Anotações de Banco de Dados I

MySQL, Oracle, Postgre, SQLServer, MariaDB, MongDB e etc... (Tipos de banco de dados relacionais)

**SGBD:** Sistema Gerenciador de Banco de Dados;

**DBA:** DataBase Administrator;

**SQL:** Structured Query Language;

-> DML: Data Manipulation Language;

-> DDL: Data Definition Language;

-> DQL: Data Query Language;

Site para baixar o SQL --> https://www.mysql.com/

Arraste até o final e baixe o Server

**Hostname:** É o IP que a máquina interpreta como padrão, logo ela vai interpretar o local dela como ela mesma caso seja o ip 127.0.0.1,

Imaginando que você está no WhatsApp, é como se você mandasse mensagem para sí mesmo.

DDL = Data Definition Language

DML = Data Manipulation Language

DQL = Data Query Language

ERP = significa Enterprise Resource Planning, ou seja, Planejamento dos Recursos da Empresa. É um software que automatiza processos,

armazena dados e unifica a visualização de resultados em uma única plataforma.

Utilizando o MYSQL

Hostname: é o ip da máquina mesmo pois será enviado para elas os dados, sendo o ip 127.0.0.1;

Porta padrão do MySQL: 3306;

Engenharia: Primeiro desenhar e depois construir;

Engenharia Reversa: Pega a construção e transforma no desenho;

Modelagem de dados

As técnicas de modelagem são usadas para registrar os dados nos mais diferentes níveis de abordagem. O objetivo da modelagem é dar fidelidade à representação das "coisas do mundo" por meio do computador. Os dados podem mostrar toda sua elasticidade de assumir várias formas, dependendo do contexto que os envolvem.

Modelo de dados é a representação gráfica e textual das estruturas, operadores e regras que definem os dados. Cada entidade deve ser vista como um conjunto que pode ou não ter relacionamento (associação) com outro conjunto. Um BD relacional consiste em uma coleção de objetivos básicos chamados tabelas e seus relacionamentos.

Tabela: Conjunto único de atributos ou características, chamadas colunas.

Relacionamento: Associação entre 2 ou mais tabelas.

Diagrama E/R (Entidade/Relacionamento)

Ferramenta para modelagem conceitual de BD amplamente utilizada em projetos, considerada praticamente um padrão de modelagem.

Entidades: Componentes físicos e abstratos do mundo real, sobre os quais são armazenados os dados. Ex: Cliente, conta, banco, filme, etc.

Atributo: Propriedade da entidade. Não tem vida própria. Existe para caracterizar uma entidade. Ex: Nome, valor, data nascimento, etc.

Ocorrência: Conjunto de atributos de uma determinada entidade. Uma linha dentro da entidade. Ex: José da Silva, 25-02-1980 dentro da entidade cliente.

Relacionamento: Correspondência entre entidades. Associação entre 2 ou mais conjuntos de dados. Ex: Uma conta está sempre associada a pelo menos 1 cliente.

Identificador ou atributo determinante: Um atributo ou coleção de atributos que determina de modo único uma ocorrência de entidade. Ex: Código de matrícula, CPF, CPNJ.

Atividade 1: Com base no exemplo desenvolvido no lab para o banco de dados PLANTS. Crie uma estrutura lógica de Banco de dados para uma imobiliária para armazenar os registros de seus imóveis.

Para isto crie uma tabela IMÓVEIS com as seguintes colunas:

- ID (int)

- Name (string)

- Description (string)

- Num de Quartos (int)

- Value (real/float)

Insira ao menos 10 registros de dados para esta entidade e aplique ao menos 3 tipos diferentes de consulta para exibição dos dados.

Exemplo do código do projeto 1 feito com o professor:

-- CRIAR BD

CREATE DATABASE GREENHOUSE;

-- DEFINIR BD COMO PADRÃO / UTILIZÁVEL INDICA O DATABASE QUE ELE VAI FAZER A PRÓXIMA ETAPA (NO MEU CASO, CRIAR UMA TABELA)

USE GREENHOUSE;

-- CRIAR TABELA/ENTIDADE

CREATE TABLE `PLANTS` ( -- Para nomes dentro do banco de dados utiliza crase

`PLANT\_NAME` CHAR(30) NOT NULL,

`SENSOR\_VALUE` FLOAT DEFAULT NULL,

`SENSOR\_EVENT` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

PRIMARY KEY `PK\_PLANTS` (`PLANT\_NAME`)

);

-- VISUALIZAR TODAS AS OCORRÊNCIAS EM PLANTS

-- LISTAR TODOS OS REGISTROS DE DADOS

SELECT \* FROM PLANTS;

-- VISUALIZAR COLUNAS ESPECÍFICAS

SELECT PLANT\_NAME, SENSOR\_VALUE, SENSOR\_EVENT

FROM PLANTS;

-- INSERIR DADOS NA TABELA/ENTIDADE PLANTS

INSERT INTO PLANTS (PLANT\_NAME, SENSOR\_VALUE)

VALUES ('Rosa', 0.2319); -- Aspas simples para string

-- INSERIR MÚLTIPLAS LINHAS DE UMA VEZ

INSERT INTO PLANTS (PLANT\_NAME, SENSOR\_VALUE)

VALUES ('Cactus', '0.2411'),

('Girassol', '0.3112'),

('Orquídea', 0.4102),

('Lírio', 0.5566);

-- CONSULTA APLICANDO FILTROS

SELECT \*

FROM PLANTS

WHERE PLANT\_NAME = 'Cactus';

-- FILTROS COMPOSTOS COM OPERADORES AND OR XOR ETC

SELECT \*

FROM PLANTS

WHERE PLANT\_NAME <> 'CANNABIS' -- <>: É o símbolo de diferente

AND SENSOR\_VALUE < 0.5566

AND SENSOR\_VALUE > 0.2411;

**VARCHAR:** Utiliza apenas o espaço utilizado;

**CHAR:** Utiliza o total estabelecido por mais que não utilize.

Cardinalidade: 1 para 1 (1:1), 1 para vários (1:N), e vários para vários(N:N) essa última é uma anomalia.

Estabelecer uma relação de chave primária e chave estrangeira serve para indicar uma relação, onde não permite criar um item sem ter uma informação na tabela pai, caso eu exclua uma item ele apaga de ambas as tabelas.

-- DEFININDO UMA FK APÓS TER CRIADO A TABELA

ALTER TABLE TRANSACAO

ADD FOREIGN KEY (ID\_CONTA)

REFERENCES CONTA(ID);

-- CRIANDO INDEX (Serve para criar um tipo de filtro para pesquisar dps)

CREATE INDEX IDX\_TRANSACAO\_TP\_TRANSACAO

ON TRANSACAO (TIPO\_TRANSACAO);

ATIVIDADE - INDIVIDUAL

Com base no exemplo executado em laboratório para o Banco de Dados UNOESC BANK, com relacionamentos de cardinalidade 1:N monte um script de BD para o seguinte problema:

Uma empresa veterinária deseja criar um simples banco de dados para armazenar os registros dos TIPOS\_ANIMAIS que atende, podendo ser CANINOS, FELINOS, SUÍNOS, CAPRINOS E EQUINOS, ETC onde cada TIPO\_ANIMAL poderá ter vários ANIMAIS, para cada animal, você deve armazenar (ID, NOME, ID\_TIPO\_ANIMAL, DT\_NASCIMENTO, COR, PESO, ALTURA).

Além disso, o BD deve armazenar um histórico de vacinação, onde ANIMAL pode ter 'N' VACINAS. Para a tabela VACINA, armazene (ID, NOME, DATA\_APLICACAO, ID\_ANIMAL [FK])

--------------------------------------------------------------

CREATE TABLE `ITEM\_NOTA\_FISCAL`(

`NRO\_NOTA` INT NOT NULL,

`COD\_PRODUTO` INT NOT NULL,

`QTD\_PRODUTO` INT NOT NULL,

`VL\_PRECO` FLOAT NOT NULL,

PRIMARY KEY(NRO\_NOTA, COD\_PRODUTO), -- Uma única primary key porém ela é composta, foi utilizado aqui pois não se pode repetir o mesmo produto e o nro da nota, em vez de adicionar duas linhas de codigo 1, numero da nota 2, e qtde 1, deve-se colocar uma linha com o codigo 1, numero da nota 2 e a qtde 2.

CONSTRAINT FK\_NRO\_NOTA\_NOTA\_FISCAL

FOREIGN KEY (NRO\_NOTA)

REFERENCES NOTA\_FISCAL (NRO\_NOTA),

CONSTRAINT FK\_COD\_PRODUTO\_PRODUTO

FOREIGN KEY (COD\_PRODUTO)

REFERENCES PRODUTO(COD\_PRODUTO)

);

FAZENDO UPDATE COM WHERE NA TABELA PRODUTOS

SELECT \* FROM `PRODUTO` WHERE COD\_PRODUTO = 3;

UPDATE PRODUTO

SET VL\_PRODUTO = 45

WHERE COD\_PRODUTO = 3;

DELETANDO REGISTROS COM O DELETE

DELETE FROM PRODUTO

WHERE COD\_PRODUTO = 3;

JOIN = SERVE PARA PERSONALIZAR MEU SELECT (FUNÇÃO DE RELATÓRIO)

LEFT JOIN: TABELA A e TABELA B = DADOS QUE ESTÃO NA TABELA "A" E QUE ESTÃO NA "A" E NA "B";

INNER JOIN: TABELA A e TABELA B = DADOS QUE ESTÃO APENAS NAS TABELAS "A" E "B";

RIGHT JOIN: DADOS QUE ESTÃO NA TABELA "B" E QUE ESTÃO NA "A" E NA "B";

FULL OUTER JOIN = RECEBE TUDO DE AMBAS TABELA "A" E "B"

CTRL + ENTER: DEPOIS DE SELECIONAR OS COMANDO PARA EXECUTAR;

ATIVIDADE

COM BASE NO BANCO DE DADOS DE VETERINÁRIA, ELABORE CONSULTAS SQL PARA RESPONDER AS SEGUINTES QUESTÕES:

A) QUAL A QUANTIDADE DE ANIMAIS REGISTRADAS NO SISTEMA?

B) QUAL O TOTAL DE VACINAS APLICADAS REGISTRADAS NO SISTEMA?

C) QUANTOS ANIMAIS TEMOS AGRUPADOS REGISTRADOS PARA CADA CATEGORIA?

D) QUAL A CATEGORIA DE ANIMAIS QUE MAIS VACINAS RECEBEU?

ATV SEMINÁRIO

Com base no enunciado a seguir, elabore o diagrama de entidade e relacionamento do script de criação do banco de dados, e um script de semeadura de dados (Folha de código).

Um simpósio será organizado na universidade.

\A organização é de responsabilidade de uma ou mais pessoas.

Serão apresentados nesse simpósio vários minicursos, e cada um deles será ministrado por apenas uma pessoa.

Também serão aceitos artigos, sendo que cada um deles possuirá um tema (Banco de dados, redes, compiladores, realidade virtual e etc).

Estes artigos são escritos por no mínimo uma pessoa e não possuem limite de número de autores.

Cada artigo será encaminhado para uma comissão científica, sendo que há uma comissão científica para cada tema, que por sua vez é formada por uma ou mais pessoas, que emitirão um parecer sobre o artigo.

As pessoas se inscreverão no simpósio, podendo escolher se querem assistir os minicursos (E quais) e/ou palestras (E quais).

Precisamos entregar:

Diagrama de entidade e relacionamento;

Script de semeadura de dados;

Normalização

Serve para analisar tabelas e organizá-las de modo que sua estrutura seja simples, relacional e estável, para que o gerenciamento possa ser eficiente e seguro. Os objetivos são evitar a perda e a repetição de informação e atingir uma forma de representação adequada para o que se deseja armazenar.

Técnica empregada para melhorar as estruturas de dados resultantes de um projeto de dados do sistema. Evita problemas de redundância e anomalia através da decomposição de uma entidade ou mais entidades.

As formas normais (FN) são as regras de normalização e podem ocasionar uma proliferação de entidades que nem sempre é o ideal do ponto de vista de performance, devendo ser balanceadas as vantagens e desvantagens antes da efetivação dos resultados de uma FN (NF). Assim, a normalização é um processo sistemático de geração de tabelas podendo ser aplicadas regras que vão da primeira até a quinta FN. Em grande parte dos casos, a aplicação da terceira forma normal já produz um conjunto de tabelas que organiza satisfatoriamente um schema.

Para adequar uma tabela à uma FN deve-se redesenhar seu formato. A normalização também pode ser utilizada para projetar tabelas a partir de documentos, considerando-os como uma única tabela e aplicando-os as regras.

|  |
| --- |
| Exemplo: Nota Fiscal |
| nr. nota, |
| nm. cliente, |
| endereço, |
| vendedor, |
| data de emissão, |
| produto 1, |
| produto 2, |
| produto n. |

A tabela acima está desnormalizada, pois os campos, produto 1, produto 2 e o produto n não devem estar ali naquele local, pois a tabela continuará a crescer verticalmente, o que não é o correto.

**1ª FN: Exige que uma tabela não contenha alinhamentos (atributos repetitivos).**

Realiza o que se chama de nivelamento, gerando uma tabela com todos os dados coletados em atributos atômicos.

A PK desta tabela será formada pela composição da PK de cada tabela aninhada, além da PK da tabela externa.

Caso existam atributos multivalorados, estes devem se tornar componentes da chave primária uma hora que seus valores estarão associados várias vezes com os demais dados da tabela.

Se existia aninhamento, provavelmente não é o estado final, uma vez que haverá redundância nos dados da tabela externa.

Uma tabela só está na primeira FN se nenhum dos seus atributos possui domínio multivalorado, ou seja, devem ser eliminados registros de dados repetitivos.

Na nota fiscal, vemos que os produtos são atributos repetidos.

|  |
| --- |
| Nota.fiscal |
| #nr.nota |
| Nm.cliente |
| Endereço |
| Nm.vendedor |
| Dt.emissão |
| Vl.total.nota |

1

Para

N

|  |
| --- |
| Item.nota.fiscal |
| Nr.nota |
| Cod.produto |
| Nm.produto |

**2ª FN:**

Exige o entendimento do conceito de dependência funcional (DF).

Existe DF em uma tabela ou mais atributos determina o valor de outro conjunto de um ou mais atributos.

A regra diz que uma tabela está na ZFN se e somente se estiver na 1FN e todo atributo não-chave depende funcionalmente de toda a chave primária e não apenas de parte dela (Deve existir Dependência Funcional, ou seja, DF de toda a PK).  
Nesse caso, cada atributo não-chave de uma tabela deve ser analisado. Caso seja verificado DF de parte da PK, gera-se uma tabela que tenha esse atributo não-chave e essa parte da PK (Que se torna agora PK completa).

Se existirem outros atributos não-chave com DF dessa mesma parte da PK, esses serão inseridos nesta nova tabela.

O resultado é um conjunto de tabelas onde só existem DFs da PK completa.

Obviamente, tabelas com PK simples ou que não possuem atributos não chave já estão na 2FN.

Mesmos após a aplicação da 2FN podem ainda existir redundâncias e anomalias.

Aplicação:

A projeção do atributo determinante de produtos, mais os outros atributos determinantes de produtos, mais ou outros atributos que dependem apenas dele em outra tabela. A PK será o atributo determinante projetado.

|  |
| --- |
| **Nota.fiscal** |
| #nr.nota |
| Id.cliente |
| Endereço |
| Id.vendedor |
| Dt.emissao |
| Vl.total |

|  |
| --- |
| **Vendedor** |
| Id.vendedor |
| nome |

|  |
| --- |
| **Item.nota** |
| #nr.nota |
| #cod.produto |
| Qt.produto |
| Vl.unit |
| Vl.total |

|  |
| --- |
| **Produto** |
| cod.produto |
| descricao |
| Un.medida |
| Vl.unit |

**3ª FN:**

A regra é clara, uma tabela está na 3FN se e somente se todo atributo não chave depende funcional e diretamente da PK, e não de um conjunto de um ou mais atributos não-chave. Obviamente, apenas tabelas com mais de um atributo não chave são consideradas, para efeito de análise na 3FN.

Caso seja percebida uma DF (Dependência Funcional) indireta, uma nova tabela é gerada, onde os atributos não-chave que determinam outros tornam-se PK e os atributos com DF indireta migram para ela como atributos não-chave. Na tabela antiga os atributos determinantes tornam-se chaves estrangeiras.

A tabela estará na 3FN quando está na 2FN e não há DF transitiva entre seus atributos.

Dependência Funcional Transitiva: Significa que um atributo não depende diretamente do atributo determinante e sim de algum outro atributo, que por sua vez depende do seu determinante.

Resumo sobre cada FN até a 3FN:

 **1ª Forma Normal (1FN):** Elimina grupos repetitivos, garantindo que cada campo contenha apenas um valor por registro.

 **2ª Forma Normal (2FN):** Elimina dependências parciais, exigindo que todos os atributos dependam da chave primária completa.

 **3ª Forma Normal (3FN):** Elimina dependências transitivas, garantindo que atributos não-chave dependam apenas da chave primária.

**4ª FN:**

Ocorre quando há tabelas que mantém relacionamentos ternários ou superiores e são detectadas dependências funcionais multivaloradas, ou seja, um ou mais atributos determinam vários valores de um outro atributo. São consideradas, para efeito de análise da 4FN, apenas tabelas com chave primária tripla PK (quádrupla e assim por diante) que não tenham atributos não chave.

Normalmente, detecta-se a necessidade da aplicação da 4FN quando se tem mais de um atributo multivalorado em uma tabela não normalizada. Esses atributos acabam fazendo parte da chave primária nessa tabela na 1FN, porém, isolados, eles não determinam nenhum atributo não chave. Isto resulta em uma tabela com chave primária tripla (no mínimo), onde um deles é o identificador da tabela propriamente dita.

Uma tabela está na 4FN se está na 3FN e não apresenta mais um fator multivalor em relação a uma entidade descrita pela tabela.

O objetivo da 4FN é evitar redundância em situações em que há fatos multivalorados em vez de dependências funcionais.

Para isso, constroem-se tantas tabelas quantos fatos multivalorados há para certa entidade descrita na tabela. Esta entidade estará representada em todas as tabelas resultantes.

**5ª FN:**

Esta regra também se aplica apenas a tabelas que representam relacionamentos ternários, sem atributos não-chave.

É dito que uma tabela está na 5FN se um relacionamento triplo puder ser, decomposta em 3 tabelas de relacionamentos binários (PK é um par) sem que isso gere dados incorretos quando eles forem combinados novamente em uma tabela de relacionamento triplo.

Essa garantia de consistência exige que uma premissa seja sempre verdadeira:

Toda vez que um dado A se relaciona com outros dois (B e C) e estes outros também por sua vez (gerando 3 relações binárias; (A,B), (A,C), (B,C)), então é verdade que ocorre na realidade os 3 dados combinados (A,B,C). Um exemplo é pensar em A como sendo um agente de vendas, B como sendo a empresa fabricante e C um produto.

A premissa seria: Se A representa a empresa B, A é especialista na venda do produto C e B produz o produto C, então é verdade que A vende o produto C para a empresa B.

Essa forma normal elimina o relacionamento ternário e a redundância de pares de valores relacionados, como, por exemplo, a redundância da afirmação de que o agente A representa a empresa B.

Uma tabela está na 5FN se está na 4FN e seu conteúdo não pode ser reconstituído a partir de tabelas menores.

O seu objetivo é impedir que a decomposição de tabelas gere informação inconsistente na junção.

Verifique sempre se há ou não a necessidade de conceber ou não relacionamentos de grau maior que 2.

Em todo caso, dificilmente o projetista aplicará a normalização segundo a 5FN, por que qualquer tabela que não for 5FN.

Geralmente falha em uma das FN anteriores.

**Atividade:**

Um cliente entrou em contato para transformar um formulário físico em bancos de dados digitais.

O formulário a seguir:

**Dados cadastrais**

Matrícula: Nome:

Data nasci: Nacionalidade: Sexo:

Est. Civil: RG: CPF:

Endereço: Data Admissão:

Cargos Ocupados

Cargo Dt. Início Dt. Fim

Cargo Dt. Início Dt. Fim

Cargo Dt. Início Dt. Fim

Departamento de Lotação

Depto Dt. Início Dt. Fim

Depto Dt. Início Dt. Fim

Depto Dt. Início Dt. Fim

Dependentes

Nome: Dt. Nascimento

Nome: Dt. Nascimento

Nome: Dt. Nascimento

**Stored Procedures (Procedimentos Armazenados)**

Se trata, basicamente, de uma função ou um método utilizado no C#.

Tupla: É uma linha com os dados das colunas.

**Triggers (Gatilhos)**