



UC8 - BANCO DE DADOS

OBJETIVO DO DIA



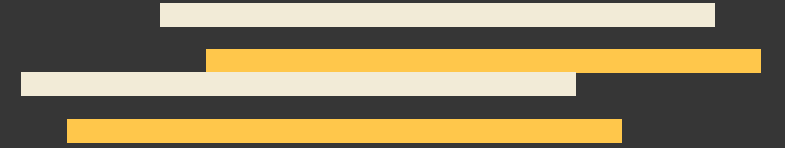
- Apresentar a UC8.
- Integrar a turma e verificar logins e acessos
- Explicar o que será aprendido.
- Mostrar exemplos de Bancos de Dados.
- Instalar e configurar ferramentas.

INTEGRAÇÃO E APRESENTAÇÃO



- Dinâmica rápida de apresentação dos alunos.

OBJETIVO DA NOSSA UC



Implementar banco de dados para web

O aluno será capaz de aplicar e manipular dados em um sistema gerenciador de banco de dados realizando a conexão da linguagem de programação orientada a objeto com segurança, versionamento e testes para troca e armazenamento de informações.

O QUE É BANCO DE DADOS



- Conjunto organizado de informações
- Permite armazenar, consultar e manipular dados

Exemplo prático:

- Agenda de contatos (nome, telefone, email)
- Cadastro de clientes (nome, CPF, endereço)

SGBD

(SISTEMA GERENCIADOR DE BANCO DE DADOS)

- Software que administra o banco de dados
- Funções: criar, atualizar, excluir e consultar dados com segurança

Exemplos de SGBD:

- MySQL (open source, muito usado em aplicações web)
- PostgreSQL
- SQL Server
- Oracle

SGBD (SISTEMA GERENCIADOR DE BANCO DE DADOS)



SQL VS NOSQL



SQL (relacional):

- Estrutura em tabelas (linhas e colunas)
- Schema fixo (tipos e restrições definidos)
- Usa SQL (Structured Query Language)
- Ex.: MySQL, PostgreSQL

Cód	Mercadoria	Qtda_Estoque	Fornecedor	Validade
189	Azeitonas Pretas	50	05	02/08/2018
222	Peixe Congelado	26	08	22/09/2019
236	Enlatado	48	04	15/11/2018

Chave Primária

Chave Estrangeira

Coluna com restrição NOT NULL

SQL VS NOSQL



Exemplo::

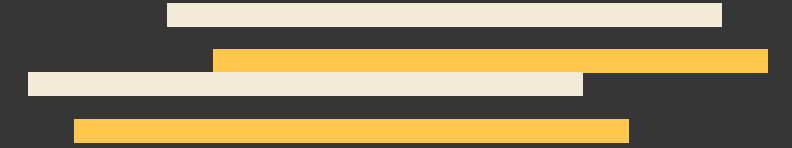
Documento JSON flexível:

```
{  
  "nome": "Maria",  
  "email": "maria@email.com",  
  "compras": ["livro", "notebook"]  
}
```

NoSQL (não relacional):

- Estruturas mais flexíveis (documentos, chave-valor, grafos)
- Sem schema fixo
- Ideal para grandes volumes de dados distribuídos
- Ex.: MongoDB, Cassandra

SQL VS NOSQL



GERENCIADORES DE BANCOS DE DADOS

RELACIONAIS	NÃO RELACIONAIS
<ul style="list-style-type: none">• MySQL• SQLite• PostgreSQL• SQL Server• Oracle• Microsoft Access	<ul style="list-style-type: none">• MongoDB• Redis• Azure DB• Cassandra• DynamoDB• CouchDB

MODELOS DE DADOS



- Conceitual: visão geral (Entidades e Relacionamentos)
- Lógico: traduz para tabelas (chaves primárias e estrangeiras)
- Físico: implementado no SGBD (tipos de dados, índices)

Exemplos de SGBD:

- Conceitual: Cliente — compra — Produto
- Lógico:
 - Tabela Cliente(id, nome, email)
 - Tabela Produto(id, nome, preço)
 - Tabela Compra(id, cliente_id, produto_id, data)
- Scripts criados

POR QUE APRENDER MYSQL COM XAMPP?



- MySQL é um dos bancos mais utilizados no mundo
- Gratuito, open source e rápido
- XAMPP traz:
 - Apache (servidor web)
 - MySQL
 - PHP
 - phpMyAdmin (interface gráfica web para gerenciar BD)



DO QUE VAMOS PRECISAR?

XAMPP



Passo a Passo de Instalação

- Baixar e instalar o XAMPP
- Iniciar módulos Apache e MySQL no painel do XAMPP
- Acessar phpMyAdmin no navegador:
 - <http://localhost/phpmyadmin>
- Criar primeira conexão:
 - Usuário padrão: root
 - Sem senha (por padrão no XAMPP)

EXPLORANDO O PHPMYADMIN



- Criar novo banco de dados:
 - Nome: loja
 - Charset: utf8_general_ci
- Explorar menus:
 - Estrutura → mostra tabelas
 - SQL → editor para comandos SQL
 - Exportar / Importar

Comando:

```
CREATE DATABASE loja;
```

CONTA NO GITHUB



útil para salvar, compartilhar e publicar os projetos.

Passo a Passo

- 1. Acessar o site: <https://github.com>
- 2. Clicar em Sign up
- 3. Preencher dados da conta
 - username: será o endereço do perfil, ex: github.com/seu-usuario
- 4. Verificação de segurança: captcha
- 5. Confirmar email
- 6. Escolher plano (free já atende o que vamos precisar)

CONTA NO GITHUB



continuação

Passo a Passo

- 7. Configuração inicial do perfil (pode pular essa parte se quiser)
- 8. Conta criada
 - ja tem acesso ao painel do GitHub
 - A URL do seu perfil será algo como:
 - <https://github.com/seu-usuario>



Dúvidas e Perguntas ?





Bom descanso.





Aula 2



TIPOS DE DADOS



Tipos de String (Texto)

Para armazenar dados textuais.

- CHAR: Armazena strings de tamanho fixo.
- VARCHAR: Para strings de tamanho variável, utilizando apenas o espaço necessário para o dado.
- TEXT: Para textos mais longos de comprimento variável, com um limite maior que o VARCHAR.

TIPOS DE DADOS



Tipos Numéricos

São usados para armazenar números.

- INT (Integer): Para números inteiros, positivos ou negativos.
- BIGINT: Para números inteiros maiores que o INT.
- DECIMAL ou NUMERIC: Para números decimais com precisão fixa, ideais para valores monetários, pois garantem exatidão.
- FLOAT e DOUBLE: Para números de ponto flutuante com precisão simples e dupla, respectivamente, que representam números com frações, mas podem não ser exatos.
- float - precisao simple 32 bites (4 bytes)
- double - precisao dupla 64 bites(8 bytes)

TIPOS DE DADOS



Tipos de Data e Hora

Para armazenar informações de tempo.

- DATE: Armazena datas no formato “AAAA-MM-DD”.
- TIME: Armazena horários no formato “HH:MM:SS”.
- DATETIME: Combina data e hora no formato “AAAA-MM-DD HH:MM:SS”.
- TIMESTAMP: Semelhante ao DATETIME, mas armazena a data e hora como o número de segundos passados desde uma data de referência, o que o torna útil para registrar o momento de criação ou atualização de um registro.
- YEAR: Para armazenar o ano com quatro dígitos no formato “AAAA”.

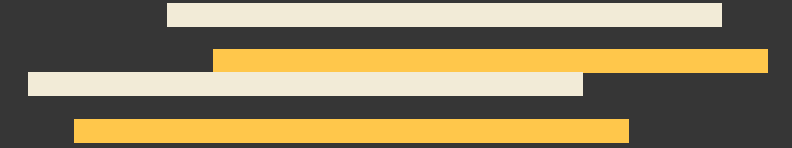
TIPOS DE DADOS



Outros Tipos

- ENUM: Permite definir uma lista de valores permitidos para uma coluna, funcionando como uma lista de opções restritas.
- Espacial: Para armazenar dados geométricos, como pontos em uma superfície geográfica.
- JSON: Para armazenar documentos em formato JSON (JavaScript Object Notation).

EXERCICIO 1



Requisitos

- Crie um banco de dados chamado escola.
- Selecione o banco de dados escola.
- Crie uma tabela chamada alunos com os seguintes campos:
 - id → inteiro, chave primária, auto-increment
 - nome → varchar(50)
 - idade → tinyint
 - curso → varchar(30)

SOLUÇÃO 1

-- 1. Criar banco de dados

```
CREATE DATABASE escola;
```

-- 2. Selecionar o banco de dados

```
USE escola;
```

-- 3. Criar tabela alunos

```
CREATE TABLE alunos (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nome VARCHAR(50),  
    idade TINYINT,  
    curso VARCHAR(30)  
);
```

-- Conferir estrutura da tabela

```
DESCRIBE alunos;
```



EXERCICIO 2



Requisitos

- Insira 5 alunos na tabela alunos com nomes, idades e cursos diferentes.
- Exiba todos os dados da tabela.
- Exiba apenas os nomes e idades dos alunos.
- Exiba apenas os alunos que têm idade maior que 18 anos.

SOLUÇÃO 2



-- 1. Inserir 5 alunos

```
INSERT INTO alunos (nome, idade, curso) VALUES  
( 'João', 20, 'Informática'),  
( 'Maria', 18, 'Administração'),  
( 'Pedro', 15, 'Matemática'),  
( 'Ana', 22, 'Engenharia'),  
( 'Lucas', 17, 'Biologia');
```

-- 2. Exibir todos os dados da tabela

```
SELECT * FROM alunos;
```

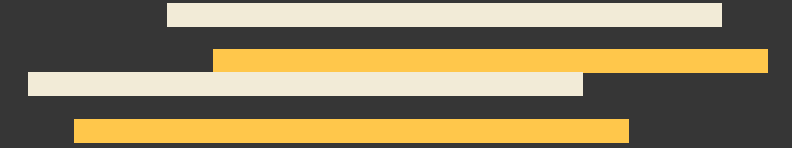
-- 3. Exibir apenas nomes e idades

```
SELECT nome, idade FROM alunos;
```

-- 4. Exibir alunos com idade maior que 18

```
SELECT * FROM alunos WHERE idade > 18;
```

EXERCICIO 3



Requisitos

- Atualize o curso de um aluno específico, por exemplo, mude o curso do aluno “João” para “Matemática”.
- Remova da tabela todos os alunos com idade menor que 16 anos.
- Exiba novamente todos os dados da tabela para verificar as alterações.

SOLUÇÃO 3



-- 1. Atualizar curso do aluno João para Matemática

```
UPDATE alunos
```

```
SET curso = 'Matemática'
```

```
WHERE nome = 'João';
```

-- 2. Remover alunos com idade menor que 16

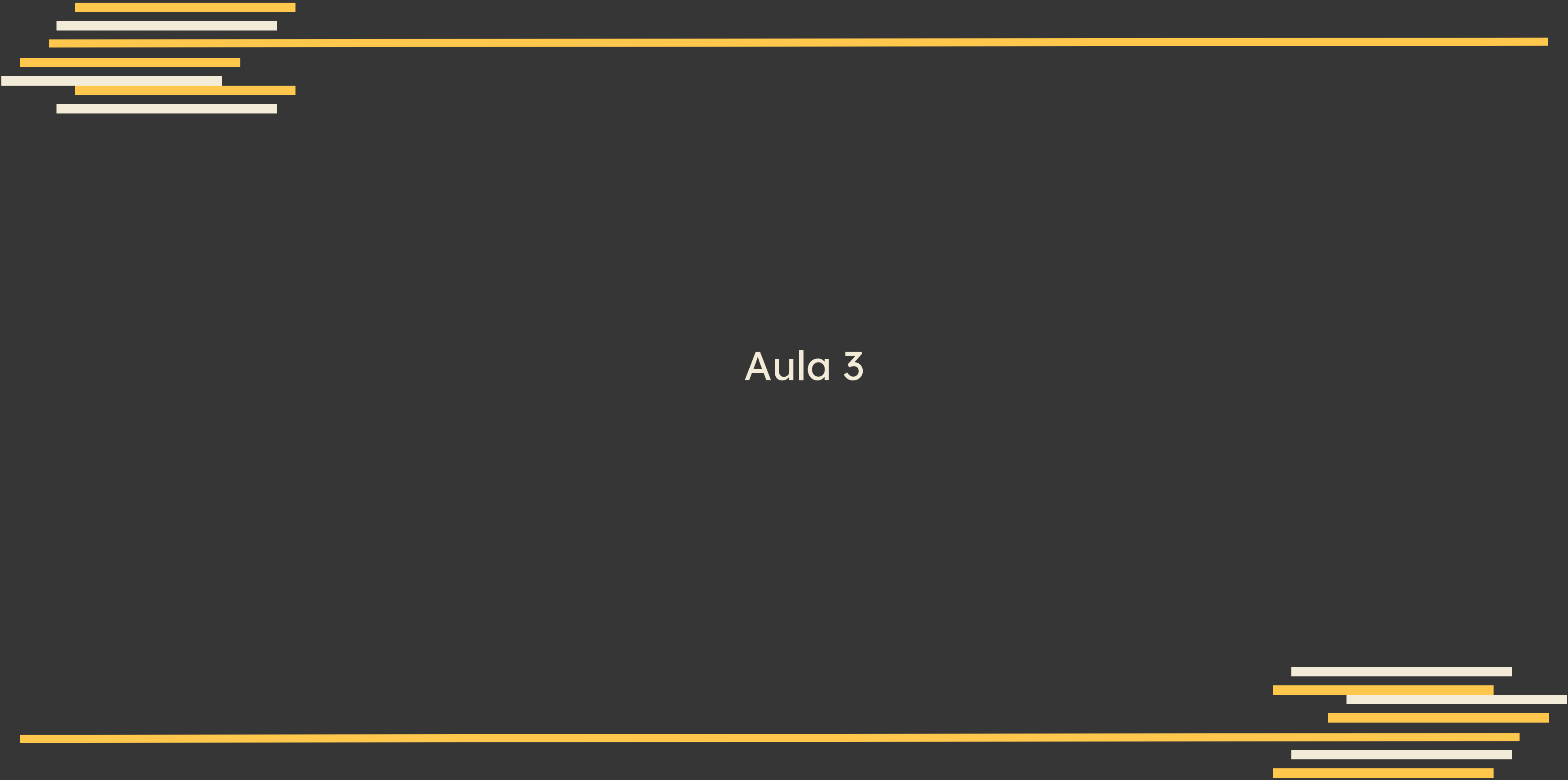
```
DELETE FROM alunos
```

```
WHERE idade < 16;
```

-- 3. Conferir alterações

```
SELECT * FROM alunos;
```

Aula 3



OBJETIVO DE HOJE



Chave Primária, Chave Estrangeira e Relacionamentos

REVISÃO

Já aprendemos:

- Criar tabelas
- Inserir dados
- Alterar estrutura
- Agora vamos aprender como relacionar tabelas.

O QUE É CHAVE PRIMÁRIA (PRIMARY KEY)?

- Identificador único de cada registro da tabela
- Não pode ser nulo
- Não pode se repetir
- Normalmente é um campo numérico com
AUTO_INCREMENT

EXEMPLO

sql

```
CREATE TABLE pessoas (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nome VARCHAR(30),  
    nascimento DATE  
);
```

O QUE É CHAVE ESTRANGEIRA (FOREIGN KEY)?

- Campo em uma tabela que referencia a chave primária de outra
- Garante a integridade referencial
- Define um relacionamento entre tabelas

EXEMPLO



sql

```
CREATE TABLE enderecos (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    rua VARCHAR(50),  
    cidade VARCHAR(30),  
    pessoa_id INT,  
    FOREIGN KEY (pessoa_id) REFERENCES pessoas(id)  
);
```

INTEGRIDADE REFERENCIAL



Banco de dados garante que:

- Não existam registros órfãos (endereços sem pessoa).
- Alterações em pessoas.id afetam registros relacionados.
- Pode ser configurada com:
 - ON DELETE CASCADE
 - ON UPDATE CASCADE

RELACIONAMENTOS



- 1:1 → uma pessoa tem um único passaporte
- 1:N → uma pessoa pode ter vários endereços
- N:M → alunos e cursos

EXEMPLO 1:

RELACIONAMENTO 1:N



sql

```
CREATE TABLE pedidos (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    data DATE,  
    pessoa_id INT,  
    FOREIGN KEY (pessoa_id) REFERENCES pessoas(id)  
);
```


EXEMPLO 2:

RELACIONAMENTO N:M

sql

```
CREATE TABLE cursos (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nome VARCHAR(50)  
);  
  
CREATE TABLE aluno_curso (  
    aluno_id INT,  
    curso_id INT,  
    PRIMARY KEY (aluno_id, curso_id),  
    FOREIGN KEY (aluno_id) REFERENCES pessoas(id),  
    FOREIGN KEY (curso_id) REFERENCES cursos(id)  
);
```



Aula 4



COMANDOS SQL



DDL (Definição de Dados)

DML (Manipulação de Dados)

DQL (Consulta de Dados)

DCL (Controle de Dados)

TCL (Transações)

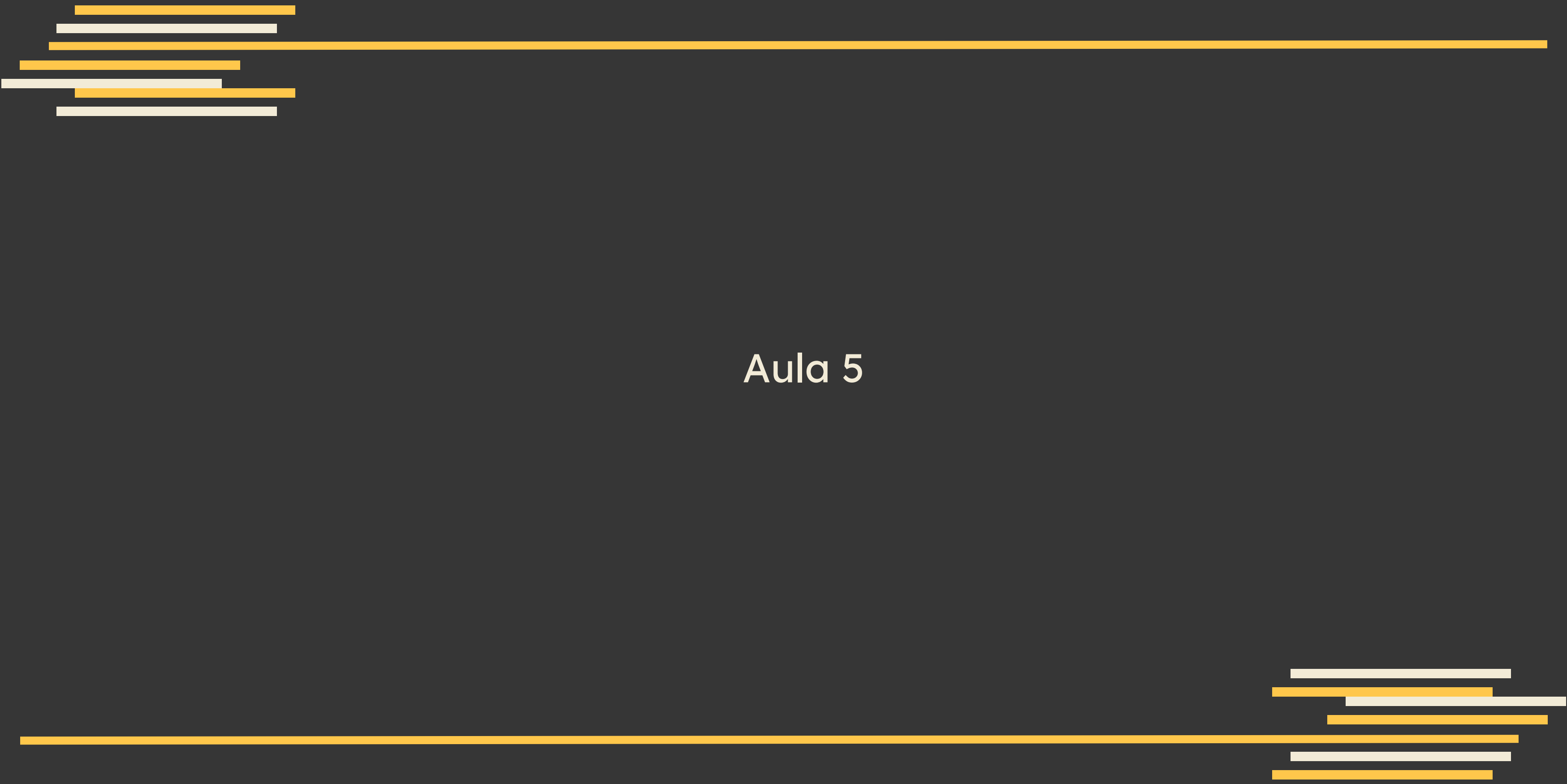
e

Consultas e Operações Avançadas

Transações e Integridade

Administração de Banco de Dados

Aula 5



COMANDOS SQL



DDL (Definição de Dados)

DML (Manipulação de Dados)

DQL (Consulta de Dados)

DCL (Controle de Dados)

TCL (Transações)

Consultas e Operações Avançadas

Transações e Integridade

Administração de Banco de Dados

COMANDOS SQL



São os comandos usados para criar e modificar a estrutura do banco de dados.

- Trabalham com tabelas, colunas, índices, views, esquemas.
- Quando executados, geralmente fazem mudanças permanentes na estrutura.

DDL (Definição de Dados)

CREATE - Cria bancos, tabelas, índices, views

ALTER - Altera estrutura de tabelas

DROP - Remove tabelas ou objetos

TRUNCATE - Limpa dados mantendo a tabela

RENAME - Renomeia tabelas/colunas

COMMENT - Adiciona comentários

CREATE INDEX - Cria índices

DROP INDEX - Remove índices

CREATE VIEW - Cria uma view

DROP VIEW - Remove uma view

COMANDOS SQL



São os comandos usados para manipular os dados dentro das tabelas.

- Eles inserem, atualizam ou excluem registros.
- Afetam apenas os dados, e não a estrutura do banco.

DML (Manipulação de Dados)

INSERT - Insere registros

UPDATE - Atualiza registros

DELETE - Remove registros

MERGE - Mescla dados (INSERT/UPDATE)

REPLACE - Substitui registros (MySQL)

UPSERT - Combina INSERT e UPDATE

COMANDOS SQL



- Basicamente, é onde usamos SELECT para buscar dados.
- Pode incluir filtros (WHERE), agrupamentos (GROUP BY), ordenações (ORDER BY) e junções (JOIN).

DQL (Consulta de Dados)

SELECT - Recupera dados

SELECT DISTINCT - Valores únicos

SELECT INTO - Cria tabela a partir de consulta

EXPLAIN / DESCRIBE - Estrutura/plano de execução

WITH - CTEs temporárias

COMANDOS SQL



São os comandos usados para controlar os acessos e permissões dos usuários no banco.

- Definem quem pode ver, alterar ou excluir informações.
- Muito usados na administração de segurança do banco.

DCL (Controle de Dados)

GRANT - Concede permissões

REVOKE - Remove permissões

DENY - Nega permissões (SQL Server)

AUDIT - Habilita auditoria (Oracle)

NOAUDIT - Desabilita auditoria

COMANDOS SQL



São os comandos usados para gerenciar transações, garantindo consistência dos dados.

- Uma transação é um conjunto de operações SQL que deve ser executado de forma atômica (tudo ou nada).
- Garantem as propriedades ACID:
Atomicidade, Consistência, Isolamento, Durabilidade.

TCL (Transações)

COMMIT - Confirma alterações

ROLLBACK - Desfaz alterações

SAVEPOINT - Cria ponto intermediário

RELEASE SAVEPOINT - Remove savepoint

SET TRANSACTION - Define propriedades da

transação

LOCK TABLE - Bloqueia tabela para consistência

COMANDOS SQL



Consultas e Operações Avançadas

Filtragem - WHERE, LIKE, BETWEEN, IN

Ordenação - ORDER BY

Agrupamento - GROUP BY, HAVING

Joins

INNER JOIN

LEFT JOIN

RIGHT JOIN

FULL JOIN

Subconsultas (Subquery)

Views, Stored Procedures, Functions, Triggers

COMANDOS SQL



Transações e Integridade

Transações ACID - Atomicidade, Consistência,
Isolamento, Durabilidade

Controle de concorrência - Locks, Deadlocks

Integridade de dados

Primary Key

Foreign Key

Unique

Check

Not Null

COMANDOS SQL



Administração de Banco de Dados

Backup e Restore

Monitoramento de desempenho

Indexação e otimização de consultas

Usuários e permissões

Auditoria e logs



Aula 6



MODELAGEM DE DADOS

MODELO CONCEITUAL

MODELO LÓGICO

MODELO FÍSICO

MODELO CONCEITUAL

Focado no entendimento dos requisitos do sistema e do negócio

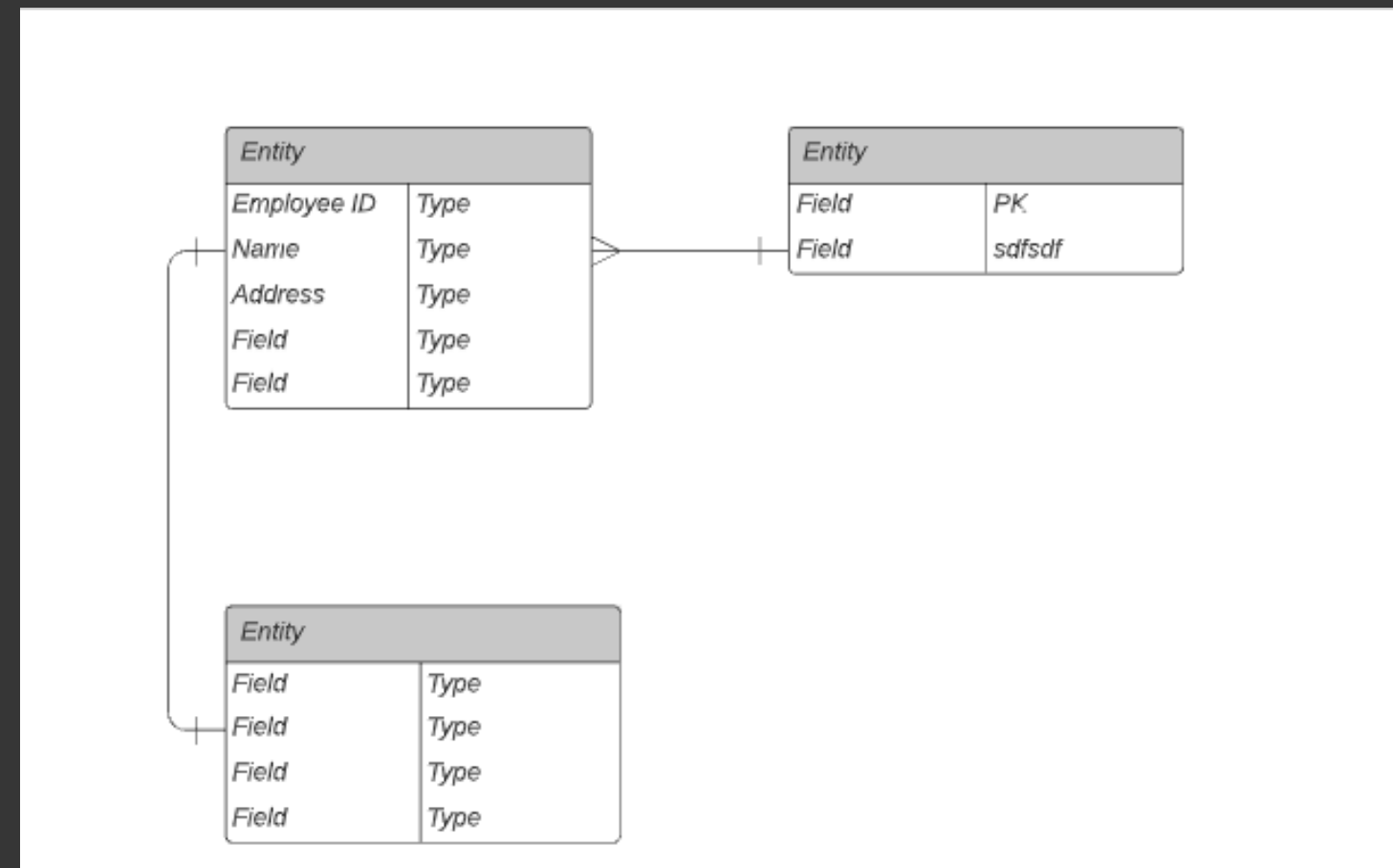
Entender como o negócio funciona para estruturar o BD

Existem outros modelos: o lógico e o físico.

MODELO LÓGICO

Criado para realizar a descrição de como os dados serão armazenados no sistema.

Nesse modelo, descrevemos as entidades, os atributos, as chaves primárias e estrangeiras e os seus relacionamentos.

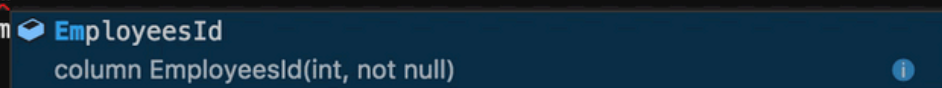


MODELO FÍSICO

Criado para descrever as tabelas, suas colunas e os relacionamentos.

Utilizamos uma linguagem padrão para realizar essa representação: a linguagem SQL, utilizada para trabalhar com BD relacionais.

```
30
31  -- Insert rows into table 'Employees'
32  INSERT INTO Employees
33      ([EmployeesId],[Name],[Location])
34  VALUES
35      ( 1, N'Jared', N'Australia'),
36      ( 2, N'Nikita', N'India'),
37      ( 3, N'Tom', N'Germany'),
38      ( 4, N'Jake', N'United States')
39  GO
40
41  -- Query all employee information
42  SELECT e.em
43  FROM dbo.Em EmployeesId
44  GO
45
46
```

A screenshot of a SQL IDE with a dark theme. A tooltip is visible over the 'EmployeesId' column in the 'FROM' clause of the second query. The tooltip is a dark blue rectangle with white text that reads 'column EmployeesId(int, not null)' and includes a small information icon on the right side.