SQL na Plataforma 9¾:

A Magia nos Dados

Microsoft® SQL Server®

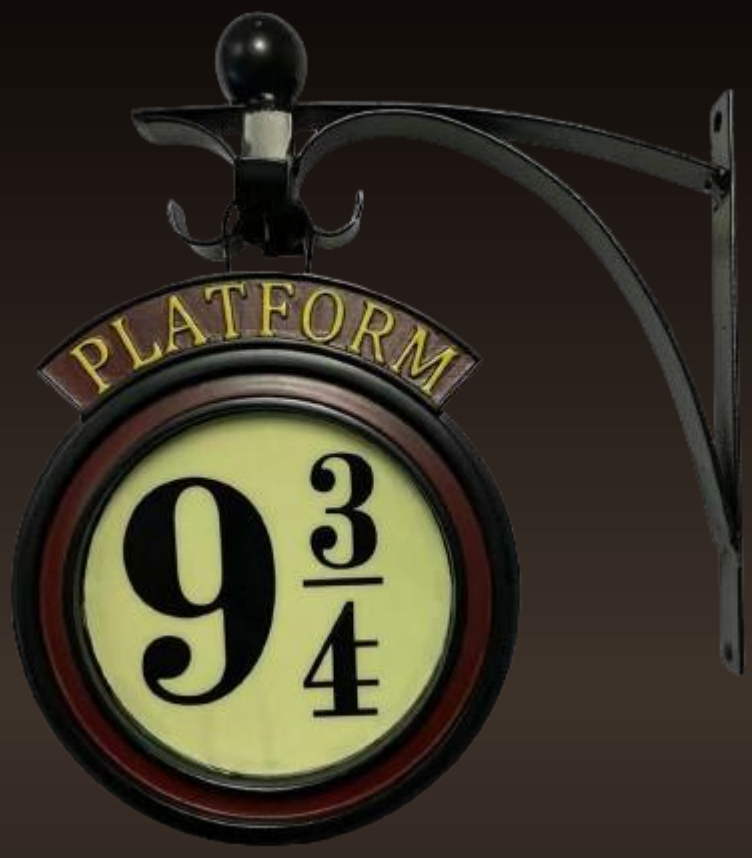
Introdução

Bem-vindo à "Plataforma 9¾: Magia nos Dados", onde exploraremos a fascinante interseção entre o mundo mágico e a ciência dos dados com SQL. Este e-book é sua passagem para descobrir como a linguagem SQL pode transformar dados brutos em informações mágicas e insights poderosos.

=

Capítulo 1:

A Chegada à Plataforma



Descubra as origens da Structured Query Language e sua evolução ao longo dos anos.

Microsoft® SQL Server®

A Chegada à Plataforma 9¾

Origem do SQL E Conceitos Básicos

A Structured Query Language, ou SQL, é a linguagem padrão para gerenciamento de banco de dados relacionais. Ela surgiu na década de 1970. Desde então, SQL passou por várias padronizações e evoluções, solidificando-se como a ferramenta essencial para interação com bancos de dados relacionais em muitas aplicações.

SQL, apesar de ser uma linguagem poderosa, possui uma estrutura simples e intuitiva que são compostas de palavras-chave que indicam a ação a ser executada, seguidas por especificações sobre quais dados devem ser manipulados. Aqui está um exemplo básico de uma consulta SQL:

```sql
SELECT nome, idade
FROM alunos
WHERE idade > 18;
```

Microsoft® SQL Server®

Sintaxe Básica

Decompondo o exemplo anterior:

SELECT: Palavra-chave que indica que queremos selecionar dados.
FROM: Palavra-chave que indica de qual tabela estamos extraindo os dados.
WHERE: Palavra-chave que especifica a condição que os dados devem satisfazer
NOME E IDADE: As colunas que desejamos retornar.
ALUNOS: A tabela de onde os dados serão extraídos.
IDADE > 18: A condição que os dados devem cumprir.

Tipos de Dados

Aqui estão alguns dos tipos de dados mais comuns que podem ser armazenados em uma tabela.

- **INT**: Números inteiros.
- **VARCHAR**: Cadeias de caracteres de comprimento variável.
- **DATE**: Datas.
- **FLOAT**: Números de ponto flutuante.

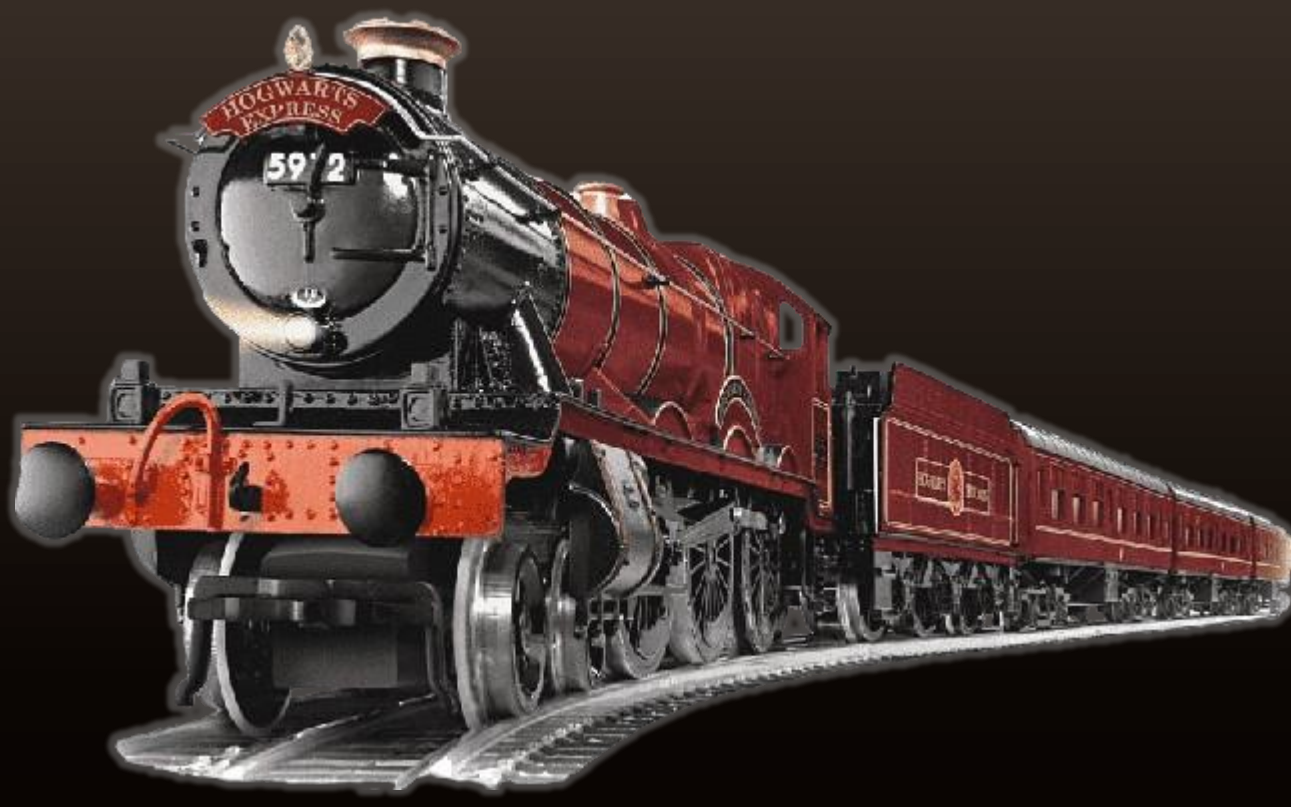
Cada coluna em uma tabela deve ser declarada com um tipo de dado específico, que define o tipo de valores que a coluna pode armazenar.

Microsoft® SQL Server®

Capítulo 2:

O Expresso de Hogwarts e a Estrutura dos Dados

Domine a arte de criar e organizar tabelas para armazenar dados de maneira eficiente.



Microsoft®
SQL Server®

O Expresso de Hogwarts e a Estrutura dos Dados

Modelagem de Dados

A modelagem de dados é fundamental para organizar e estruturar dados de maneira eficiente. Este capítulo aborda três conceitos principais: tabelas e esquemas, chaves primárias e estrangeiras, e normalização.

Tabelas e Esquemas

As tabelas são a base de um banco de dados relacional, organizando dados em linhas e colunas. Um esquema é uma coleção de tabelas relacionadas que define a estrutura do banco de dados. Para criar uma tabela, usamos o comando CREATE TABLE, especificando colunas e tipos de dados.

```sql
CREATE TABLE alunos (
    id INT PRIMARY KEY,
    nome VARCHAR(100),
    idade INT,
    casa VARCHAR(50)
    );
```

Microsoft SQL Server

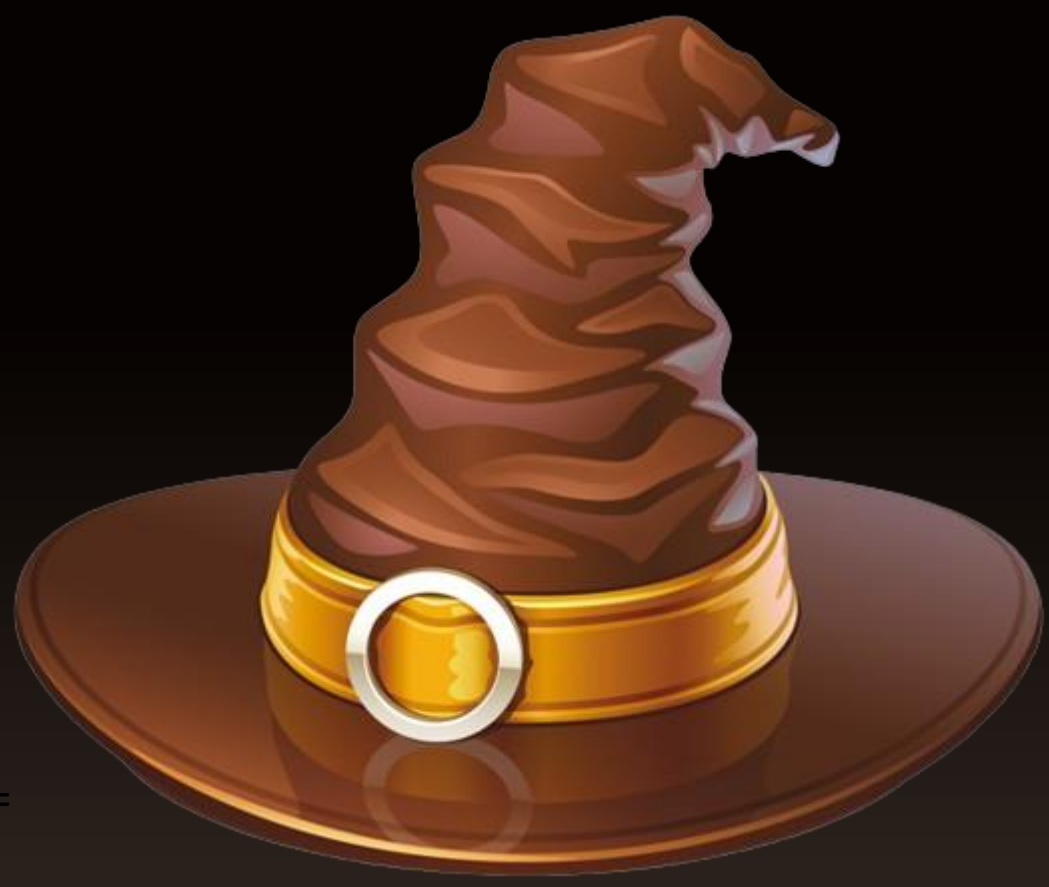
Chaves Primárias e Estrangeiras

Chaves primárias identificam unicamente cada registro em uma tabela, garantindo a integridade dos dados. Chaves estrangeiras criam vínculos entre tabelas, referenciando chaves primárias de outras tabelas para estabelecer relacionamentos claros entre os dados.

```sql
CREATE TABLE matriculas (
    id INT PRIMARY KEY,
    aluno_id INT,
    disciplina_id INT,
    FOREIGN KEY (aluno_id) REFERENCES alunos(id),
    FOREIGN KEY (disciplina_id) REFERENCES disciplinas(id)
);
```

Normalização

A normalização organiza os dados para reduzir a redundância e melhorar a integridade. Dividida em várias formas normais (1NF, 2NF, 3NF), a normalização assegura que os dados sejam armazenados de maneira eficiente, evitando duplicações e mantendo a consistência.



=

Capítulo 3:

O Chapéu Seletor e a Consulta de Dados

Este capítulo explora como usar SQL para selecionar e manipular dados de forma eficiente

Microsoft® SQL Server®

O Chapéu Seletor e a Consulta de Dados

Consultas Básicas e Avançadas

Select Básico

A instrução SELECT é a base das consultas SQL, usada para extrair dados de tabelas.

```sql
SELECT coluna1, coluna2 FROM tabela;
```

Filtrando Dados com Where

A cláusula WHERE permite filtrar os dados retornados por uma consulta, especificando condições que os dados devem atender. Por exemplo, para buscar alunos com mais de 18 anos:.

```sql
SELECT nome, idade
FROM alunos
WHERE idade > 18;
```

Microsoft SQL Server

Junções (JOINS)

São usados para combinar dados de múltiplas tabelas com base em um relacionamento comum. Existem vários tipos de JOINS:

INNER JOIN: Retorna apenas as linhas que têm correspondências em ambas as tabelas.

```sql
SELECT alunos.nome, disciplinas.nome
FROM alunos
INNER JOIN matriculas ON alunos.id = matriculas.aluno_id
INNER JOIN disciplinas ON matriculas.disciplina_id = disciplinas.id;
```

LEFT JOIN: Retorna todas as linhas da tabela à esquerda, e correspondências da tabela à direita, ou NULL se não houver correspondência.

```sql
SELECT alunos.nome, matriculas.disciplina_id
FROM alunos
LEFT JOIN matriculas ON alunos.id = matriculas.aluno_id;
```

Microsoft® SQL Server®

RIGHT JOIN: Retorna todas as linhas da tabela à direita, e correspondências da tabela à esquerda, ou NULL se não houver correspondência.

```sql
SELECT alunos.nome, matriculas.disciplina_id
FROM alunos
RIGHT JOIN matriculas ON alunos.id = matriculas.aluno_id;
```

Subconsultas

É uma consulta dentro de outra consulta, usada para retornar dados que serão usados pela consulta externa. Por exemplo:

```sql
SELECT nome FROM alunos WHERE idade = (SELECT MAX(idade) FROM alunos);
```

Common Table Expressions (CTEs)

CTEs permitem definir uma consulta temporária que pode ser referenciada dentro da consulta principal, facilitando a leitura e manutenção de consultas complexas. A sintaxe básica é:

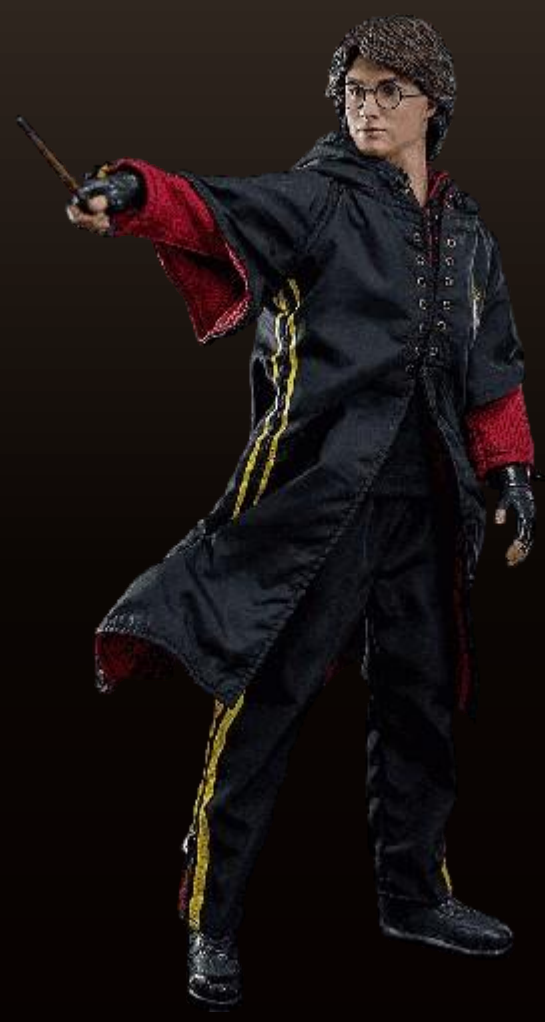
```sql
WITH CTE AS (
    SELECT nome, idade FROM alunos WHERE idade > 18
)
SELECT * FROM CTE;
```

Neste exemplo, a CTE armazena o resultado da consulta de alunos com mais de 18 anos, que é então utilizada na consulta principal.

Capítulo 4:

Feitiços de Manipulação de Dados

Com o domínio dos feitiços de manipulação de dados, você estará bem equipado para adicionar, atualizar e remover registros de maneira eficiente e segura



Microsoft® SQL Server®

Feitiços de Manipulação de Dados

Operações DML (Data Manipulation Language)

Manipular dados em um banco de dados é como lançar feitiços poderosos que permitem adicionar, atualizar e remover registros. As operações DML são essenciais para a interação dinâmica com os dados armazenados. Neste capítulo, vamos explorar três feitiços principais: INSERT, UPDATE e DELETE.

Insert: Adicionando Novos Registros às Tabelas

O feitiço INSERT é usado para adicionar novos registros em uma tabela. Imagine que você está adicionando novos alunos à lista de Hogwarts. A sintaxe básica é:

```
INSERT INTO tabela (coluna1, coluna2, coluna3)
VALUES (valor1, valor2, valor3);

Para adicionar um novo aluno à tabela alunos:

INSERT INTO alunos (nome, idade, casa)
VALUES ('Harry Potter', 17, 'Grifinória');
```

Microsoft SQL Server

Update: Atualizando Dados Existentes

O feitiço UPDATE permite modificar dados existentes em uma tabela. Através dele podemos corrigir informações ou atualizar registros conforme necessário. A sintaxe básica é:

```
UPDATE tabela SET coluna1 = valor1, coluna2 = valor2
WHERE condição;

Para atualizar a idade de Harry Potter na tabela alunos:

UPDATE alunos SET idade = 18
WHERE nome = 'Harry Potter';
```

Neste exemplo, o comando altera a idade de Harry Potter para 18 anos sendo o nome igual a 'Harry Potter'. É importante incluir uma cláusula WHERE para especificar quais registros devem ser atualizados, evitando modificar todos os registros da tabela inadvertidamente

Microsoft® SQL Server®

Delete: Removendo Dados Indesejados

O feitiço DELETE é usado para remover registros indesejados de uma tabela. Este comando deve ser usado com cuidado, pois a remoção de dados é permanente. A sintaxe básica é:

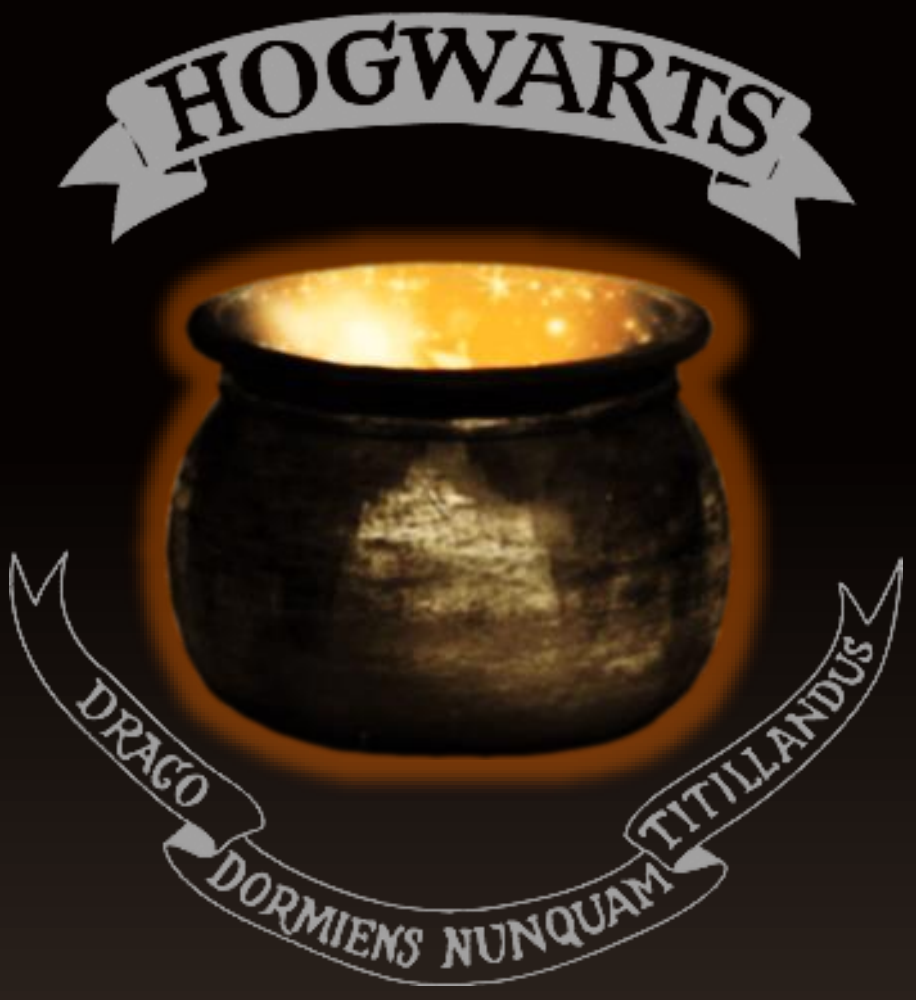
```
DELETE FROM tabela WHERE condição;


Para remover um aluno chamado Draco Malfoy da tabela alunos:

DELETE FROM alunos WHERE nome = 'Draco Malfoy';
```

No exemplo, o comando remove todos os registros onde o nome é 'Draco Malfoy'. Assim como no comando UPDATE, é crucial usar uma cláusula WHERE para especificar quais registros devem ser excluídos.

Boas Práticas de Manipulação de Dados

Para garantir a integridade e segurança dos dados ao realizar operações DML, considere as seguintes boas práticas:

- Backups Regulares: Faça backups regulares dos dados antes de realizar operações DML, especialmente operações DELETE.
- Validação de Dados: Valide os dados antes de inserir ou atualizar, garantindo que eles atendam aos critérios de qualidade e consistência.
- Cláusula WHERE: Sempre use cláusulas WHERE específicas em comandos UPDATE e DELETE para evitar alterações não intencionais em todos os registros da tabela.



=

Capítulo 5:

Poções e Procedimentos Armazenados

Assim como as poções em Hogwarts podem simplificar tarefas complexas, procedures e funções no SQL automatizam e otimizam operações no banco de dados.

Microsoft® SQL Server®

Poções e Procedimentos Armazenados

Procedimentos Armazenados

São conjuntos de instruções SQL que são armazenados no banco de dados e podem ser executados sob demanda. Eles são úteis para automatizar tarefas repetitivas e garantir que operações complexas sejam executadas de maneira consistente. A sintaxe básica para criar um procedimento armazenado é:

```
CREATE PROCEDURE nome_do_procedimento
AS
BEGIN
    -- Instruções SQL
END;

Para um procedimento de adicionar um novo aluno

CREATE PROCEDURE AddAluno
    @Nome VARCHAR(100),
    @Idade INT,
    @Casa VARCHAR(50)
AS
BEGIN
    INSERT INTO alunos (nome, idade, casa)
    VALUES (@Nome, @Idade, @Casa);
END;
```

Para executar o procedimento armazenado, usamos o comando EXEC:

```
EXEC AddAluno 'Hermione Granger', 17, 'Grifinória';
```

Microsoft SQL Server

Funções Definidas pelo Usuário (UDFs)

As UDFs permitem criar funções personalizadas que podem ser usadas em consultas SQL para realizar operações repetitivas ou complexas. Existem dois tipos principais de UDFs: funções escalares, que retornam um único valor, e funções de tabela, que retornam uma tabela.

Funções Escalares

A sintaxe básica para criar uma função escalar é:

```
CREATE FUNCTION nome_da_funcao (@parametro tipo_de_dado)
RETURNS tipo_de_dado
AS
BEGIN
    DECLARE @Resultado tipo_de_dado;
    -- Cálculos ou operações
    RETURN @Resultado;
END;
```

Função para calcular a idade de um aluno com base no ano de nascimento:

```
CREATE FUNCTION CalcularIdade (@AnoNascimento INT)
RETURNS INT
AS
BEGIN
    DECLARE @Idade INT;
    SET @Idade = YEAR(GETDATE()) - @AnoNascimento;
    RETURN @Idade;
```

Microsoft SQL Server

Podemos usar a função em uma consulta SQL como qualquer outra função integrada:

```sql
SELECT nome, dbo.CalcularIdade(ano_nascimento) AS idade FROM alunos;
```

Função de Tabela

Funções de tabela retornam um conjunto de resultados. A sintaxe básica é:

```sql
CREATE FUNCTION nome_da_funcao (@parametro tipo_de_dado)
RETURNS TABLE
AS
RETURN
(
    -- Instruções SELECT que retornam uma tabela
);

Função que retorna todos os alunos de uma casa específica:

CREATE FUNCTION AlunosPorCasa (@Casa VARCHAR(50))
RETURNS TABLE
AS
RETURN
(
    SELECT nome, idade FROM alunos WHERE casa = @Casa
);
```

Podemos usar a função em uma consulta SELECT como uma tabela:

```
SELECT * FROM dbo.AlunosPorCasa('Grifinória');
```

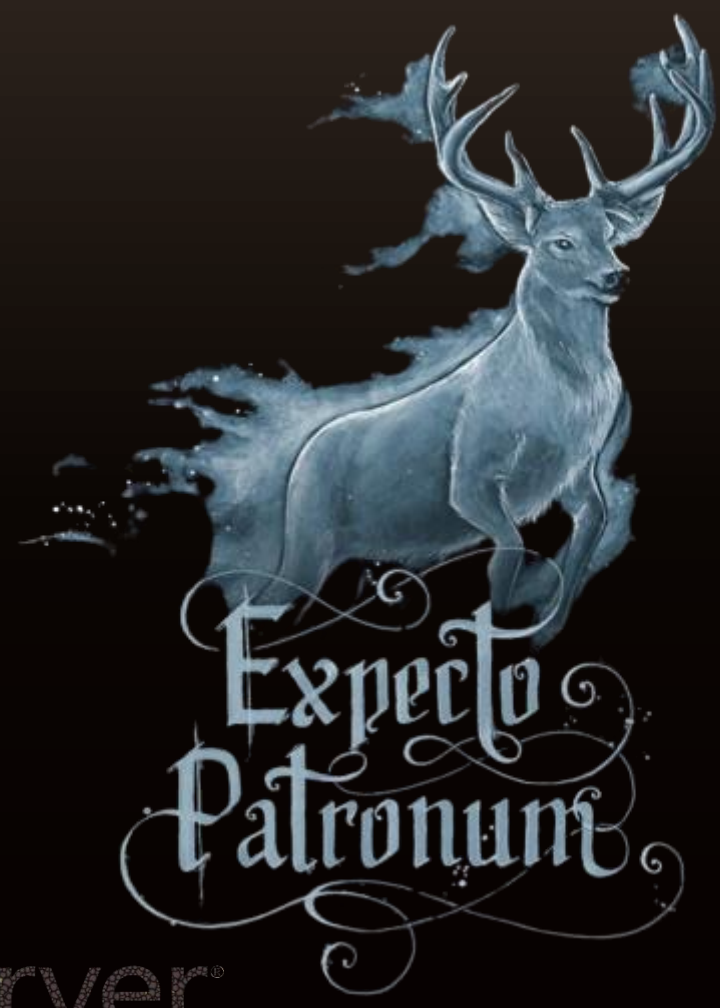
Benefícios de Procedures e Funções

- Reutilização de Código: Procedures e funções permitem encapsular lógica complexa que pode ser reutilizada em várias partes do banco de dados.
- Manutenção: Centralizar a lógica em procedures e funções facilita a manutenção e a atualização do código.
- Desempenho: Procedures armazenadas podem ser pré-compiladas, melhorando o desempenho das operações repetitivas.

Capítulo 6:

Defesa Contra as Artes das Consultas

Ao dominar essas técnicas de otimização, você será capaz de defender seu banco de dados contra consultas ineficientes, garantindo um desempenho robusto e sustentável



Microsoft® SQL Server®

Defesa Contra as Artes das Consultas

Índices

Índices são estruturas que melhoram a velocidade das buscas em tabelas. Eles atuam como atalhos para localizar dados rapidamente. Criar índices nas colunas usadas frequentemente em condições de busca (WHERE) e junções (JOIN) é crucial para melhorar a performance das consultas. A criação de índices pode ser feita usando a instrução CREATE INDEX:

```sql
CREATE INDEX idx_nome ON alunos(nome);
```

Este comando cria um índice na coluna nome da tabela alunos, melhorando a performance das consultas que filtram por nome.

Microsoft® SQL Server®

Análise de Plano de Execução

O plano de execução detalha a estratégia do banco de dados para acessar e processar dados. Analisar e entender este plano ajuda a identificar e corrigir gargalos, otimizando o caminho que as consultas seguem para acessar os dados. Ferramentas como EXPLAIN no SQL Server permitem visualizar e ajustar esses planos.

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

Este comando fornece um detalhamento de como a consulta será executada, mostrando operações como scans de tabelas e índices, junções e ordenações.

Microsoft® SQL Server®

Dicas de Performance

Escrever consultas eficientes é fundamental para manter o desempenho do banco de dados. Práticas recomendadas incluem:

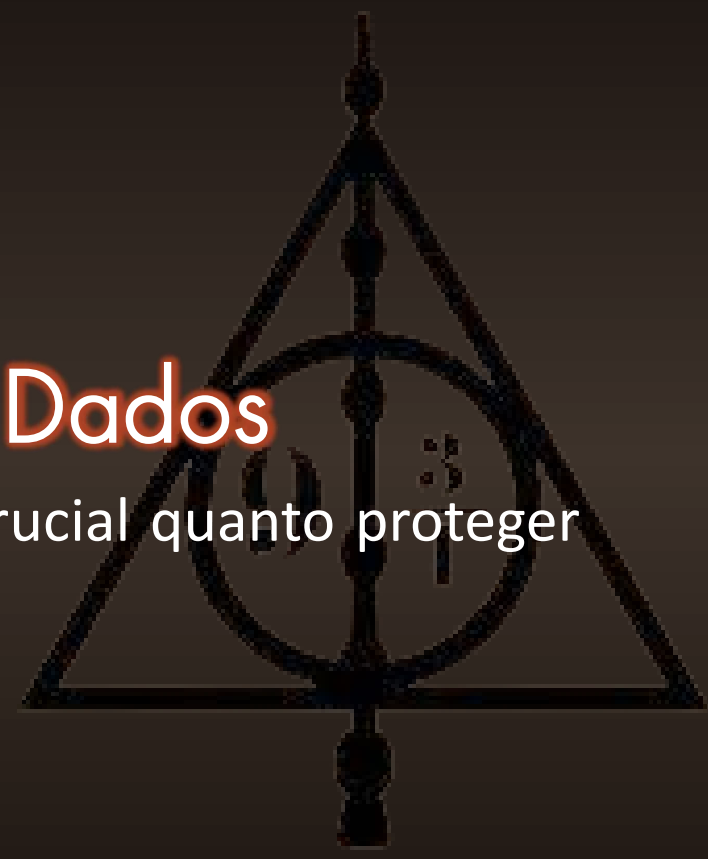
- Selecione Apenas o Necessário: Evite SELECT * e especifique apenas as colunas necessárias.
- Use Índices Apropriadamente: Crie índices nas colunas usadas frequentemente em condições de busca e junções.
- Evite Funções em Colunas Filtradas: Funções aplicadas em colunas na cláusula WHERE podem impedir o uso de índices.
- Normalize suas Tabelas: Aplique a normalização para evitar redundância e melhorar a eficiência das consultas.
- Limite o Uso de Subconsultas: Prefira junções e Common Table Expressions (CTEs) quando apropriado.

Microsoft® SQL Server®

Capítulo 7:

Proteção das Relíquias de Dados

Proteger os dados de um banco é tão crucial quanto proteger relíquias preciosas.



Microsoft® SQL Server®

Proteção das Relíquias de Dados

Controle de Acesso

O controle de acesso envolve a gestão de permissões e roles para proteger os dados contra acessos não autorizados. Permissões podem ser concedidas ou revogadas a usuários específicos, enquanto roles permitem agrupar permissões para simplificar o gerenciamento.

Permissões determinam o que um usuário pode fazer no banco de dados. Elas podem ser concedidas ou revogadas para tabelas, colunas e outros objetos do banco de dados.

```
Para conceder permissão:

GRANT SELECT ON alunos TO usuario1;

Para revogar permissão:

REVOKE SELECT ON alunos FROM usuario1;
```

Microsoft® SQL Server®

Roles

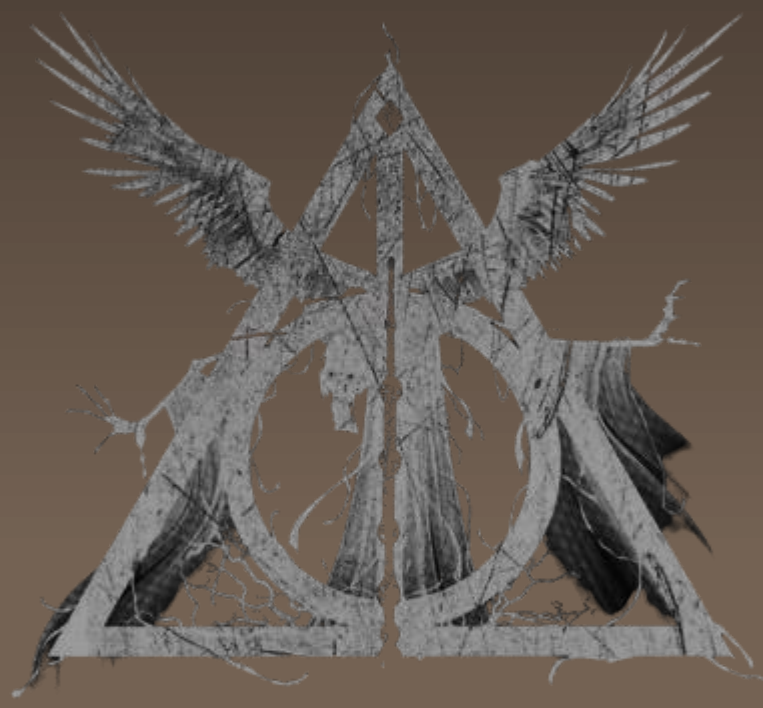
São conjuntos de permissões que podem ser atribuídos a usuários ou grupos de usuários. Elas simplificam o gerenciamento de permissões, especialmente em ambientes com muitos usuários. Por exemplo, criar uma role com permissões de leitura e atribuí-la a um usuário:

```
CREATE ROLE LeituraDados;
GRANT SELECT ON alunos TO LeituraDados;
GRANT LeituraDados TO usuario1;
```

Backup e Recuperação

Garantir a integridade e a disponibilidade dos dados em casos de falhas é essencial. Estratégias de backup e recuperação são fundamentais para proteger os dados contra perdas e garantir a continuidade das operações.

A recuperação de dados é o processo de restaurar o banco de dados a partir dos backups. Este processo deve ser testado regularmente para garantir que os backups funcionem corretamente e que os dados possam ser restaurados rapidamente em caso de necessidade.



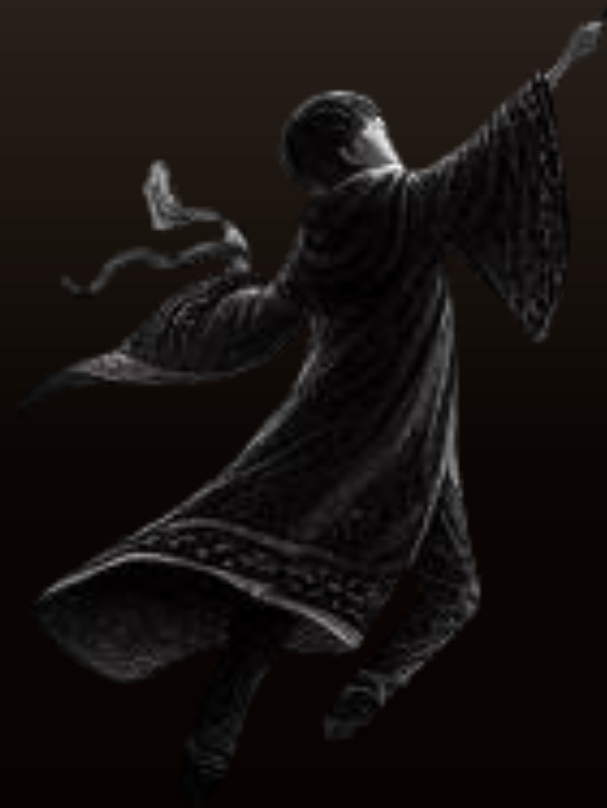
Microsoft® SQL Server®

=

Capítulo 8:

A Câmara Secreta das Técnicas Avançadas

Entrar na Câmara Secreta em SQL é explorar poderosas funcionalidades que permitem o controle preciso e a automação de tarefas complexas no banco de dados.



Microsoft® SQL Server®

A Câmara Secreta das Técnicas Avançadas

Transações

São um blocos de operações SQL executadas como uma única unidade de trabalho. Elas garantem que todas as operações dentro da transação sejam concluídas com sucesso ou nenhuma delas seja aplicada, mantendo a integridade dos dados. As propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade) são fundamentais para transações.

```
Transferir fundos entre duas contas:

BEGIN TRANSACTION;
UPDATE contas SET saldo = saldo - 100 WHERE conta_id = 1;
UPDATE contas SET saldo = saldo + 100 WHERE conta_id = 2;
IF @@ERROR <> 0
    ROLLBACK;
ELSE
    COMMIT;
```

Se qualquer operação falhar, um ROLLBACK pode reverter as mudanças:

Microsoft® SQL Server®

Concurrency (concorrência)

Lida com o acesso simultâneo aos dados por múltiplos usuários. Técnicas de controle de concorrência, como bloqueios (locks) e níveis de isolamento, ajudam a evitar conflitos e garantir a integridade dos dados.

Os níveis de isolamento controlam o grau de visibilidade das mudanças feitas por uma transação a outras transações:

- Read Uncommitted: Permite leituras sujas (dirty reads).
- Read Committed: Evita leituras sujas.
- Repeatable Read: Evita leituras não repetíveis.
- Serializable: Oferece a maior proteção, evitando leituras fantasmas.

Triggers

São instruções SQL que se ativam automaticamente em resposta a eventos específicos em tabelas, como inserções, atualizações ou deleções. Eles permitem a automação de respostas a eventos e a aplicação de regras de negócios complexas. A sintaxe para criar um trigger varia conforme o SGBD, mas geralmente segue este formato:

```
CREATE TRIGGER NomeDoTrigger
ON NomeDaTabela
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    -- Ações a serem executadas
END;

Para trigger que registra alterações em uma tabela de auditoria:

CREATE TRIGGER RegistroAuditoria
ON alunos
AFTER UPDATE
AS
BEGIN
    INSERT INTO auditoria_alunos (aluno_id, data_alteracao, detalhes)
    SELECT id, GETDATE(), 'Dados atualizados'
    FROM inserted;
END;
```

Microsoft® SQL Server®

Cursors

Permitem a manipulação linha por linha de conjuntos de resultados de consultas. Eles são úteis quando operações complexas precisam ser executadas em cada linha de um conjunto de dados. A sintaxe básica para declarar, abrir, buscar e fechar um cursor é:

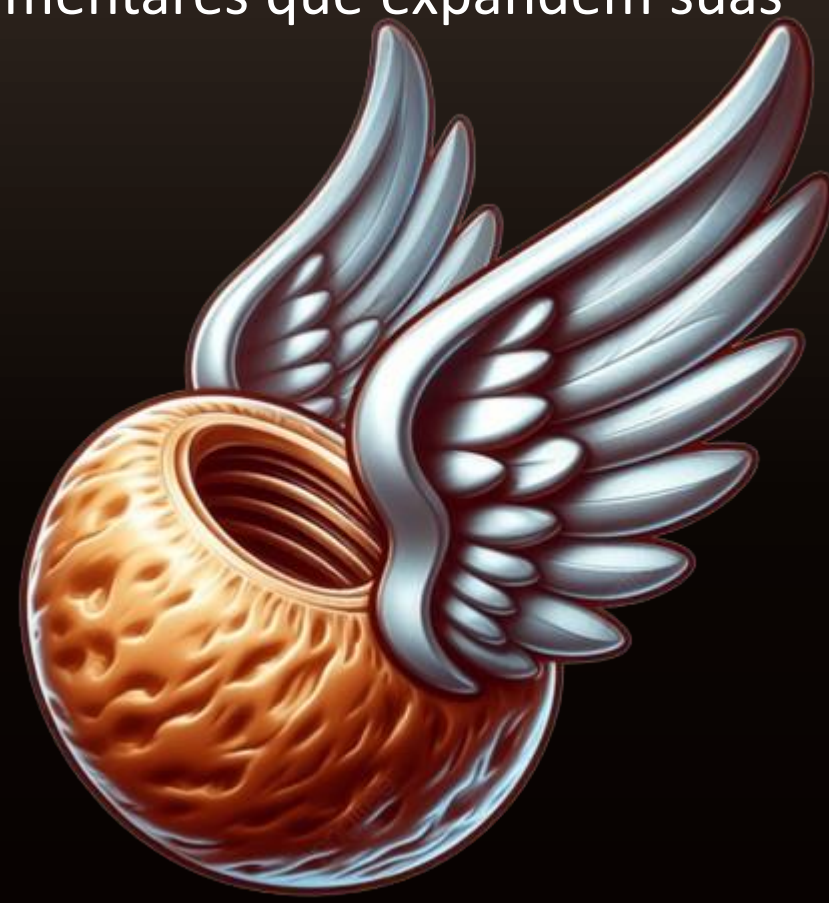
```sql
DECLARE nome_cursor CURSOR FOR
SELECT coluna1, coluna2 FROM tabela;
OPEN nome_cursor;
FETCH NEXT FROM nome_cursor INTO @var1, @var2;
WHILE @@FETCH_STATUS = 0
BEGIN
    -- Operações a serem realizadas em cada linha
    FETCH NEXT FROM nome_cursor INTO @var1, @var2;
END; CLOSE nome_cursor; DEALLOCATE nome_cursor;

Cursor que calcula e atualiza a média de notas dos alunos:

DECLARE cursor_notas CURSOR FOR
SELECT aluno_id, AVG(nota) FROM notas GROUP BY aluno_id;
OPEN cursor_notas;
FETCH NEXT FROM cursor_notas INTO @aluno_id, @media_nota;
WHILE @@FETCH_STATUS = 0
BEGIN
    UPDATE alunos SET media_nota = @media_nota WHERE id = @aluno_id;
    FETCH NEXT FROM cursor_notas INTO @aluno_id, @media_nota;
END; CLOSE cursor_notas; DEALLOCATE cursor_notas;
```

Microsoft® SQL Server®

Capitulo 9:

Magia no Mundo Real

Exploraremos como a magia da SQL se manifesta no mundo real através de aplicações práticas em diferentes indústrias e a utilização de ferramentas complementares que expandem suas funcionalidades



Microsoft® SQL Server®

Magia no Mundo Real

Ferramentas e Recursos Adicionais

SQL é uma ferramenta poderosa por si só, mas pode ser ainda mais eficaz quando usada em conjunto com outras ferramentas e tecnologias.

ETL (Extract, Transform, Load)

ETL é um processo utilizado para mover e transformar dados de várias fontes para um banco de dados central. Ferramentas ETL permitem que dados sejam extraídos, transformados e carregados de maneira eficiente.

Exemplo de Processo ETL

- Extract: Extrair dados de sistemas de e-commerce e CRM.
- Transform: Limpar e padronizar os dados.
- Load: Carregar os dados em um data warehouse para análise.

Microsoft® SQL Server®

BI (Business Intelligence)

Ferramentas de Business Intelligence, como Power BI, Tableau e Looker, utilizam SQL para criar dashboards interativos e relatórios detalhados. Essas ferramentas ajudam as empresas a tomar decisões informadas com base em dados.

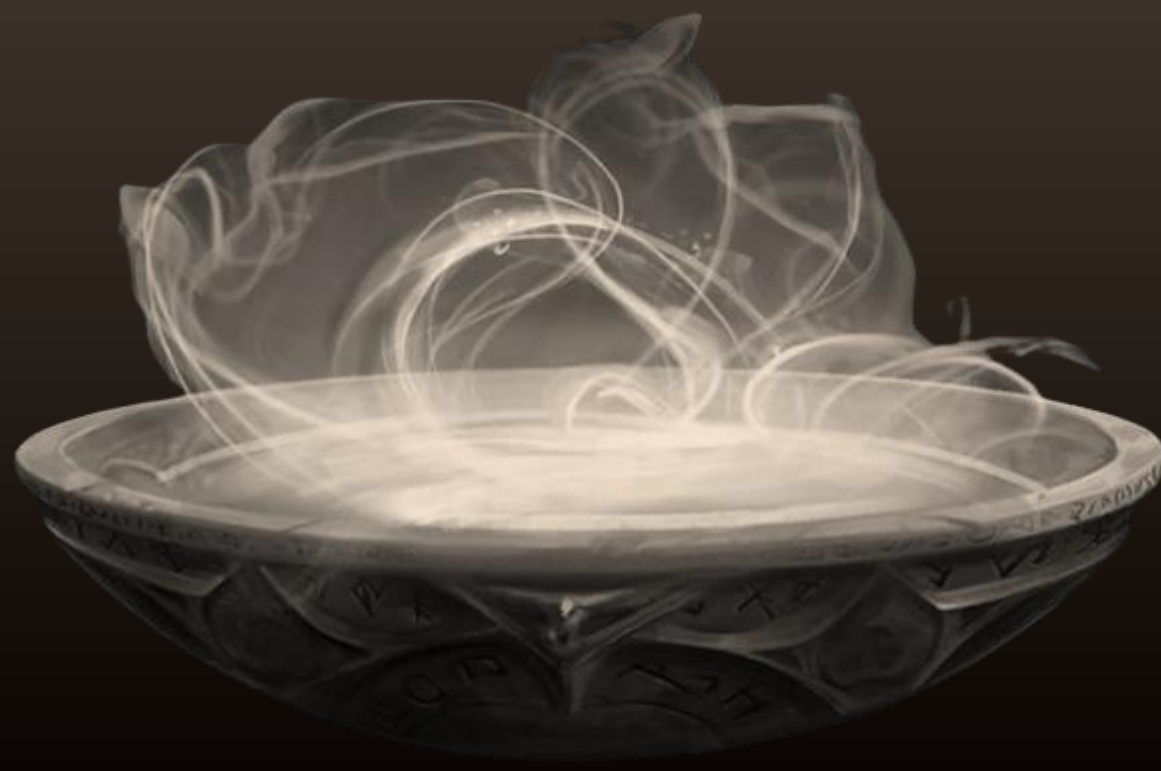
Exemplo de Integração BI
Usar SQL para extrair dados relevantes para criar um dashboard em Power BI que mostra métricas de vendas, tendências de mercado e desempenho de produtos em tempo real.

Microsoft® SQL Server®

Conclusão

A Magia Continua ...

Assim como na magia, o poder do SQL está em entender e aplicar suas regras e feitiços para transformar dados em valor.



Microsoft® SQL Server®

A Magia Continua ...

Chegamos ao fim da nossa jornada através da Plataforma 9¾, onde exploramos o mundo encantado da SQL e descobrimos o poder mágico dos dados. Desde os fundamentos até as técnicas avançadas, vimos como a SQL se destaca como uma ferramenta indispensável para o gerenciamento de dados em qualquer setor.

O SQL, abre um mundo de possibilidades para aqueles que se aventuram a dominá-lo. Este e-book é um começo, mas a jornada de aprendizado continua. Continue explorando e aplicando os conhecimentos adquiridos, sempre buscando novas formas de transformar dados em insights.

Microsoft® SQL Server®

Agradecimento

Este e-book foi especialmente criado para fins didáticos, utilizando a tecnologia de inteligência artificial para gerar todo o conteúdo, que passou por revisão humana para garantir a precisão e a clareza das informações.

Obrigada por nos acompanhar nesta viagem mágica pelo mundo da SQL. Que seu caminho seja repleto de descobertas e sucessos no fascinante universo dos dados.

DALL·E 2

ChatGPT

Microsoft® SQL Server®