TRYBE
Modulo III – Back-end

Bloco 22 – SQL - Modelagem

1) Transformando ideias em um modelo de banco de dados

Abilidade de transformar ideias, problemas ou situações em um banco de dados, importante para profissões como desenvolvimento de software, DBA ou database administrator e Data scientist.

Database Design

4 etapas:

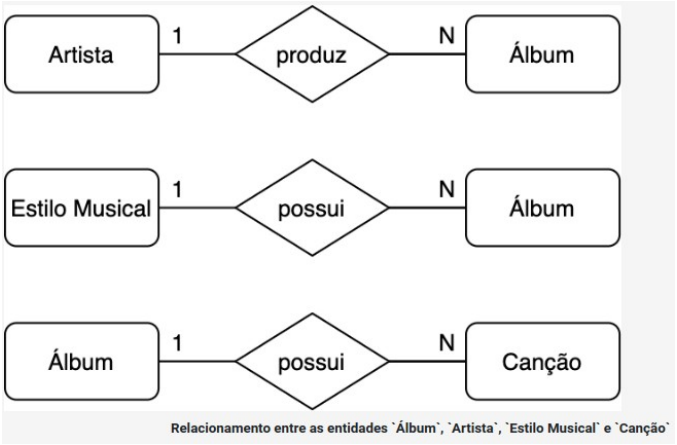
1. Identificar as entidades, atributos e relacionamentos com base na descrição do problema;

Entidade: objetos a serem manipulados
Atributos: propriedades destas entidades
Relacionamentos: 1..1, 1..N, N..1 e N..N (via tabela de junção).

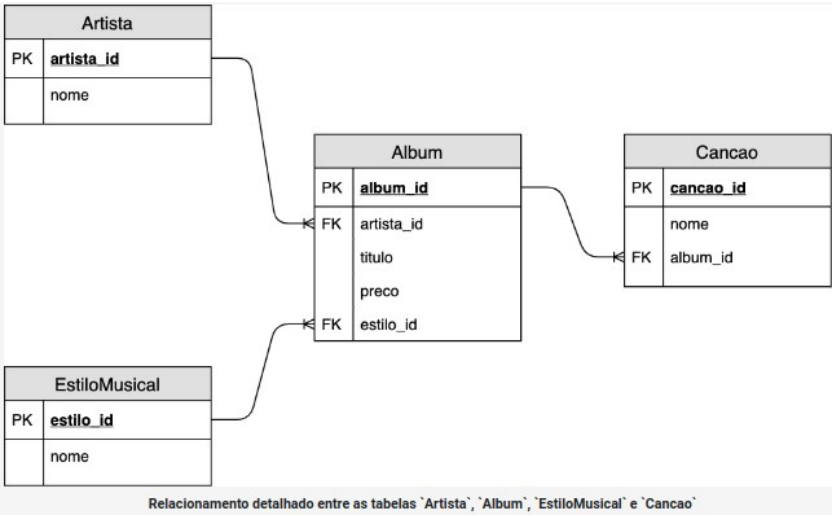
2. Construir um diagrama entidade-relacionamento

Representação gráfica para a estrutura de seu banco de dados.

A/ Modelo EntidadeA + verbo + EntidadeB || B/ Diagrama mais detalhado



Relacionamento entre as entidades `Álbum`, `Artista`, `Estilo Musical` e `Canção`



Relacionamento detalhado entre as tabelas `Artista`, `Album`, `EstiloMusical` e `Cancao`

3. Criar um banco de dados para conter suas tabelas;

-- Cria um banco de dados com o nome especificado.
CREATE DATABASE nome_do_banco_de_dados;

-- Sinônimo de CREATE DATABASE, também cria um banco de dados.
CREATE SCHEMA nome_do_banco_de_dados;

-- Verifica se o banco de dados ainda não existe.
-- Essa verificação é comumente utilizada junto ao CREATE DATABASE para evitar
-- a tentativa de criar um banco de dados duplicado, o que ocasionaria um erro.
IF NOT EXISTS nome_do_banco_de_dados;

– Tudo acima combinado:
CREATE DATABASE/SCHEMA IF NOT EXISTS nome_do_banco_de_dados;

-- Lista todos os bancos de dados existentes.
SHOW DATABASES;

-- Define o banco de dados escolhido como ativo.
USE nome_do_banco_de_dados;

4. Criar e modelar tabelas tendo o diagrama do passo 2 como base.

Entender: *tipos de dados - diferentes chaves - como criar tabela*.

---

2) Normalização, Formas Normais e Dumps

Normalização

Técnica para organizar tabelas relacionadas no banco de dados, para **reduzir redundância** de dados e suas consequências: mais **espaço ocupado, anomalias de inserção, de atualização e de exclusão**.

Boa prática: separar tabelas para garantir integridade dos dados.

1ª Forma Normal

Para criar/obter tabelas escaláveis e extensíveis.

2ª Forma Normal

- A tabela deve estar na 1ª Forma Normal
- A tabela não deve possuir dependências parciais

Diferente da dependência funcional quase incontornável, a dependência parcial é quando uma **coluna não depende de todas as outras da tabela**:

nota_id	aluno_id	materia_id	nota	professor
1	10	1	58	Cristian
2	10	2	67	Betina
3	11	1	58	Cristian
4	11	2	69	Betina
5	11	3	95	Rodrigo

// professor depende apenas de materia_id

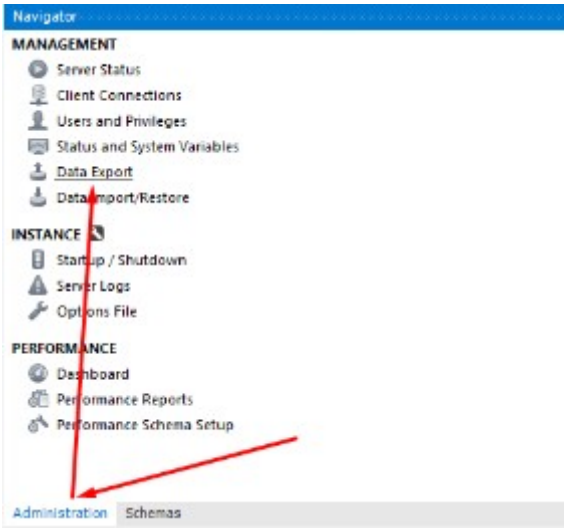
3ª Forma Normal

- A tabela deve estar na 1ª e 2ª Formas Normais;
- A tabela não deve conter atributos (colunas) que não sejam totalmente dependentes na *PK* (chave primária).

Dump

O que é: **back up** de um banco enquanto manipula e normaliza ele.

Como fazer:
1/ via **WorkBench** (Administration – Data Export – seguir escolhas e passos até Start Export – para restaurar, abrir o SQL script correspondendo com o dump e ativar ele do jeito clássico)



2/ via **linhas de comando -** seguir guia.

(form dia: conteudo próprio legal, apenas sugerir estilo melhor pra força da marca)

Dicas diversas

Para equilibrar entre a exigência dessas normas e o imperativo de evitar multiplicar demais o n° de tabelas, focar na funcionalidade: o que fizer sentido pelo usuário.

Todo fluxo de design de BD no Workbench aqui.
[Sincronizar diagram com tabela via ctrl shift Z ou database-synchronize model.
Se tiver problema ao modificar, deletar a key que deu problema e refazer resolve.]

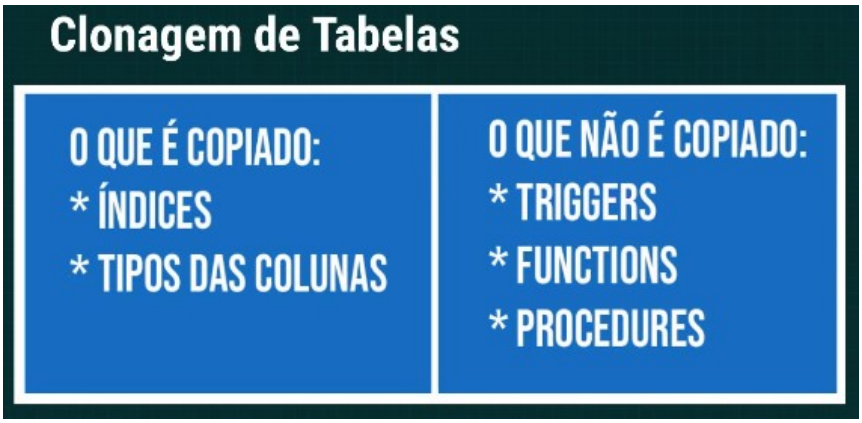
---

3) Transformando ideias em um modelo de banco de dados - Parte 2

Clonar tabelas existentes

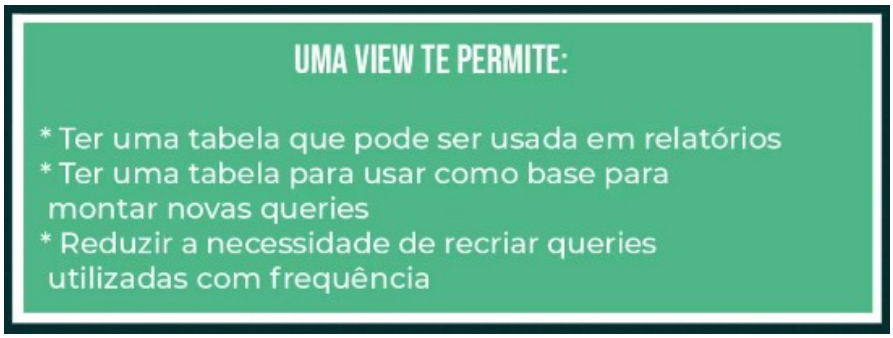
USE nome_do_banco_de_dados; -- Defina em qual banco o clone será criado
CREATE TABLE nome_para_nova_tabela LIKE tabela_a_ser_clonada;

Copia apenas a estrutura.



VIEWs

O que é: **tabela temporária** no seu banco de dados. Como se fosse apenas um **espelho**.



USE nome_do_banco_de_dados; -- Defina em qual banco a view será criada
CREATE VIEW nome_da_view AS query;

```
CREATE VIEW top_10_customers AS
    SELECT c.customer_id, c.first_name, SUM(p.amount) AS total_amount_spent
    FROM sakila.payment p
    INNER JOIN sakila.customer c ON p.customer_id = c.customer_id
    GROUP BY customer_id
    ORDER BY total_amount_spent DESC
    LIMIT 10;
```

// view é um alias para esse select

DROP VIEW nome_da_view; -- Para excluir uma VIEW

ALTER TABLE

As tabelas são alteradas com frequência. Operações mais comum:

```sql
-- Adicionar uma nova coluna
ALTER TABLE noticia ADD COLUMN data_postagem date NOT NULL;

-- Modificar o tipo e propriedades de uma coluna
ALTER TABLE noticia MODIFY noticia_id BIGINT;

-- Adicionar incremento automático a uma coluna
-- (especifique o tipo da coluna + auto_increment)
ALTER TABLE noticia MODIFY noticia_id BIGINT auto_increment;

-- Alterar o tipo e nome de uma coluna
ALTER TABLE noticia CHANGE historia conteudo_postagem VARCHAR(1000) NOT NULL;

-- Dropar/Excluir uma coluna
ALTER TABLE noticia DROP COLUMN data_postagem;

-- Adicionar uma nova coluna após outra
ALTER TABLE noticia ADD COLUMN data_postagem DATETIME NOT NULL AFTER titulo;
```

Para confirmar se a estrutura da sua tabela foi alterada corretamente:
SHOW COLUMNS FROM nome_da_tabela;

DROP tabela

DROP TABLE nome_da_tabela;

Impossível para tabela que é referenciada por uma restrição de chave estrangeira.

INDEX

Solução para problemas de **performance**. Maneira de estruturar dados de maneira mais eficiente, indicar onde está para consultas mais rápidas.

Tipos:
PRIMARY KEY - UNIQUE INDEX – INDEX - FULLTEXT INDEX

Quando criar: adicionar index a uma tabela, durante a criação da tabela, ou ao alterar uma tabela.

Quando usar:

```
Pontos Positivos:
* Acelera suas queries SELECT (resultados entregues mais rápidos)
* Permite tornar uma coluna única(quando usamos o UNIQUE INDEX ou UNIQUE)
* Permite entregar pesquisas que envolvem buscas em grandes blocos de texto mais rápidos(no caso do FULLTEXT INDEX)
* Aceleram as operações de UPDATE que usam WHERE

Pontos negativos:
* Índices ocupam espaço no disco
* Índices tornam as operações INSERT, UPDATE e DELETE mais lentas, sendo que cada índice precisa ser atualizado junto com os
dados.
```

Resumindo, é melhor não usar:
- Em tabelas *pequenas*;
- Em colunas que retornarão uma *grande quantidade* de dados quando filtradas;
- Em tabelas que frequentemente têm *atualizações* em grande escala;
- Em colunas que são *frequentemente manipuladas*;
- Em colunas que possuem *muitos valores nulos*.

Sintaxe

-- Criando um índice em uma coluna
CREATE [INDEX | FULLTEXT INDEX | UNIQUE INDEX] nome_indice
ON tabela (coluna);

-- Criando um índice composto, em duas ou mais colunas
CREATE [INDEX | FULLTEXT INDEX | UNIQUE INDEX] nome_indice
ON tabela (coluna1, coluna2);

-- Excluindo índices
DROP INDEX nome_do_indice ON tabela;

Comparar performances
* INDEX via WHERE
* FULL TEXT INDEX, via WHERE, MATCH e AGAINST.
* UNIQUE
Lembrando que PRIMARY KEY por padrão tem restrições UNIQUE INDEX + NOT NULL

Reações dinâmicas com Triggers

O que é: blocos de código SQL que são **disparados em reação a alguma atividade no BD.**

Quando dispara: BEFORE – AFTER

O que dispara: INSERT – UPDATE – DELETE

O que acessa: OLD e NEW value dentro de uma coluna
Casos de disponibilidade:

Operação	OLD	NEW
INSERT	Não	Sim
UPDATE	Sim	Sim
DELETE	Sim	Não

Sintaxe

```
DELIMITER $$

CREATE TRIGGER nome_do_trigger
[BEFORE | AFTER] [INSERT | UPDATE | DELETE] ON tabela
FOR EACH ROW
BEGIN
    -- o código SQL entra aqui
END $$

DELIMITER ;
```

Listagem 1. Sintaxe básica de criação da trigger PL/SQL.

```
1  CREATE [OR REPLACE ] TRIGGER trigger_name
2  {BEFORE | AFTER | INSTEAD OF }
3  {INSERT [OR] | UPDATE [OR] | DELETE}
4  [OF col_name]
5  ON table_name
6  [REFERENCING OLD AS o NEW AS n]
7  [FOR EACH ROW]
8  WHEN (condition)
9  DECLARE
10 Declaration-statements
11 BEGIN
12 Executable-statements
13 EXCEPTION
14 Exception-handling-statements
15 END;
```

Info completa neste link

Dicas diversas

// A palavra-chave `NEW` é utilizada para acessar e modificar as propriedades da tabela

Clone vs Dump: Dump puxa todos procedures, triggers, functions, dados inseridos enquanto o Clone é apenas a estrutura de uma tabela especifica.

View vs Function ou Procedure: View é para fazer apenas uma consulta e não pode ter param. View substitui um Select complexo e repetido.