BANCO DE DADOS I

MySQL - Introdução

Prof: Ricardo Rufino

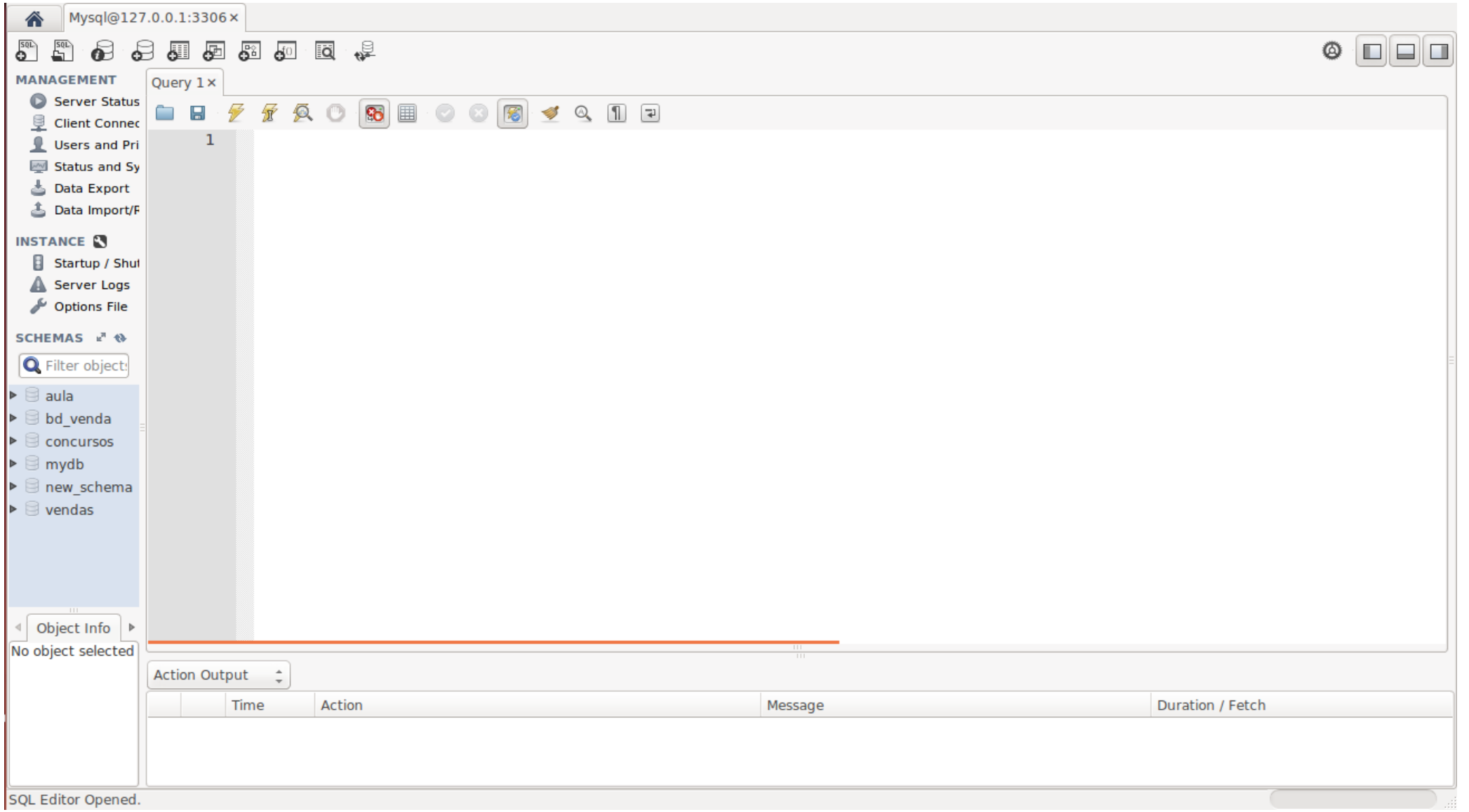
TEMAS ABORDADOS

- **Manipulação do banco de dados**
  - DDL – Linguagem de definição de dados
  - DML – Linguagem de manipulação de dados
- **Linguagem de definição de dados**
  - CREATE
  - DROP
  - ALTER
- **Linguagem de manipulação de dados**
  - INSERT
  - UPDATE
  - DELETE
  - SELECT

MySQL

- É um banco de dados relacional
- Oracle matem o projeto
- Software para a administração do SGBD
  → MysqlWorkbech

MySQL Workbench



Manipulando o banco de dados

- Para uma melhor familiarização com o SGBD, utilizaremos apenas comandos SQL.
- No MySQL Workbench é possível criar e manter um Banco de Dados, pois oferece suporte a execução de todos os comandos disponíveis.
  - Criar estruturas
  - Consultar dados
  - Incluir, alterar e excluir dados
  - Entre outros...

Manipulando o banco de dados

- A linguagem SQL é formada por dois grupos de comandos:
  - DDL:
    - Data Definition Language ou Linguagem de Definição de Dados
  - DML:
    - Data Manipulation Language ou Linguagem de Manipulação de Dados

Linguagem de Definição de Dados - DDL

- A **DDL** possui comandos que servem para **CRIAR**, **APAGAR** e **ALTERAR** objetos que fazem parte da **ESTRUTURA** do banco de dados como:
  - BANCO DE DADOS
  - TABELAS (ENTIDADES)
  - VIEWS
- Basicamente temos os seguintes comandos
  - CREATE
    - Cria objetos como banco de dados tabelas etc.
  - DROP
    - Remove
  - ALTER
    - Altera

Criando um banco de dados

- Abra a ferramenta MySQL Workbench.

Criando uma tabela

- Para criar uma tabela, será utilizado o comando CREATE TABLE como segue na sintaxe abaixo:

```
CREATE TABLE <nome da tabela>(
        <Em seu corpo, será criado os atributos da tabela>
);
```

- Após a criação da tabela, vamos efetuar a exclusão:

```
DROP TABLE <nome da tabela>;
```

Criando atributos

- Criaremos a tabela **aluno** novamente, mas desta vez com alguns atributos:
  - Nome, RA, idade

```
CREATE TABLE aluno(
    <atributo> <tipo>,
    <atributo> <tipo>,
    <atributo> <tipo>
);
```

Inserindo dados

- Posteriormente iremos inserir um aluno, com o seguinte comando:
  - INSERT INTO aluno VALUES('nome','ra',20);

    ou

  - INSERT INTO aluno

    (nome, ra, idade)VALUES('nome','ra',20);

Listando dados

- Para selecionar os dados inseridos, utilizaremos o comando SELECT.
  - SELECT * FROM aluno;

ou

  - SELECT nome,ra,idade FROM aluno;

Alterando os dados

- Para alterar os dados inseridos, utilizaremos o comando UPDATE.
  - UPDATE aluno SET idade=idade+5;

Deletar os dados

- Para deletar os dados inseridos, utilizaremos o comando DELETE.
  - DELETE FROM aluno;

Chave primaria

- Chave primaria, também conhecida como primary key “PK”, tem a função de tornar um registro único em uma tabela, de modo com que ele nunca se repita.

```
CREATE TABLE cliente (
    cliente_cpf varchar(11),
    PRIMARY KEY (cliente_cpf)
);
```

```
CREATE TABLE cliente (
    cliente_id auto_increment,      ← AUTO INCREMENTO
    PRIMARY KEY (cliente_id)
);
```

Chave estrangeira

- Chave estrangeira, também conhecida como foreign key “FK”, tem a função de relacionar registros entre tabelas. Desde que o registro relacionado exista, e que seja do mesmo tipo da PK que é referenciado.

```
CREATE TABLE cliente (
    cliente_cpf varchar(11),
    PRIMARY KEY (cliente_cpf)
);

CREATE TABLE venda (
    venda_id auto_increment,
    cliente_cpf  VARCHAR(11),
    PRIMARY KEY (venda_id),
    FOREIGN KEY (cliente_cpf) REFERENCES cliente(cliente_cpf)
);
```

Chave primaria composta

- Chave primaria composta consiste na junção de dois campos para formar um elemento único na tabela, por exemplo o CEP e o número de uma residencia.

```
CREATE TABLE endereco (
    end_cep varchar(20) NOT NULL, ← CAMPO OBRIGATORIO
    end_num varchar(20) NOT NULL, ← CAMPO OBRIGATORIO
    PRIMARY KEY (end_cep, end_num)
);
```

- Onde a junção dos dois registro foram uma chave única.

Constraints (Restrições)

- CHECK
  - Permite especificar que o valor de uma determinada coluna deve satisfazer uma expressão booleana (valor verdadeiro). Por exemplo, para exigir preços positivos do produto, você pode usar:

```
CREATE TABLE products (
    product_no integer,
    name text,
    price numeric CHECK (price > 0)
);
```

Constraints (Restrições)

- DEFAULT
  - A coluna pode ter atribuído com um valor padrão. Quando uma nova linha é criada e nenhum valor é especificado para algumas das colunas, as colunas serão preenchidas com os respectivos valores padrão.

```
CREATE TABLE products (
    product_no integer,
    name text,
    price numeric DEFAULT 9.99
);
```

Constraints (Restrições)

- NOT NULL
  - A restrição de não-nulo simplesmente especifica que uma coluna não deve assumir o valor nulo. Um exemplo de sintaxe:

```
CREATE TABLE products (
    product_no integer NOT NULL,
    name text NOT NULL,
    price numeric
);
```

Constraints (Restrições)

- UNIQUE
  - Restrição de unicidade garante que os dados contidos em uma coluna ou um grupo de colunas, é único no que diz respeito a todas as linhas da tabela. A sintaxe é:

```
CREATE TABLE products (
    product_no integer UNIQUE,
    name text,
    price numeric
);
```

Constraints (Restrições)

- PRIMAEY KEY
  - Tecnicamente, uma restrição de chave primária é simplesmente a combinação da restrição de unicidade e uma restrição de não-nulo. Assim, as duas definições de tabela abaixo aceitam os mesmos dados:

```
CREATE TABLE products (
    product_no integer UNIQUE NOT NULL,
    name text,
    price numeric
);

CREATE TABLE products (
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);
```

Constraints (Restrições)

- FOREIGN KEY
  - A restrição de chave estrangeira especifica que os valores em uma coluna (ou um grupo de colunas) deve corresponder aos valores que aparecem nas linha de outra tabela. Dizemos que isso mantém a integridade referencial entre duas tabelas relacionadas:

Constraints (Restrições)

- **FOREIGN KEY**

  CREATE TABLE orders (

  order_id **integer PRIMARY KEY**,

  product_no **integer**,

  quantity **integer**,

  PRIMARY KEY (order_id),

  FOREIGN KEY (product_no) **REFERENCES** products (product_no)

  );

- **OU PODEMOS ADICIONAR COM ALTER TABLE**

  ALTER TABLE orders **ADD FOREIGN KEY** (product_no) **REFERENCES** products (product_no);

- **PARA REMOVER UMA CONSTRAINT FOREIGN KEY**

  ALTER TABLE orders **DROP FOREIGN KEY** orderns_ibfk_1;

ALTER TABLE

- ADD COLUMN
  - Adiciona uma nova coluna à tabela, usando a mesma sintaxe do comando CREATE TABLE. Como mostra o exemplo abaixo.

ALTER TABLE distributors **ADD COLUMN** address varchar(30);

ALTER TABLE

- DROP COLUMN
  - Remove uma coluna de uma tabela. Os índices e as restrições da tabela que envolvem a coluna será descartado automaticamente também. Você vai precisar de CASCADE se algum objeto fora da tabela depender da coluna, por exemplo, referências de chaves estrangeiras.

ALTER TABLE distributors **DROP COLUMN** address;

ALTER TABLE

- ALTER TYPE
  - Esta forma muda o tipo de uma coluna de uma tabela. Índices e restrições de tabela simples que envolvem a coluna será convertido automaticamente para usar o novo tipo de coluna .

ALTER TABLE distributors **MODIFY** address **varchar(80);**

ALTER TABLE

- ADD CONSTRAINT
  - Adiciona uma nova restrição a uma tabela utilizando a mesma sintaxe do comando CREATE TABLE.

ALTER TABLE distributors **MODIFY** street **varchar(255) NOT NULL**;

- DROP CONSTRAINT
  - Esta forma remove a restrição especificada em uma tabela.

ALTER TABLE distributors **MODIFY** street **varchar(255)**;

ALTER TABLE

- ALTERAR NOME DE UMA TABELA

ALTER TABLE pessoa **RENAME TO** pessoas;

- ALTERAR NOME DE UMA COLUNA DA TABELA

ALTER TABLE pessoa **CHANGE** nome nome_pessoa **VARCHAR(255);**

Clausula WHERE

- Clausula de comparação, amplamente utilizada em códigos DML, pois seu objetivo é filtrar o dados.
- WHERE = ONDE.

Código:

SELECT cli_nome **FROM** cliente **WHERE** cli_nome='FULANO';

Tradução:

SELECIONE cli_nome **DA TABELA** cliente **ONDE** cli_nome **SEJA IGUAL A** 'FULANO';

Clausula WHERE

- Podemos utilizar em:
  - SELECT
    - Para selecionar apenas o que for solicitado.
  - UPDATE
    - Para alterar apenas o que for solicitado
  - DELETE
    - Para deletar apenas o que for selecionado