negócio, baseado nas funcionalidades do sistema atual ou a ser desenvolvido. Para modelar um negócio, é necessário conhecer em detalhes sobre do que se trata.
- Possui uma sintaxe robusta, bem definida;
- Técnica amplamente difundida e utilizada. Atualmente, a maioria dos bancos de dados disponíveis no mercado utiliza a abordagem relacional como modelo de dados;
- Diagramas fáceis de entender e alterar.

Os objetivos de uma modelagem entidade-relacionamento são:

- Obter todas as informações requeridas sobre o negócio antes de sua implementação, tornando claras suas dependências;
- Dentro do possível, uma informação aparecer apenas uma vez no banco de dados. Uma modelagem que prevê o armazenamento de uma mesma informação em dois locais

diferentes, deixa o sistema vulnerável quanto a possibilidade destas informações não serem as mesmas. No caso de uma inconsistência dos dados, qual delas deverá ser descartada?

- Facilitar o projeto do banco de dados, possibilitando a especificação de sua estrutura lógica.

Entidade e Instância

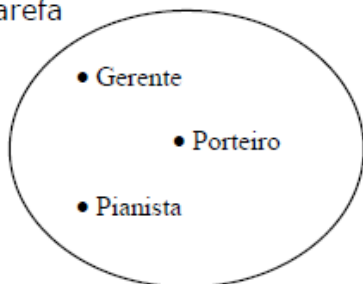
Num MER, uma entidade é um objeto, real ou abstrato, de relevância para o negócio. É uma categoria de ideias que são importantes ao negócio, as quais devem ser traduzidas em informação. Dois importantes aspectos de uma entidade é que possui instâncias e estas instâncias também são de interesse ao negócio.

Pode-se considerar que uma instância identifica individualmente uma entidade. O quadro abaixo mostra alguns exemplos de entidades e instâncias:

Entidade	Instância
Pessoa	João, José, Antônio
Produto	Prego 12x12, File de Peixe Merluza
Tipo de produto	Plástico, papel, madeira
Tarefa	Professor, pianista, gerente
Versão do documento	1.2, 10.5

Observa-se que uma entidade possui várias instâncias e que cada instância está relacionada a uma entidade. Uma entidade representa um conjunto de instâncias que interessam ao negócio. Segue o exemplo a seguir:

Tarefa



Em termos físicos, uma entidade será uma *tabela* do BD e cada instância será uma *linha* (ou *registro* ou *tupla*). Exemplo:

Tabela Tarefa

Código	Descrição
405	Gerente
564	Porteiro
321	Pianista

Entidade

Instâncias

Atributo e Domínio

Um atributo também representa algo significativo ao negócio. Um atributo é uma propriedade de uma entidade. É uma porção de informação que descreve, quantifica, qualifica, classifica e especifica uma entidade. Normalmente, uma entidade possui vários atributos. Interessa, em termos de modelagem conceitual, que estes atributos representem informações relevantes ao negócio. Atributos possuem valores (um número, um caracter, uma data, uma imagem, um som, etc), chamados de tipos de dados ou formato. Para um atributo particular, todas suas instâncias possuem os mesmos formatos. O quadro abaixo apresenta exemplos de entidades, instâncias e atributos.

Entidade	Instância	Atributo
Empregado	João, Antônio	Nome, idade, tamanho pé, dependentes, cidade
Carro	Escort, Gol	Cor, preço, modelo
Tarefa	Gerente	Código, Depto, Valor/hora, descrição

Algumas questões:

- O atributo Idade, da entidade Empregado, não parece ser uma boa escolha. O ideal seria Data Nascimento, ficando o cálculo da idade quando necessário. O armazenamento da informação idade é de difícil, senão impossível, atualização;

- O atributo Tamanho pé, de Empregado, dependerá das regras de negócio. Imaginando que a finalidade de entidade Empregado seja a de armazenar dados sobre funcionários numa

empresa que forneça uniforme de trabalho, o atributo é coerente. Se a empresa não possui esta política, ele é desnecessário;

- Uma importante decisão precisa ser tomada em relação a armazenar uma informação como um atributo ou uma entidade. O atributo Cidade, de Empregado, terá a cidade na qual o empregado reside. Se Cidade fosse uma entidade, alguns possíveis atributos seriam População, Área e Data Fundação. Aqui, novamente a escolha passa pelas regras de negócio. Normalmente, uma informação será um atributo se for de natureza atômica e será uma entidade quando possuir informações que possam (ou necessitem) ser relacionadas a outras entidades.

Em termos físicos, os atributos serão as *colunas* de uma tabela do BD. Exemplo:

Tabela Empregado

Nome	Idade
João	32
Antônio	25

Colunas

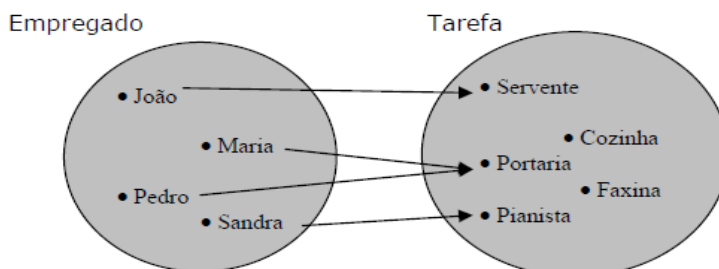
- **Atributo monovalorado ou atômico:** assume um único valor, num certo instante de tempo, para cada instância. Exemplo: Nome de Empregado
- **Atributo composto:** formado por um ou mais subatributos. Exemplo: Cidade, de Empregado. Cidade pode ser composto pelo nome da cidade e o estado.
- **Atributo multivalorado:** assume diversos valores. Seu nome, normalmente, é no plural. Exemplo: Dependentes de Empregado.
- **Atributo determinante:** identifica cada entidade como única. Exemplo: Código, em Tarefa.

Domínio de um atributo: conjunto de valores possíveis para um atributo. Exemplo: Idade deve estar entre 18 e 60, Estado deve ser “RS”, “RJ”, “SP”, etc.

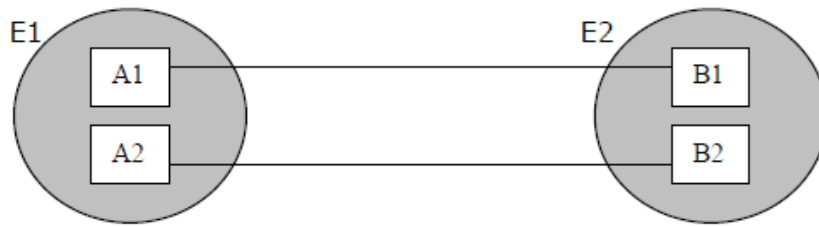
Relacionamentos

É uma estrutura que indica uma associação entre duas ou mais entidades. Alguns exemplos:

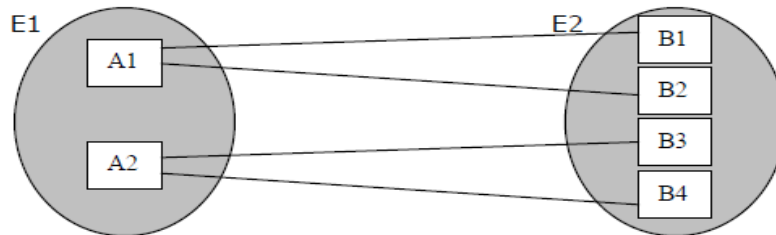
Entidade	Relacionamento	Entidade
Empregado Tarefa	Exerce É exercida por	Tarefa Empregado
Produto Tipo Produto	É classificado por um É uma classificação de	Tipo Produto Produto
Pessoa Reserva	Faz É feita por	Reserva Pessoa



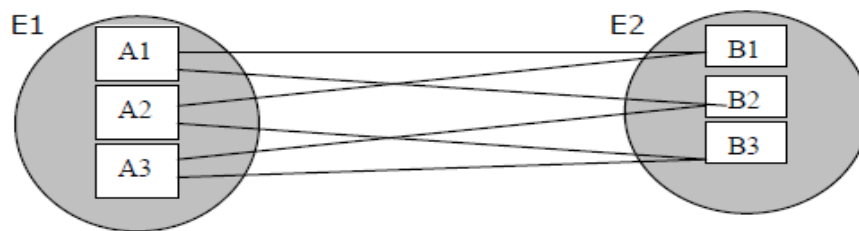
- Todos os empregados exercem tarefas;
- Nenhum empregado tem mais de uma tarefa;
- Nem todas as tarefas são exercidas por empregados;
- Algumas tarefas são exercidas por mais de um empregado.



A cardinalidade um-para-vários (1:N) ocorre quando uma instância de E1 está associada a qualquer número de instâncias de E2, enquanto que uma instância de E2 está associada no máximo a uma instância de E1.



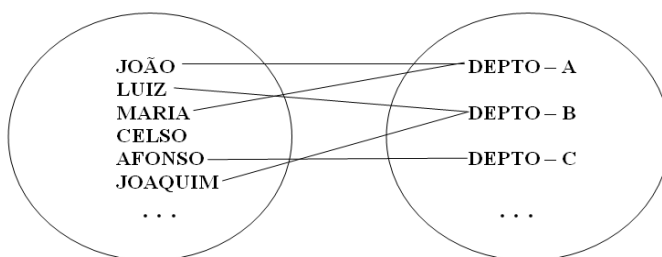
A cardinalidade vários-para-vários (N:N) ocorre quando uma instância de E1 está associada a qualquer número de instâncias de E2 e uma instância de E2 está associada a qualquer número de instâncias de E1.



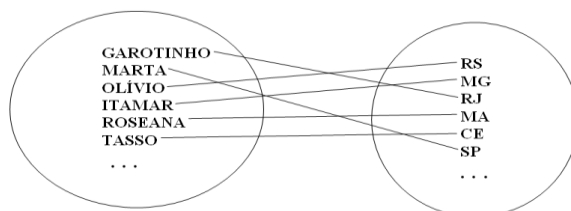
EXERCÍCIOS PARA FIXAÇÃO

Informe a cardinalidade para as situações a seguir

- a) ENTIDADE EMPREGADO ENTIDADE DEPARTAMENTO



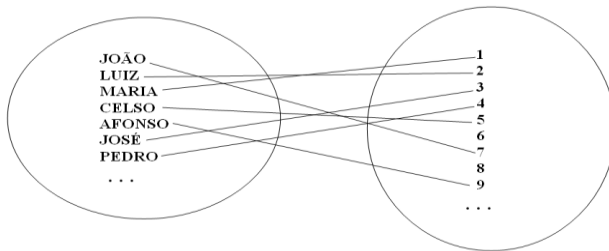
- b) ENTIDADE GOVERNADOR ENTIDADE ESTADO



c-)

ENTIDADE
PASSAGEIRO

ENTIDADE
POLTRONA DO AVIÃO



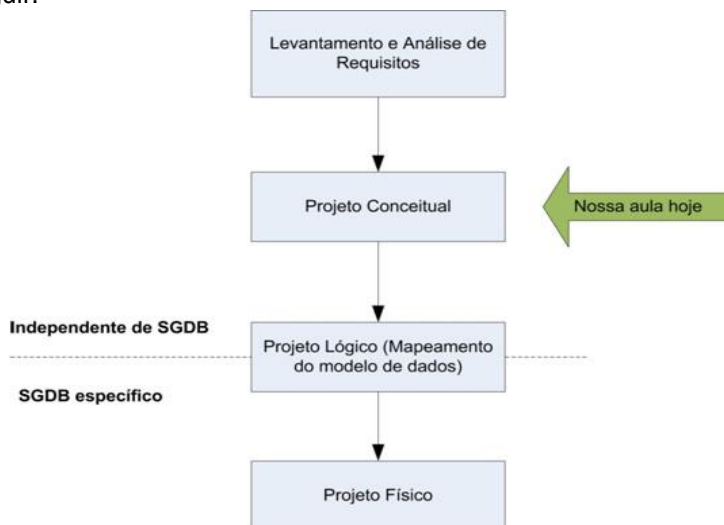
Modelo de Entidade e Relacionamento

História

O modelo de entidades e relacionamentos é um modelo conceitual onde descrevemos o nosso banco de dados. Representamos esse modelo por um diagrama de Entidade e Relacionamento (ER).

A estrutura lógica geral de um banco de dados pode ser expressa graficamente por um Diagrama de Entidade-Relacionamento. É uma representação gráfica do modelo ou parte do modelo. De acordo com o grau de complexidade do negócio ou o nível de detalhamento impresso pelo projetista do banco, um modelo pode ser representado por vários diagramas.

Mas, antes de falar sobre os conceitos, vamos conhecer a história do Modelo de Entidade e Relacionamento. Tudo começou quando o Dr. Peter Chen em 1976 propôs o modelo Entidade-Relacionamento (ER) para projetos de banco de dados. Isso deu uma nova e importante percepção dos conceitos de modelos de dados. O modelo ER proposto pelo Dr. Peter possibilitava ao projetista concentrar-se apenas na utilização dos dados sem se preocupar com estrutura lógica de tabelas. Por esse motivo, o modelo ER é utilizado pelo projeto conceitual para modelar os conceitos do banco de dados de forma independente de SGDB como vemos na figura a seguir.

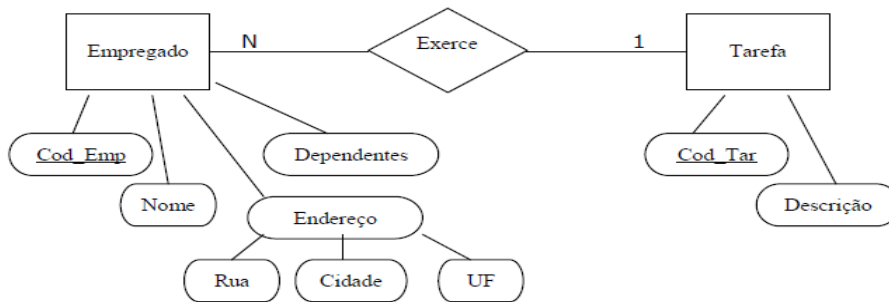


Professor Doutor Peter Chen
Cientista da computação americano
Página oficial: <http://www.csc.lsu.edu/~chen/>

Um DER utiliza-se de um número de elementos gráficos. Infelizmente, não existe uma padronização na representação do diagrama. Serão apresentados dois padrões: um deles segundo Peter Chen e o outro uma pequena variação de um desenvolvido pela Oracle.

Os componentes do Diagrama de Entidade-Relacionamento, de acordo com Peter Chen são:

- Retângulos: representam as entidades
- Elipses: representam os atributos
- Losangos: representam os relacionamentos
- Linhas: ligam atributos a entidades e entidades a relacionamentos.

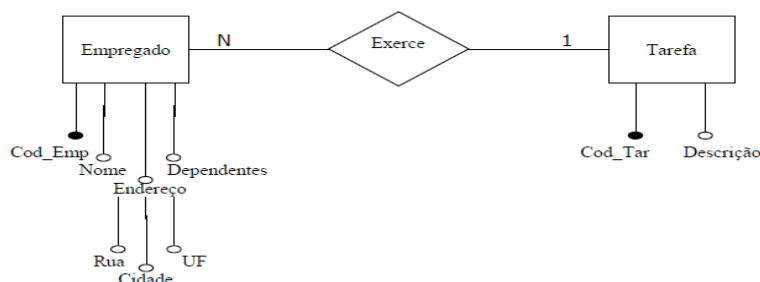


EXEMPLO DE DER SEGUNDO PETER CHEN

Observa-se o seguinte:

- São duas as entidades: Empregado e Tarefa
- Atributos da entidade Empregado:
 - Cod_Emp (determinante - está sublinhado)
 - Nome (monovalorado)
 - Dependentes (multivalorado)
 - Endereço (composto)
 - Rua (monovalorado)
 - Cidade (monovalorado)
 - UF (monovalorado)
- Atributos da entidade Tarefa:
 - Cod_Tar (determinante - está sublinhado)
 - Descrição (monovalorado)
- O relacionamento entre Empregado e Tarefa possui cardinalidade 1:n

Uma variação do diagrama anterior é apresentada a seguir. Nota-se que os atributos determinantes não mais são sublinhados, pois possuem símbolos diferentes dos demais, e adotaremos este modelo para nossas aulas.



Exemplo para fixação na prática em aula

Variação 1:

Banco de dados de um hospital onde possuem médicos. Criem os atributos determinantes para as duas entidades, mais os que julgarem necessário, por fim relacione as duas entidades e determine a cardinalidade. **Lembre-se que o hospital precisará pelo menos de um médico cadastrado.**

Variação 2:

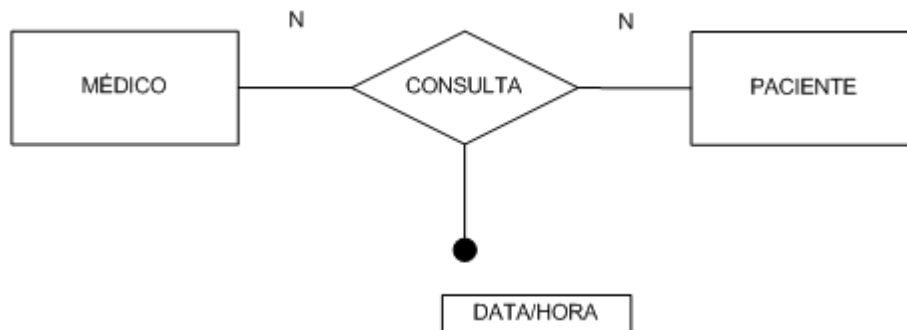
Levando em consideração o mesmo banco já criado, acrescente a entidade pacientes que serão atendidos pelos médicos, nesta nova entidade, crie os atributos necessários (determinantes e não determinantes) e por fim relacione com a entidade médico e determine a cardinalidade. **Lembre-se que um médico pode atender nenhum, um ou vários pacientes.**

Variação 3:

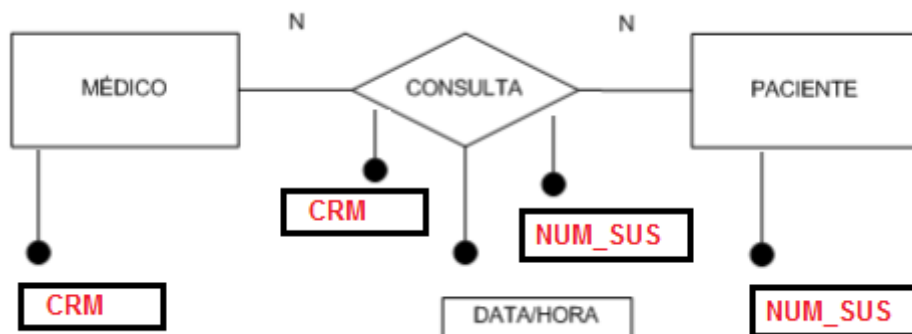
Ainda levando em consideração o mesmo banco já criado, acrescente a entidade medicamento, que serão fornecidos aos pacientes atendidos, nesta nova entidade, crie os atributos necessários (determinantes e não determinantes) e por fim relacione com a entidade paciente e determine a cardinalidade. **Lembre-se que um paciente pode ou não receber medicamentos.**

Há casos onde pode ser necessário relacionar ocorrências de mesmas entidades mais de uma vez. Por exemplo, em um modelo de consultas médicas, determinado paciente pode realizar consultas mais de uma vez com o mesmo médico.

Neste caso, podemos utilizar um atributo identificador no relacionamento (data/hora).



Isto também chamará de tabela de relacionamento, ou seja, precisamos ter os atributos determinantes tanto da entidade médico como da entidade paciente no relacionamento, para todo o relacionamento de N para N, o que denominamos em banco de dados como atributo estrangeiro ou chave estrangeira e assim o mesmo se tornará uma tabela. Veja:



Para exercitarmos este conceito vamos praticar as variações de acordo com o seguinte problema proposto:

Variação 1:

Um banco de dados de uma loja, primeiramente quer cadastrar os seus Clientes e Fornecedores, crie os atributos (principalmente o(s) determinante(s)) necessários para cada Entidade.

Variação 2:

Levando em consideração o banco já criado, agora iremos acrescentar os Produtos desta loja, é claro que estes produtos são vendidos a clientes, crie os atributos

necessários para a Entidade, e os atributos do relacionamento, não se esqueçam dos atributos estrangeiros e da cardinalidade.

Variação 3:

Neste mesmo banco implemente o relacionamento de Produto com Fornecedor, que chamará Fornece que deverá ser uma tabela, defina os atributos do relacionamento, e a cardinalidade.

Variação 4:

Crie mais uma Entidade e seu respectivo Relacionamento.

História e Implementações do SQL

No início dos anos 70, o trabalho produtivo do colega de pesquisa da IBM Dr. E. F. Codd levou ao desenvolvimento de um produto modelo de dado relacional chamado SEQUEL ou Linguagem de Consulta em Inglês Estruturado (em inglês: Structured English Query Language). SEQUEL ultimamente se transformou em SQL ou Linguagem de Consulta Estruturada (em inglês: Structured Query Language).

IBM, junto com outros fornecedores de banco de dados relacionais, queria um método padronizado para acessar e manipular dados em um banco de dados relacional. Embora IBM tenha sido a primeira a desenvolver a teoria de banco de dados relacional, a Oracle foi a primeira a comercializar a tecnologia. Através do tempo, SQL se provou popular o suficiente no mercado de trabalho para atrair a atenção do American National Standards Institute (ANSI), que lançou padrões para SQL em 1986, 1989, 1992, 1999, 2003 e 2006. Este texto cobre o padrão ANSI 2003, pois o padrão 2006 lida com elementos do SQL fora do escopo dos comandos descritos neste livro. (Em essência, o padrão SQL2006 descreve como o XML deveria ser usado no SQL.)

Desde 1986, várias linguagens concorrentes permitiram programadores e desenvolvedores a acessarem e manipularem dados relacionais. No entanto, poucos foram fáceis de aprender ou aceitos universalmente como SQL. Programadores e administradores agora têm o benefício de serem capazes de aprender uma única linguagem que, com pequenos ajustes, é aplicável a uma vasta variedade de plataformas de banco de dados, aplicações e produtos, como exemplos temos:

- MySQL;
- Oracle Database 11g
- PostgreSQL;
- SQL Server 2008 da Microsoft.

Breve História do MySQL

O MySQL surgiu a partir da necessidade da equipe que criou o SGBD, de utilizar algum mecanismo que permitisse a conexão de tabelas criadas na linguagem SQL para um determinado fim. A princípio, o grupo iria utilizar o mSQL(Mini SQL), mas logo perceberam que esta ferramenta não era rápida o suficiente para atender às necessidades do projeto. O jeito foi criar uma solução própria. Nascia o MySQL.

O MySQL foi criado por Michael Widenius na companhia suíça TcX. Por volta de 1979 Michael desenvolveu um banco de dados chamado UNIREG, "sendo rescritos em várias linguagens desde então". Em 1994, a empresa TcX começou o desenvolvimento de aplicações baseadas na Web, tendo como base o banco UNIREG, porém esse banco possuía muito "overhead" para obter sucesso em uma aplicação para geração de páginas dinâmicas na Web. Então a empresa TcX começou a procurar por outro banco o mSQL(Mini SQL), uma ferramenta baseada em SQL mas com características pobres não possuindo por exemplo suporte a índices, e com desempenho inferior ao UNIREG.

Foi então que o desenvolvedor do banco UNIREG contatou o David Hughes criador do mSQL(Mini SQL), para saber do interesse dele em unir os dois bancos. Sendo positivo o interesse de David, a empresa TcX resolveu desenvolver um novo banco, mas mantendo ao máximo a compatibilidade com mSQL(Mini SQL). TcX foi esperta o suficiente para não reinventar o que já estava bem feito, ela construiu seu servidor baseado na estrutura que já estava montada do UNIREG e utilizou grande número de utilitários escritos para mSQL(Mini SQL) e fez API's para o novo servidor praticamente iguais ao mSQL(Mini SQL). Como resultado usuários do mSQL(Mini SQL) que decidissem mudar para o novo servidor da TcX, teriam apenas que fazer

pequenas e simples mudanças nos códigos existentes.

Então foi em maio de 1995 que, definitivamente, a primeira versão do MySQL foi lançada. Um dos parceiros da TcX sugeriu a distribuição do servidor na Internet, o objetivo disso era a utilização de um modelo pioneiro desenvolvido por Aladdin Peter Deutsch. O resultado foi um maior flexibilidade em sem "copyright", que fez do MySQL mais difundido gratuitamente do que mSQL (Mini SQL).

Um fato importante a ser destacado sobre o MySQL é que esse SGBD também possui uma licença comercial, isto é, paga. Neste caso, é possível obter suporte diferenciado dos desenvolvedores.

Vale ressaltar também que, em fevereiro de 2008, o MySQL foi comprado pela Sun Microsystems, que pagou a quantia de 1 bilhão de dólares pela aquisição.

O Banco de Dados MySQL

O MySQL é um dos sistemas de gerenciamento de banco de dados mais populares que existe e, por ser otimizado para aplicações Web, é amplamente utilizado na internet. É muito comum encontrar serviços de hospedagem de sites que oferecem o MySQL e a linguagem PHP, justamente porque ambos trabalham muito bem em conjunto.

Outro fator que ajuda na popularidade do MySQL é sua disponibilidade para praticamente qualquer sistema operacional, como Linux, FreeBSD (e outros sistemas baseados em Unix), Windows e Mac OS X. Além disso, o MySQL é um software livre sob licença GPL (General Public License), o que significa que qualquer um pode estudá-lo ou alterá-lo conforme a necessidade.

Entre as características técnicas do SGBD MySQL, estão:

- Alta compatibilidade com linguagens como PHP, Java, Python, C#, Ruby e C/C++;
- Baixa exigência de processamento (em comparação como outros SGBD);
- Vários sistemas de armazenamento de dados (database engine), como MyISAM, MySQL Cluster, CSV, Merge, InnoDB, entre outros;
- Recursos como transactions (transações), conectividade segura, indexação de campos de texto, replicação, etc;
- Instruções em SQL como indicam o nome.

Utilização

Uma característica fundamental do MySQL, quiçá na origem do seu sucesso, é ser desenvolvido em código aberto e funcionar num grande número de sistemas operacionais: Windows, Linux, FreeBSD, BSDI, Solaris, Mac OS X, SunOS, SGI, etc.

É reconhecido pelo seu desempenho e robustez e também por ser multitarefa e multiusuário. A própria Wikipédia, usando o programa MediaWiki, utiliza o MySQL para gerenciar seu banco de dados, demonstrando que é possível utilizá-lo em sistemas de produção de alta exigência e em aplicações sofisticadas.

No passado (até à versão 3.x), devido a não possuir funcionalidades consideradas essenciais em muitas áreas, como stored procedures, two-phase commit, subselects, foreign keys ou integridade referencial, era frequentemente considerado um sistema mais "leve" e para aplicações menos exigentes, sendo preterido por outros sistemas como o PostgreSQL.

Características do MySQL

- Banco de dados de código aberto e gratuito;
- As tabelas criadas podem ter tamanho de até 4 GB;
- Suporta diferentes plataformas: Win32, Linux, FreeBSD, Unix, etc;
- Suporte às API's das seguintes linguagens: PHP, Perl, C, C++, Java, Python, etc;
- Suporte a múltiplos processadores;
- Um sofisticado sistema de senhas criptografadas flexível e Seguro.
- Suporte à ODBC, você pode facilmente conectar o Access a um banco de dados do MySQL;
- Suporta até 16 índices por tabela;
- Código fonte escrito em C e C++ e testado com uma variedade de diferentes compiladores;
- O Cliente conecta no MySQL através de conexões TCP/IP;
- Capacidade para manipular bancos com até 50 milhões de registros;

- Reduz a administração, engenharia e a sustentação custa por até 50%.

O Que o MySQL Faz de Melhor

- Aplicações Web;
- Aplicações de nível corporativo;
- Suporte a código fonte aberto;
- Requisitos de sistema baixo;
- Tabelas com tamanho grande;
- Estabilidade.

Segurança no MySQL

O MySQL possui componentes de segurança contra ameaças externas como crackers e outros, e também proteger os dados dos próprios usuários. O mysql apresenta vários níveis de segurança em relação ao acesso. Todas as informações de segurança estão armazenadas no banco mysql.

A filosofia de segurança em banco de dados refere-se a fornecer ao usuário apenas o que é essencial para o seu trabalho.

O MySQL é Gratuito?

Pessoas confundem "free" com "grátis" o que é comum aqui no Brasil. Mas em se tratando de software este "free" é de open source e não gratuito. Para poder utilizar o MySQL sob a licença GPL (General Public License) e não precisar pagar, o produto desenvolvido precisa ser GPL (General Public License) também, senão, orientamos a compra da licença comercial, com baixo custo, sendo comercializada por servidor, sem limites de usuários e processadores e ainda com garantia perpétua de atualização de versão para o resto da vida.

Empresas Que o Utilizam:

AOL
Departamento De Census dos E. U.
Dow Jones
EarthLink
Ericsson
Google
Hewlett-Packard
Instrumentos De Texas
Lucent
Lufthansa
NASA
Siemens
Steaks De Omaha
Suzuki
Tempo Inc.

Qual o Tamanho Que as Tabelas do MySQL Podem Ter?

No MySQL versão 3.23 o tamanho máximo foi expandido até 8 milhões de terabytes. Com este tamanho de tabela maior permitido, o tamanho máximo efetivo das tabelas para o banco de dados MySQL é normalmente limitado pelas restrições do sistema operacional quanto ao tamanho dos arquivos, não mais por limites internos do MySQL.

O Logo

O logo do golfinho do MySQL foi desenhado pela empresa finlandesa Finnish Advertising Agency Priority em 2001. O golfinho foi escolhido como o símbolo para o banco de dados MySQL pois o animal é esperto, rápido e ágil. O nome do golfinho é Sakila. Esse nome foi

escolhido pelos fundadores da MySQL AB em uma enorme lista de nomes sugeridos pelos usuários numa campanha chamada de "Name the Dolphin" realizada por ideia da empresa.

O nome vencedor foi enviado por Ambrose Twebaze, um desenvolvedor de software Open Source da Swaziland (África). De acordo com Ambrose, "Sakila" é um nome feminino, que tem suas raízes no Siswati, dialeto local de Swaziland.



Principais Tipos de Dados

INT: Representa um valor inteiro. Pode ser com sinal ou sem sinal

REAL: Representa um valor com ponto flutuante. Oferece uma grande precisão e uma extensa faixa de valores

CHAR(n): Representa um valor caractere com tamanho fixo.

TEXT: Representa um valor para caractere com tamanho variável

DATE: Representa um valor de data padrão. Formato: YYYY-MM-DD (2001-01-01)

TIME: Representa um valor de tempo padrão. Armazena a hora de um dia independente de uma data particular. Formato : hh:mm:ss (06:00:00)

Compreendendo os tipos de dados do MySQL

Os tipos de dados que pode ter um campo, podem-se agrupar em três grandes grupos:

1. Tipos numéricos
2. Tipos de Data
3. Tipos de Cadeia

1) Tipos numéricos:

Existem tipos de dados numéricos, que se podem dividir em dois grandes grupos, os que estão em vírgula flutuante (com decimais) e os que não.

Int: número inteiro com ou sem sinal. Com sinal a margem de valores válidos é desde -2147483648 até 2147483647.

Float: número pequeno em vírgula flutuante de precisão simples. Os valores válidos vão desde -3.402823466E+38 até -1.175494351E-38,0 até desde 175494351E-38 até 3.402823466E+38. (é o tipo para ser usado em números reais, preço, e números que usa decimal, ex: 10,32).

2) Tipos data:

Na hora de armazenar datas, há que ter em conta que MySQL não verifica de uma maneira estrita se uma data é válida ou não. Simplesmente comprova que o mês está compreendido entre 0 e 12 e que o dia está compreendido entre 0 e 31.

Date: tipo data armazena uma data. A margem de valores vai desde o 1 de Janeiro de 1001 ao 31 de dezembro de 9999. O formato de armazenamento é de ano-mês-dia.

Time: armazena uma hora. A margem de horas vai desde -838 horas, 59 minutos e 59 segundos. O formato de armazenamento é 'HH:MM:SS'.

3) Tipos de cadeia:

VarChar(n): armazena uma cadeia de longitude variável. A cadeia poderá conter desde 0 até 255 caracteres.

MySQL Workbench

O MySQL Workbench é uma ferramenta gráfica para modelagem de dados, integrando criação e designer que pretende ser uma evolução do já famoso DBDesigner4. A ferramenta possibilita trabalhar diretamente com objetos *schema*, além de fazer a separação do modelo lógico do catálogo de banco de dados.

Toda a criação dos relacionamentos entre as tabelas pode ser baseada em chaves estrangeiras. Outro recurso que a ferramenta possibilita é realizar a engenharia reversa de esquemas do banco de dados, bem como gerar todos os scripts em SQL.

Ferramenta de modelagem repleta de recursos MySQL Workbench apresentar opções que não o deixam para trás em comparação aos outros programas para modelagem de dados. A ferramenta faz sincronia dos modelos de banco de dados, importa e exporta modelos do DBDesigner4, bem como exporta scripts SQL.

A modelagem dos seus bancos de dados pode assumir níveis conceituais, lógicos e físicos. MySQL Workbench apresenta uma arquitetura extensível, sendo possível visualizar a representação de tabelas, funções, entre outros.

Comandos Básicos do SQL

Iremos ver os comandos básicos do SQL, utilizados pela maioria dos bancos de dados, inclusive o MySQL.

Observe que todo comando SQL termina com um ; (**ponto e vírgula**).

Comando Create Database

Este comando permite a criação do banco de dados.

Sintaxe:

CREATE DATABASE < nome_db >;

onde:

- **nome_db** - indica o nome do Banco de Dados a ser criado.

Exemplo:

Create database focus;

Comando Drop Database

Este comando permite a exclusão do banco de dados. Caso existam tabelas criadas no banco, as mesmas serão apagadas.

Sintaxe:

DROP DATABASE < nome_db >;

onde:

- **nome_db** - indica o nome do Banco de Dados a ser criado.

Exemplo:

Drop database focus;

Comando Use

Este comando serve para selecionarmos o banco de dados a ser utilizado.

Sintaxe:

USE < nome_db >;

onde:

- **nome_db** - indica o nome do Banco de Dados a ser criado.

Exemplo:

Use focus;

Comando Create Table

Este comando permite a criação de tabelas no banco de dados.

Sintaxe:

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

onde:

- **nome_table** - indica o nome da tabela a ser criada.
- **nome_atributo** - indica o nome do campo a ser criado na tabela.
- **tipo** - indica a definição do tipo de atributo (integer(n), char(n), ...).

Exemplo:

```
Create table alunos (
    Id_aluno INT(3) NOT NULL,
    nome VARCHAR(40) NOT NULL,
    endereco VARCHAR(50) NOT NULL,
    turma VARCHAR(20) NOT NULL,
    PRIMARY KEY (Id_aluno)
);
```

CREATE TABLE (com Cláusula DEFAULT)

A cláusula DEFAULT permite definir um valor padrão para um campo, que será utilizado caso não seja informado nenhum valor para esse campo na inserção de um registro na tabela.

Sintaxe:

sexo char(1) **default 'M'**

No exemplo acima, caso o campo “sexo” da tabela não seja preenchido com um valor durante a inserção de um registro, será assumido o valor ‘M’ para o campo. Para campos do tipo NUMÉRICO, o valor DEFAULT é escrito sem aspas. Exemplo:

salario decimal(10,2) **default 0**,

Exemplo:

```
CREATE TABLE clientes(
    cpf integer unsigned not null,
    nome varchar(100) not null,
    data_nascimento date not null,
    sexo char(1) default 'M',
    salario decimal(10,2) default 0,
    profissao varchar(30),
    primary key(cpf)
);
```

Comando Drop Table

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

Sintaxe:

```
DROP TABLE < nome_tabela > ;
```

Exemplo:

Drop table alunos;

Comando Insert

Este comando insere os dados em tabelas.

Sintaxe:

INSERT INTO < nome_tabela > (<campos_data_tabela>) VALUES(<valores_campos>);

Exemplo:

insert into tbl_alunos (ra, nome, data_nasc, turma) values(1, 'MARCOS COSTA', '1981-02-06', '1DSN');

Comando Select

Consulta os dados em tabelas.

Sintaxe:

SELECT * FROM < nome_tabela > WHERE <condição>;

Exemplo:

SELECT * FROM tbl_alunos WHERE ra = 1;

Comando Update

Atualiza os dados em tabelas.

Sintaxe:

UPDATE < nome_tabela > SET <campos> WHERE <condição>;

Exemplo:

UPADATE tbl_alunos SET nome = “ANTONIO JOSÉ DA SILVA” WHERE ra = 1;

Comando Delete

Deleta os dados em tabelas (CUIDADO).

Sintaxe:

DELETE < nome_tabela > WHERE <condição>;

Exemplo:

DELETE FROM tbl_alunos WHERE ra = 1;

Comando Alter

Este comando permite inserir/eliminar atributos nas tabelas já existentes.

Sintaxe:

ALTER TABLE < nome_tabela > ADD / DROP (
nome_atributo1 < tipo > [NOT NULL],
nome_atributoN < tipo > [NOT NULL]
);

Exemplos:

Alter table alunos ADD COLUMN turno char(10) NOT NULL;

Alter table alunos DROP COLUMN turno;

Exercícios para fixação de conteúdo, em conjunto com o professor

De acordo com o Layout de Tela, criaremos o modelo conceitual com o brModelo e finalmente o banco físico no MySQL Workbench e definindo as tabelas, chaves primárias e estrangeiras compreendendo os dados apresentados:

Cadastro de produtos

Código do produto:

Descrição:

Categoria de produto:

Estoque mínimo:

Estoque máximo:

Tipo de produto:

Unid. Medida:

ComboBox

☒ Montadora
☒ Reposição
☒ Exportação

ComboBox

Atenção:

Categoria de produtos:

* Forjado

* Usinado

* Fundido

Unidade de medida:

* Pç

* Kg

* Cj

* Lt

Resolução:

1ª parte, vamos resolver com a criação de uma tabela, vemos abaixo:

```

1  #criar banco de dados empresa
2 • create database empresa;
3
4  #selecionar o banco de dados
5 • use empresa;
6
7  #criar a tabela de produtos
8 • create table produto(
9      codigo int primary key,
10     descricao varchar(30),
11     categoria varchar(10),
12     est_minimo int,
13     est_maximo int,
14     tp_montadora int,
15     tp_reposicao int,
16     tp_exportacao int,
17     unid_medida char(2)
18 );

```

```

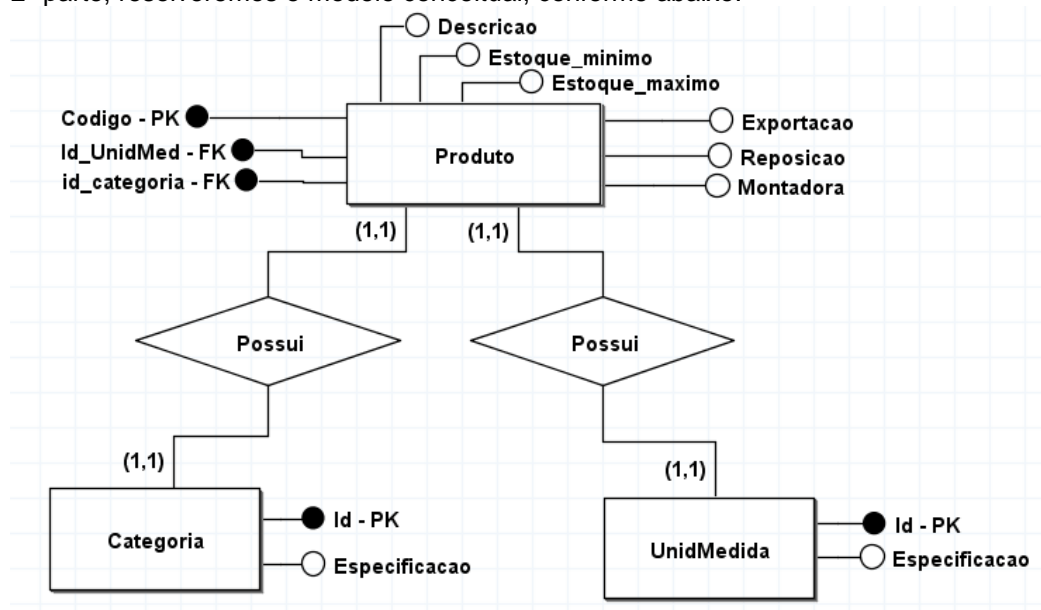
20  #inserir dados
21 • insert into produto (codigo, descricao, categoria,
22                        est_minimo, est_maximo,
23                        tp_montadora, tp_reposicao,
24                        tp_exportacao, unid_medida)
25  values(1, 'Parafuso', 'Forjado', 10, 100, 1, 1, 0, 'Pç');
26
27 • insert into produto (codigo, descricao, categoria,
28                        est_minimo, est_maximo,
29                        tp_montadora, tp_reposicao,
30                        tp_exportacao, unid_medida)
31  values(2, 'Prego', 'Uzinado', 10, 100, 1, 1, 0, 'Peça');
32
33 • select * from produto;

```

Teremos o seguinte resultado, lembrando que no segundo insert, colocamos dados errados, como escrita da categoria e unidade de medida:

	codigo	descricao	categoria	est_minimo	est_maximo	tp_montadora	tp_reposicao	tp_exportacao	unid_medida
1	1	Parafuso	Forjado	10	100	1	1	0	Pç
2	2	Prego	Uzinado	10	100	1	1	0	Pe
*									

2ª parte, resolveremos o modelo conceitual, conforme abaixo:



E codificaremos o banco de dados da seguinte forma:

```

1  #criar a nova base
2 • create database empresa2;
3
4  #selecionar a base de dados
5 • use empresa2;

```

```

7 • create table categoria(
8     id int primary key,
9     descricao varchar(15)
10 );
11
12 • create table unidmedida(
13     id int primary key,
14     descricao char(02)
15 );
16
17 #criar a tabela de produtos
18 • create table produto(
19     codigo int primary key,
20     descricao varchar(30),
21     idcategoria int,
22     est_minimo int,
23     est_maximo int,
24     tp_montadora int,
25     tp_reposicao int,
26     tp_exportacao int,
27     unid_medida int,
28     foreign key (idcategoria) references categoria(id),
29     foreign key (unid_medida) references unidmedida(id)
30 );
31
32 #INSERT DAS TABELAS
33 #PRIMEIRAMENTE, PRECISAMOS INSERIR OS DADOS NAS TABELAS QUE NÃO
34 #POSSUEM CHAVES ESTRANGEIRAS
35 • insert into categoria (id, descricao) values (1, 'Forjado');
36 • insert into categoria (id, descricao) values (2, 'Usinado');
37 • insert into categoria (id, descricao) values (3, 'Fundido');
38
39 • select * from categoria;
40
41 • insert into unidmedida (id, descricao) values (1, 'Pç');
42 • insert into unidmedida (id, descricao) values (2, 'Kg');
43 • insert into unidmedida (id, descricao) values (3, 'Cj');
44 • insert into unidmedida (id, descricao) values (4, 'Lt');
45
46 • select * from categoria;
47

```



```

48 #APÓS AS TABELAS QUE NÃO POSSUEM CHAVE ESTRANGEIRAS PREECHIDAS
49 #FAZEMOS A INSERÇÃO DAS TABELAS QUE POSSUEM DEPENDÊNCIAS
50 • insert into produto(codigo, descricao, idcategoria, est_minimo,
51                       est_maximo, tp_montadora, tp_reposicao,
52                       tp_exportacao, unid_medida)
53 values(1, 'Parafuso', 1, 10, 100, 1, 0, 0, 1);
54
55 • select * from produto;

```

Exercícios de avaliação, em duplas realizem os dois exercícios usando os conceitos vistos em aula, a atividade deverá ser realizada e entregue até dia 13/12.

1-) De acordo com o Layout de Tela, crie uma tabela e defina a chave primária compreendendo os dados apresentados e realizem os *inserts* necessários:

Cadastro de Alunos

Matrícula Data de Nascimento RG

Nome completo

Endereço

Número Bairro

Cidade UF

2-) De acordo com o Layout de Tela, criem as tabelas e definam as chaves primárias e estrangeiras compreendendo os dados apresentados e realizem os *inserts* necessários:

Cadastro de Clientes PESSOAL JURÍDICA

CNPJ: Inscrição estadual: CNAE: ComboBox

Endereço: Número:

Complemento: Bairro:

Cidade: CEP: UF: ComboBox

Telefone: Responsável imediato:

☒ Matriz ☐ Filial

Áreas de interesse:

☒ Vendas ☒ Recrutamento e seleção ☒ Fornecimentos ☒ Promoções

☒ News Letter ☒ Novos produtos ☒ Garantia ☒ Eventos