MANUAL DE PROGRAMAÇÃO

Senac Vila Prudente

MySQL

O que é MySQL?

MySQL é um sistema de gerenciamento de banco de dados relacional (RDBMS) amplamente utilizado. Ele permite armazenar e recuperar dados de forma eficiente, facilitando a organização e manipulação de informações.

Instalação do MySQL

Antes de começar, é necessário instalar o MySQL no seu sistema. Você pode fazer isso baixando e instalando a versão adequada para o seu sistema operacional no site oficial do MySQL.

Conceitos Básicos

Conectando ao MySQL

A conexão ao MySQL pode ser feita por meio da linha de comando ou ferramentas gráficas. Utilizando o comando **mysql -u seu_usuario -p**, você acessa o MySQL inserindo seu nome de usuário e senha quando solicitado.

```
mysql -u seu_usuario -p
```

Criando um Banco de Dados

A criação de um banco de dados é essencial. O comando **CREATE DATABASE nome_do_banco;** estabelece um novo banco de dados para armazenar informações.

```
CREATE DATABASE nome_do_banco;
```

Selecionando um Banco de Dados

A instrução **USE nome_do_banco;** seleciona o banco de dados no qual você deseja executar comandos.

```
USE nome_do_banco;
```

Manipulação de Tabelas

Criando Tabelas

As tabelas são estruturas fundamentais no MySQL. O comando **CREATE TABLE** define uma tabela especificando os nomes e tipos de dados das colunas.

```
CREATE TABLE nome_da_tabela (
  id INT PRIMARY KEY,
  nome VARCHAR(50),
  idade INT
);
```

Inserindo Dados

O MySQL permite a inserção de dados em uma tabela usando o comando **INSERT INTO**. Isso é fundamental para preencher a tabela com informações relevantes.

```
INSERT INTO nome_da_tabela (id, nome, idade) VALUES (1, 'João', 25);
```

Consultando Dados

A instrução **SELECT** recupera dados de uma tabela. **SELECT * FROM nome_da_tabela;** retorna todos os registros da tabela especificada.

```
SELECT * FROM nome_da_tabela;
```

Atualizando Dados

Com o comando **UPDATE**, é possível modificar registros existentes em uma tabela. A cláusula **WHERE** define a condição para a atualização.

```
UPDATE nome_da_tabela SET idade = 26 WHERE nome = 'João';
```

Excluindo Dados

O comando **DELETE** remove registros de uma tabela. Assim como o **UPDATE**, a cláusula **WHERE** é usada para identificar quais registros excluir.

```
DELETE FROM nome_da_tabela WHERE nome = 'João';
```

Índices

Índices melhoram a eficiência das consultas. O comando C**REATE INDEX** cria um índice em uma coluna específica, acelerando a busca de dados.

```
CREATE INDEX idx_nome ON nome_da_tabela (nome);
```

Consultas Avançadas

Cláusula WHERE

A cláusula **WHERE** filtra os resultados da consulta com base em condições específicas. Exemplo: **SELECT * FROM nome_da_tabela WHERE idade > 30;** retorna registros onde a idade é superior a 30.

```
SELECT * FROM nome_da_tabela WHERE idade > 30;
```

Cláusula ORDER BY

A cláusula ORDER BY ordena os resultados da consulta. **SELECT * FROM nome_da_tabela ORDER BY idade DESC;** exibe os resultados ordenados por idade de forma decrescente.

```
SELECT * FROM nome_da_tabela ORDER BY idade DESC;
```

Cláusula JOIN

A cláusula JOIN combina dados de duas ou mais tabelas. Exemplo: **SELECT usuarios.nome, pedidos.produto FROM usuarios INNER JOIN pedidos ON usuarios.id = pedidos.usuario_id;** recupera informações relacionadas de duas tabelas.

```
SELECT usuarios.nome, pedidos.produto
FROM usuarios
INNER JOIN pedidos ON usuarios.id = pedidos.usuario_id;
```

Funções de Agregação

COUNT

```
SELECT COUNT(*) FROM nome_da_tabela;
```

SUM

```
SELECT SUM(idade) FROM nome_da_tabela;
```

AVG

```
SELECT AVG(idade) FROM nome_da_tabela;
```

Essas funções de agregação realizam cálculos em conjuntos de dados. **COUNT** conta o número de registros, **SUM** calcula a soma e **AVG** encontra a média.

Stored Procedures

Criando Stored Procedures

Stored procedures são conjuntos de instruções SQL que podem ser chamados pelo nome. O exemplo demonstra a criação de uma **stored procedure** para obter todos os usuários.

```
DELIMITER //
CREATE PROCEDURE obter_usuarios()
BEGIN
  SELECT * FROM usuarios;
END //
DELIMITER ;
```

Chamando Stored Procedures

Stored procedures são invocadas com o comando CALL. No exemplo, CALL **obter_usuarios();** executa a stored procedure criada.

```
CALL obter_usuarios();
```

Segurança e Práticas Recomendadas

Usuários e Permissões

Para garantir a segurança, criar usuários com senhas seguras e conceder permissões específicas é essencial. O exemplo ilustra a criação de um usuário e a concessão de privilégios.

```
CREATE USER 'seu_usuario'@'localhost' IDENTIFIED BY 'sua_senha';
GRANT ALL PRIVILEGES ON *.* TO 'seu_usuario'@'localhost';
```

Backup e Restauração

Realizar backups regulares é uma prática recomendada. O exemplo mostra como criar um backup e restaurar um banco de dados a partir dele.

```
# Backup
mysqldump -u seu_usuario -p nome_do_banco > backup.sql

# Restauração
mysql -u seu_usuario -p nome_do_banco < backup.sql
```

Transações

Iniciando uma Transação

Transações garantem a consistência dos dados. **START TRANSACTION;** inicia uma transação.

```
START TRANSACTION;
```

Commit e Rollback

COMMIT; efetiva as alterações em uma transação, enquanto **ROLLBACK;** desfaz as mudanças.

```
COMMIT;
-- ou
ROLLBACK;
```

MySQL e PHP

Conexão com PHP

O PHP pode se conectar ao MySQL utilizando a classe mysqli. O exemplo ilustra a configuração de uma conexão básica.

```
$servername = "localhost";
$username = "seu_usuario";
$password = "sua_senha";
$dbname = "nome_do_banco";

$conn = new mysqli($servername, $username, $password, $dbname);
```

Executando Consultas em PHP

Com a conexão estabelecida, consultas podem ser executadas utilizando o método query. O resultado é então processado, e os dados podem ser manipulados em PHP.

```php
$sql = "SELECT * FROM nome_da_tabela";
$result = $conn->query($sql);

while($row = $result->fetch_assoc()) {
  echo "Nome: " . $row["nome"] . ", Idade: " . $row["idade"];
}
```

Manipulação de Data e Hora

Trabalhando com Datas

O MySQL oferece funções para manipulação de datas. Por exemplo, NOW() retorna a data e hora atuais, enquanto DATE_FORMAT permite formatar datas de maneira específica.

```sql
-- Obtendo a data e hora atuais
SELECT NOW();

-- Formatando uma data
SELECT DATE_FORMAT(data_coluna, '%d/%m/%Y') AS data_formatada FROM nome_da_tabela;
```

Criando Triggers

Triggers são conjuntos de instruções automáticas ativadas por eventos no banco de dados, como inserção, atualização ou exclusão de registros.

```sql
CREATE TRIGGER nome_do_trigger
AFTER INSERT ON nome_da_tabela
FOR EACH ROW
BEGIN
  -- Instruções a serem executadas após uma inserção
END;
```

Views

Criando Views

Views são consultas armazenadas como tabelas virtuais. Elas simplificam consultas complexas e podem ser usadas para restringir o acesso a determinadas informações.

```
CREATE VIEW nome_da_view AS
SELECT coluna1, coluna2 FROM nome_da_tabela WHERE condição;
```

Índices Avançados

Índices Compostos

Índices compostos envolvem mais de uma coluna e são úteis para otimizar consultas que envolvem várias condições.

```
CREATE INDEX idx_composto ON nome_da_tabela (coluna1, coluna2);
```

Índices Full-Text

Índices full-text aprimoram pesquisas de texto em grandes conjuntos de dados.

```
CREATE FULLTEXT INDEX idx_texto ON nome_da_tabela (coluna_texto);
```

Replicação

Configuração de Replicação

A replicação MySQL permite a criação de cópias idênticas de um banco de dados em servidores diferentes, garantindo redundância e disponibilidade.

```
-- Configuração no servidor principal
CHANGE MASTER TO
  MASTER_HOST='ip_do_servidor',
  MASTER_USER='usuario_replicacao',
  MASTER_PASSWORD='senha_replicacao';

-- Iniciar replicação
START SLAVE;
```

Particionamento de Tabelas

Particionamento Horizontal

O particionamento divide grandes tabelas em partes menores, facilitando a administração e otimizando consultas.

```
CREATE TABLE nome_da_tabela (
  ...
) PARTITION BY RANGE (YEAR(data_coluna)) (
  PARTITION p0 VALUES LESS THAN (1990),
  PARTITION p1 VALUES LESS THAN (2000),
  PARTITION p2 VALUES LESS THAN (2010),
  PARTITION p3 VALUES LESS THAN MAXVALUE
);
```

Funções JSON

Manipulação de Dados JSON

O MySQL suporta funções para trabalhar com dados no formato JSON, facilitando a manipulação e extração de informações de documentos JSON.

```
-- Extraindo valor de um campo JSON
SELECT json_extract(dados_json, '$.nome') FROM nome_da_tabela;

-- Inserindo dados JSON
INSERT INTO nome_da_tabela (dados_json) VALUES ('{"nome": "Maria", "idade": 30}');
```

Integridade Referencial

Chaves Estrangeiras

As chaves estrangeiras mantêm a integridade referencial entre tabelas, garantindo que registros relacionados existam.

```
-- Adicionando chave estrangeira
ALTER TABLE tabela_filha
ADD CONSTRAINT fk_tabela_pai
FOREIGN KEY (coluna_id)
REFERENCES tabela_pai(id);
```

MySQL Shell

MySQL Shell é uma interface avançada para interagir com o MySQL. Além de SQL, oferece suporte a JavaScript e Python para automação e administração.

```
mysqlsh -u seu_usuario -p
```

Auditoria

Auditing

O MySQL oferece recursos de auditoria para registrar atividades, como consultas executadas, proporcionando transparência e segurança.

```
-- Habilitar auditoria global
SET GLOBAL audit_log=ON;
```

MySQL Workbench

MySQL Workbench é uma ferramenta gráfica que simplifica o gerenciamento de bancos de dados MySQL, fornecendo uma interface intuitiva para desenvolvimento e administração.

Sharding

Sharding envolve a distribuição de dados em vários servidores para melhorar o desempenho e a escalabilidade.

```
-- Sharding horizontal
CREATE TABLE nome_da_tabela (
  ...
) TABLESPACE = ts1;
```