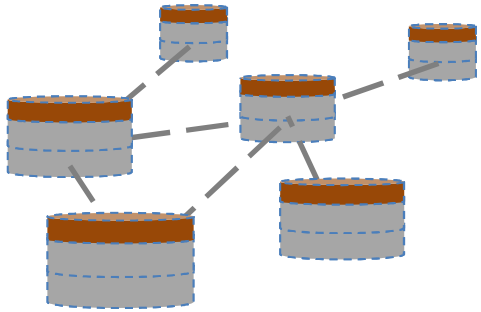


Sistemas de Bases de Dados

Notas de Leitura

05 >> A Linguagem SQL

Orlando Belo

Departamento de Informática, Escola de Engenharia, Universidade do Minho
PORTUGAL

> www.di.uminho.pt/~omb
> algoritmi.uminho.pt/orlandobelo
> https://orcid.org/0000-0003-2157-8891
> www.researchgate.net/profile/Orlando_Belo
> https://www.linkedin.com/in/orlando-belo-9431942a/?originalSubdomain=pt

Rev. 2022

05

Resumo

Nesta unidade de notas de leitura abordamos a linguagem SQL, em particular a sua aplicação prática a situações concretas que possam ocorrer no mundo real em sistemas de bases de dados relacionais. Ao longo da unidade abordaremos um conjunto vasto e diverso de instruções da SQL, cobrindo as suas principais vertentes de atuação, nomeadamente, descrição de dados, manipulação de dados e controlo de dados. Os exemplos que são apresentados foram idealizados e trabalhos no contexto da base de dados “Sakila”, uma das bases de dados que o MySQL disponibiliza aquando da sua instalação. Todos os exemplos de aplicação da SQL que estão apresentados nesta unidade foram desenvolvidos utilizando a implementação específica (dialeto) da SQL que o MySQL 8.0 utiliza. De referir que, dadas as características e funcionalidades deste sistema, as instruções apresentadas poderão ter que ser adaptadas para puderem ser utilizadas noutros sistemas, nos seus correspondentes dialetos SQL.

Créditos

Todos os elementos gráficos (imagens, gráficos, screenshots, etc.) presentes nesta unidade de notas de leitura são, obviamente, propriedade única e exclusiva dos seus respetivos autores (autores, empresas, editoras, distribuidores, etc.), tendo sido aqui utilizadas apenas como forma de ilustração. No processo de elaboração destas notas de leitura desenvolvemos os esforços necessários para referir e creditar da forma mais correta eventuais direitos de autor e seus detentores. A eventual omissão de um qualquer direito não foi intencional e será devidamente referido numa próxima revisão/edição destas mesmas notas.

MySQL é uma marca registrada da *Oracle Corporation* e/ou das suas afiliadas.

Estrutura da Unidade

1. Linguagens para Bases de Dados
2. A Linguagem SQL
3. Descrição de Dados
4. Manipulação de Dados (Inserção, Atualização e Remoção)
5. Manipulação de Dados (Consulta)
6. Controlo de Dados
7. Utilização de Vistas
8. Lista de Recursos

- Introdução
- As Linguagens

1

Linguagens para Base de Dados

Introdução

- O armazenamento de informação num sistema de dados pressupõe que, mais cedo ou mais tarde, de uma forma ou de outra, vamos precisar de a consultar ou mesmo de a modificar.
- Permitir aos utilizadores de um sistema de bases de dados meios para acederem de uma forma simples e intuitiva à informação que contêm é um dos seus principais objetivos.

Introdução

- Os sistemas de bases de dados costumam disponibilizar linguagens que permitem aos seus utilizadores expressarem os seus pedidos de informação, que são usualmente reconhecidas por *queries*.
- A linguagem mais "famosa" neste domínio é a SQL.

Ref.: Zemke, F., "What's new in SQL:2011". ACM SIGMOD Record 41.1 (2012): 67-73.
https://sigmodrecord.org/publications/sigmodRecord/1203/pdfs/10.industry.zemke.pdf

As Linguagens

- Após a formulação matemática do modelo relacional, álgebra relacional e cálculo relacional por E. Codd vários esforços foram feitos por entidades de investigação e comerciais para desenvolverem e implementarem linguagens relacionais para sistemas de computadores.
- Duas das mais famosas dessas linguagens foram a SQL - "Structured Query Language" – pronunciada como "ess-cue-ell" ou "sequel" – e a QBE - "Query-by-Example" -, ambas com origem na IBM durante os anos 70.
- Ambas as linguagens apresentam funcionalidades similares. Porém, a SQL é uma linguagem textual enquanto que a QBE é uma linguagem gráfica.

- Apresentação da Linguagem
- A Linguagem SQL
- Versões da SQL
- Vertentes de Atuação

2

A Linguagem SQL

Apresentação da Linguagem

- A SQL resultou de um projeto da IBM (1974) denominado inicialmente por “System R” e tornou-se parte do domínio público nos finais dos anos 70. Foi desenvolvida por Donald Chamberlin e Raymond Boyce.
- Em 1981 a IBM lançou a SQL/DS, um SGBD comercial que suportava a SQL, e em 1983 lançou a SQL como parte integrante do SGBD DB2.

A Linguagem SQL

- O sucesso da SQL provocou o aparecimento de diversos "dialectos" apresentados por outras empresas da área.
- A criação de um padrão (standard) para linguagem tornou-se pertinente e crucial para a utilização da própria linguagem e para o seu desenvolvimento.
- O standard da SQL foi definido pelo *American National Standards Institute* (ANSI) em 1986 e *International Organization for Standardization* (ISO) em 1987.

Ref.: Chapple, Mike. "SQL Fundamentals". Databases. About.com. https://www.thoughtco.com/sql-fundamentals-1019780

Versões da SQL

- O percurso evolutivo da SQL revelou-nos variadíssimas versões, em particular: SQL-86, SQL-89, SQL-92, SQL:1999, SQL:2003, SQL:2006, SQL:2008, e a SQL:2016. Esta última versão é a mais estável e, obviamente, mais atual.

A Sintaxe da SQL:2016

Ref.: ISO/IEC 9075-1:2016 - Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework).
https://www.iso.org/standard/63555.html

Vertentes de Atuação da SQL

- Através da linguagem SQL podemos realizar vários tipos de operações sobre uma base de dados, nomeadamente operações de:
  - Definição ou descrição de dados – *Data Description Language*.
  - Manipulação de dados – *Data Manipulation Language* – e Interrogação de dados – *Data Querying Language*.
  - Gestão, controlo e acesso de dados – *Data Control Language*.

Domínios

- Um domínio é um conjunto do qual um atributo de uma relação toma os seus valores.
- A SQL disponibiliza um conjunto de tipos de dados pré-definidos assim como permite que os utilizadores possam definir os seus próprios tipos de dados.
- Tipos de dados pré-definidos são considerados domínios.

Tipos de Dados

- A SQL permite trabalhar com diferentes tipos de dados, nomeadamente:
  - Números exatos:
    - INT[EGER], SMALLINT, NUMERIC [(p[,s])], DECIMAL[(p[,s])]
  - Números aproximados:
    - FLOAT, REAL, DOUBLE PRECISION
  - *Strings* e Caracteres:
    - CHAR[ACTER] [n], CHAR[ACTER] VARYING(n) ou VARCHAR(n), NATIONAL CHAR[ACTER] [n], NATIONAL CHAR[ACTER] VARYING(n)
  - Objectos de dados não estruturados de grande dimensão.
    - TEXT, CLOB, BLOB

Tipos de Dados

- Strings binárias:
  - BIT[(n)], BIT VARYING(n)
- Tempo e Datas
  - DATE, TIME, TIMESTAMP, TIME WITH TIME ZONE, TIMESTAMP TIME WITH TIME ZONE
- Outros tipos
  - BOOLEAN

Palavras Reservadas

- Apesar da SQL possuir um vocabulário considerado limitado, quando comparada com outras linguagens, a SQL tem, apesar de tudo, cerca de 300 palavras reservadas.
- Além disso, cada fabricante de sistemas de bases de dados acrescentou nos seus sistemas mais “algumas tantas” que utiliza em instruções ou em comandos disponibilizados exclusivamente nas suas implementações da SQL – os dialetos da SQL.

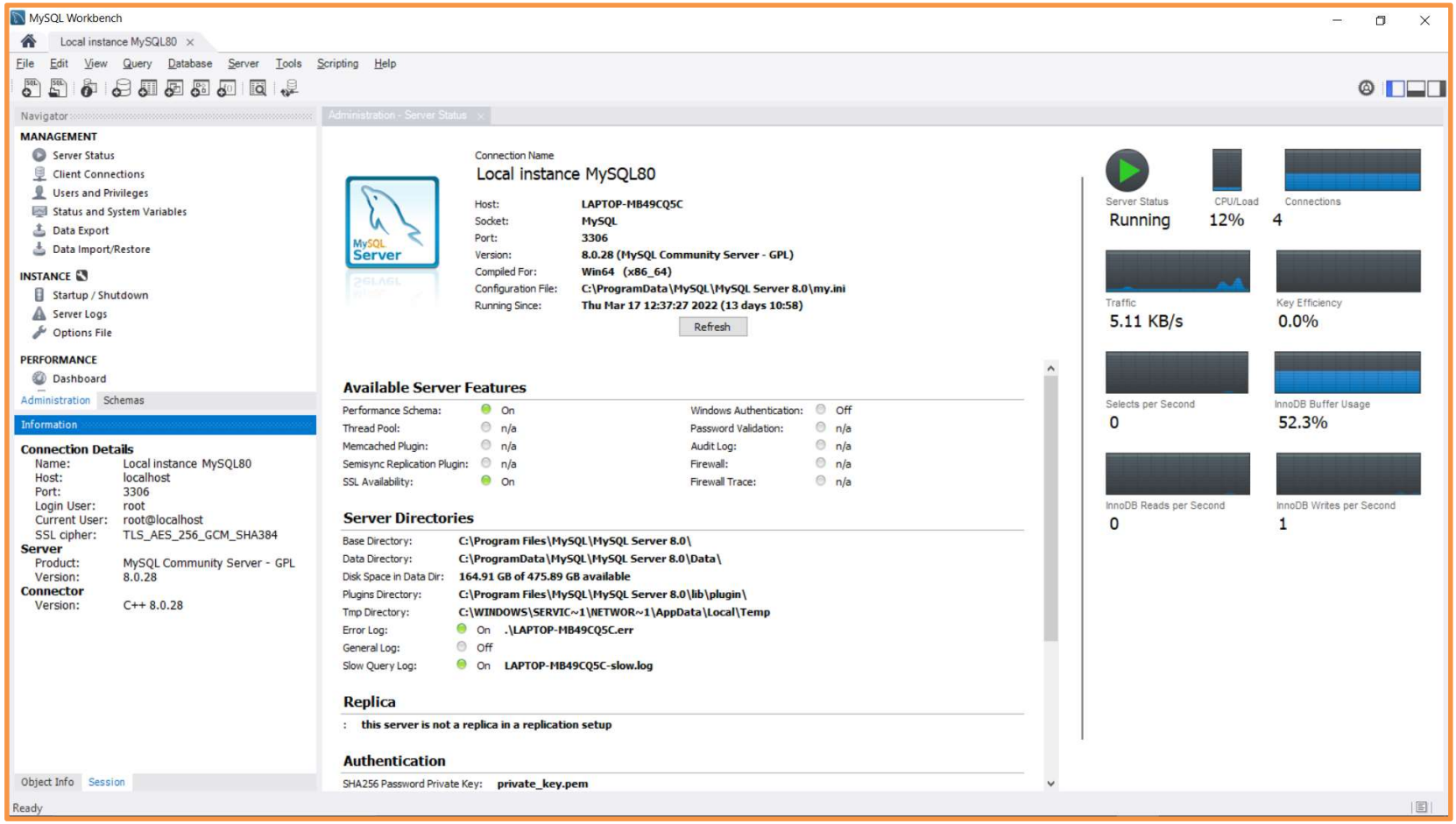
Palavras Reservadas

- Na definição de instruções deve-se ter o cuidado de respeitar as palavras reservadas da SQL, por exemplo:
  - ACTION, AGGREGATE, BEFORE, BOOLEAN, CASCADE, CASE, CONNECT, DELETE, CUBE, COLLATION, NUMERIC, LEVEL, DATA, DAY, OLD, ON, LOOP, MINUTE, SUM, WITH, SELECT, VALUE, USING, ROLE, USER, PUBLIC, SIZE, REFERENCES, OUTPUT, CATALOG, DROP, YEAR, ...

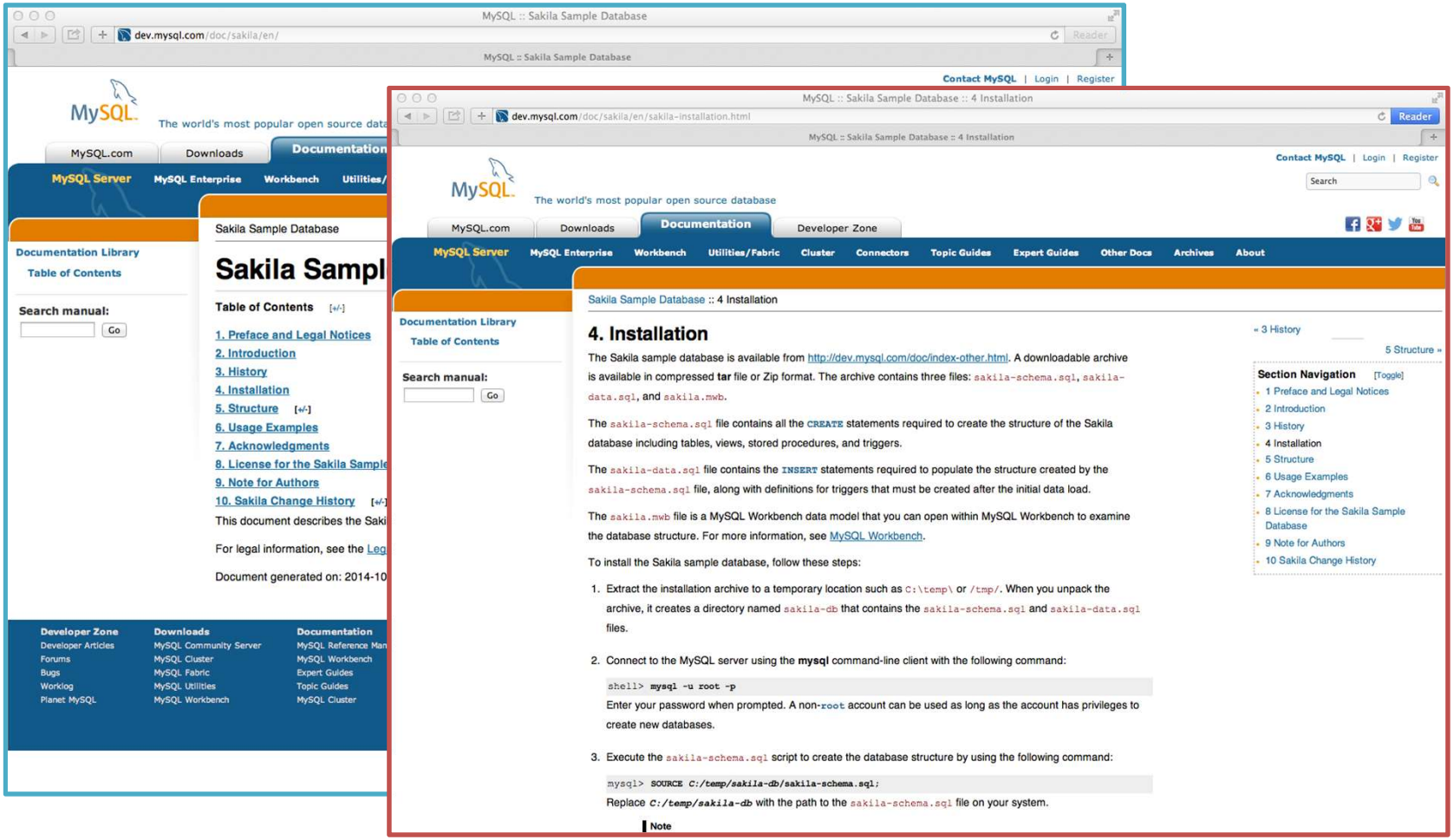
O Sistema e a Base de Dados

- Para suportarmos a exposição e demonstração das várias instruções da SQL (e suas variantes) vamos utilizar como ferramentas base:
  - Sistema de Gestão de Bases de dados (http://dev.mysql.com)
    - MySQL 8.0.28, MySQL Community Server - GPL).
    - MySQL Workbench 8.0.28 Community Edition.
  - Bases de Dados de Trabalho (https://dev.mysql.com/doc/sakila/en/)
    - Sakila (Sakila.mwb, SakilBD.sql e SakilaDt.sql)

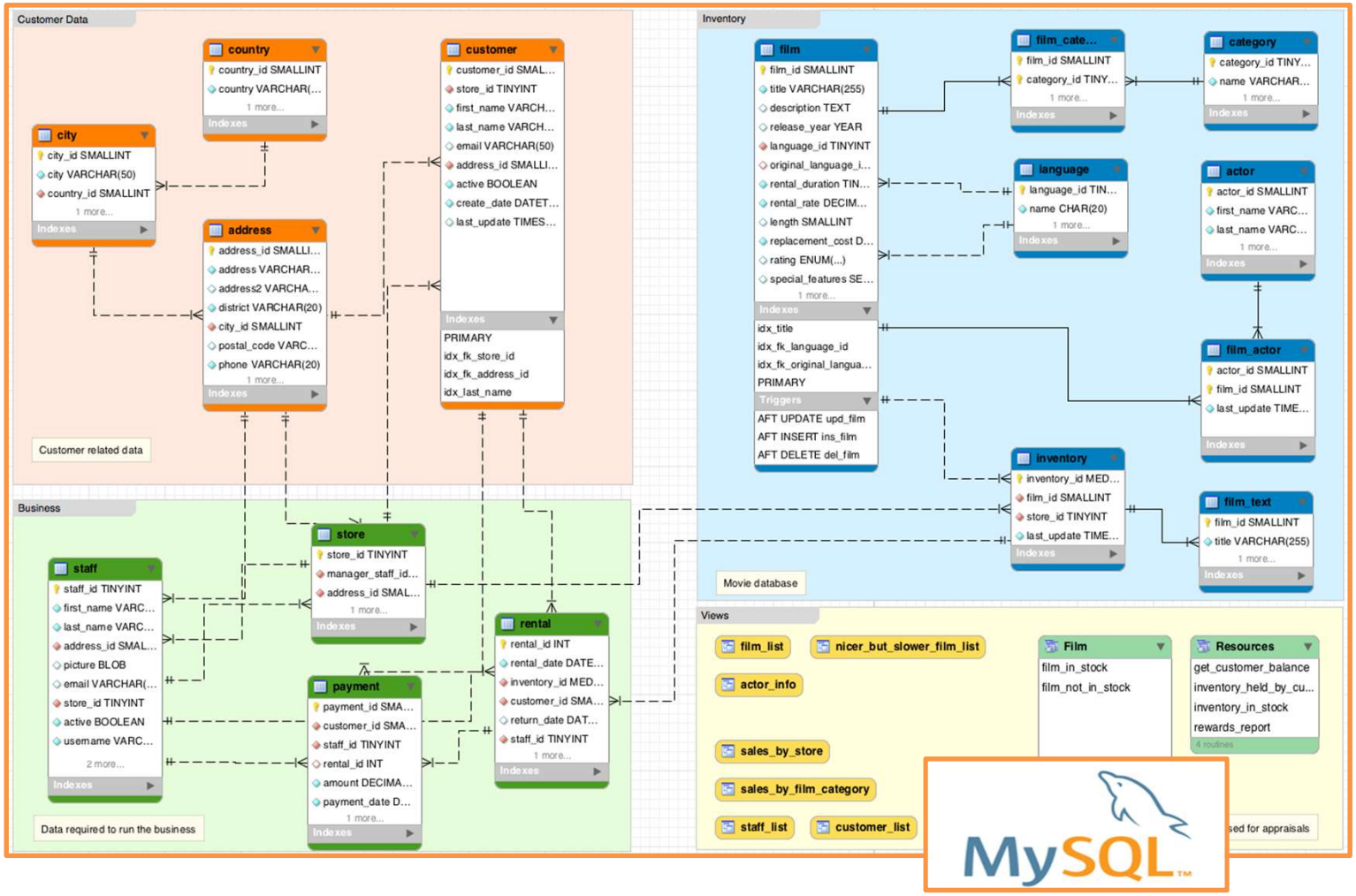
O Motor de Bases de Dados



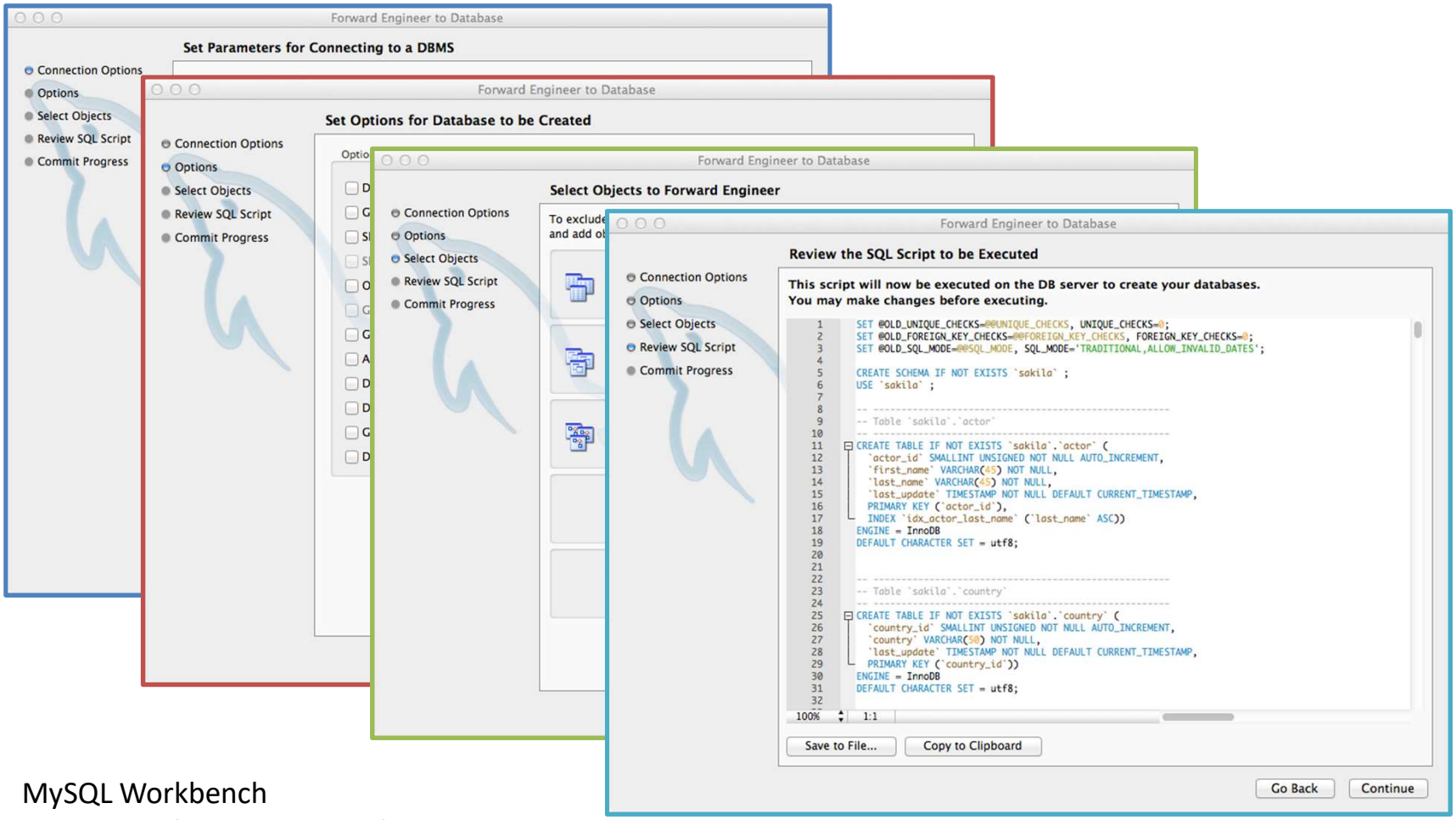
A Base de Dados de Trabalho



O Esquema da Base de Dados



Preparação da Base de Dados



MySQL Workbench
Menu: Database >> Forward Engineer...

Extrato de um Script de Criação

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

CREATE SCHEMA IF NOT EXISTS `sakila` ;
USE `sakila` ;

-- -----------------------------------------------------
-- Table `sakila`.`actor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `sakila`.`actor` (
  `actor_id` SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(45) NOT NULL,
  `last_name` VARCHAR(45) NOT NULL,
  `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`actor_id`),
  INDEX `idx_actor_last_name` (`last_name` ASC))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `sakila`.`country`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `sakila`.`country` (
  `country_id` SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `country` VARCHAR(50) NOT NULL,
  `last_update` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`country_id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

(...)
```

Criação da base de dados

Criação de uma tabela

Apenas um pequeno excerto do script SQL gerado para a criação da base de dados 'Sakila'.

Instalação da Base de Dados “Sakila”

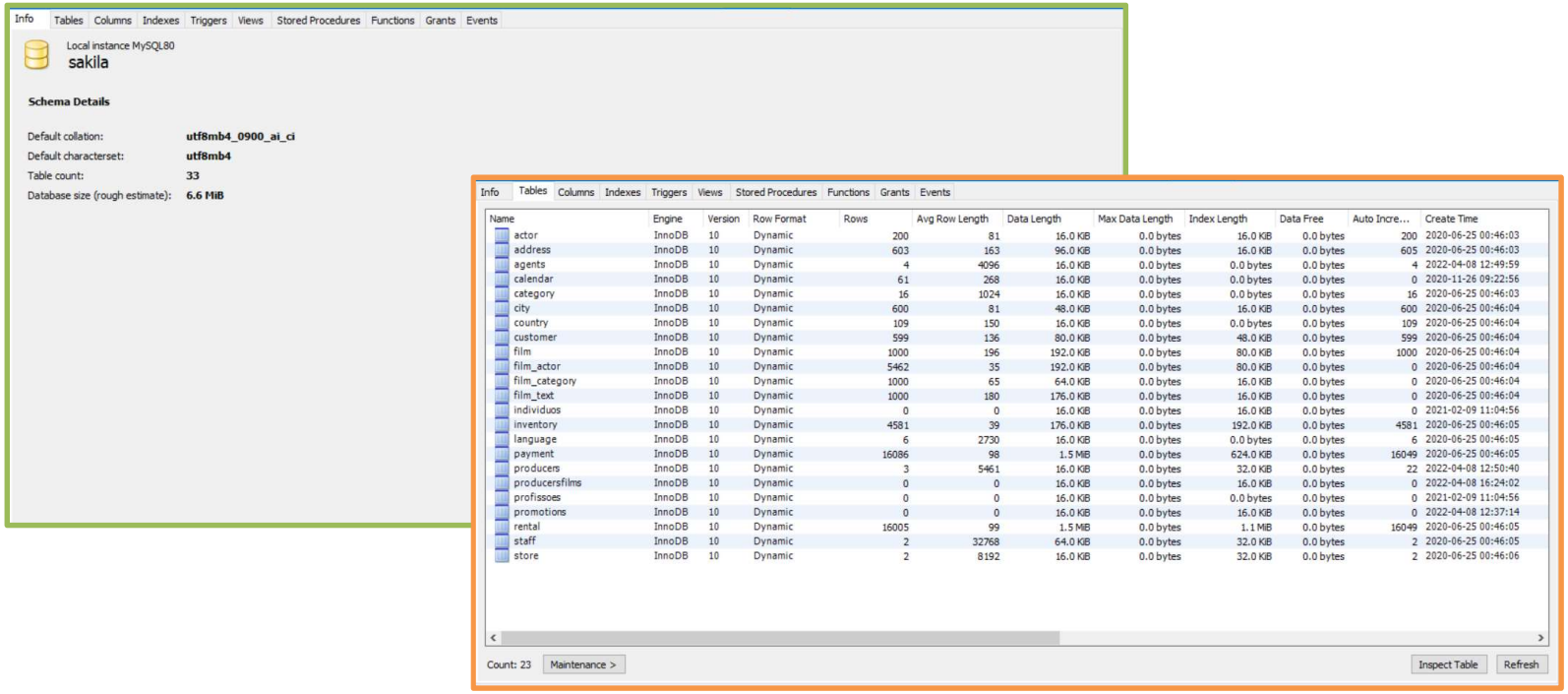
- O processo de instalação da base de dados “Sakila” é bastante simples. O processo de instalação está descrito no endereço:
  - https://dev.mysql.com/doc/sakila/en/sakila-installation.html’

Instalação da Base de Dados “Sakila”

- Alternativamente, podemos instalar a base de dados “Sakila” realizando os seguintes passos:
  - Descarregar o ficheiro zip da “Skila Database” que está disponível no endereço https://dev.mysql.com/doc/index-other.html.
  - Extrair dos ficheiro descarregado os ficheiros sakila-schema.sql (criação da base de dados) e sakila-data.sql (povoamento da base de dados)
  - Carregar cada um desses ficheiros para o ambiente de administração do MySQL Workbench, colocando cada um deles num painel de edição distinto.
  - Executar, em primeiro lugar, o ficheiro de criação do esquema da base de dados e depois, caso não ocorra nenhum erro, o ficheiro de povoamento da base de dados.

Instalação da Base de Dados “Sakila”

- Verificar a disponibilidade da base de dados “Sakila” através da opção ‘Schema Inspector’ disponível a partir do menu de comandos do painel de navegação, que é ativo com o botão direito do rato.



- Bases de Dados
- Tabelas
- Atributos
- Domínios
- Restrições

3

Descrição de Dados (DDL)

Descrição de Dados

- A criação (e atualização) dos objetos de uma base de dados – tabelas, vistas, stored procedures, triggers, etc. é realizada com um conjunto de comandos específicos da SQL que se enquadram na sua vertente conhecida como "Descrição de dados" – Data Description Language.
- Os comandos mais usuais nesta vertente de trabalho servem para criar, atualizar e remover objetos e são, respetivamente:
  - CREATE <Tipo de Objeto> <Nome do Objeto> (...).
  - ALTER <Tipo de Objeto> <Nome do Objeto> (...).
  - DROP <Tipo de Objeto> <Nome do Objeto> (...).

Descrição de Dados

```
CREATE TABLE IF NOT EXISTS `sakila`.`actor` (
  `actor_id` SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(45) NOT NULL,
  `last_name` VARCHAR(45) NOT NULL,
  `last_update` TIMESTAMP NOT NULL DEFAUL
  PRIMARY KEY
  INDEX `idx_
ENGINE = Inn
DEFAULT CHARA
```

```
CREATE TABLE IF NOT EXISTS
  `store_id` TINYINT UNSIG
  `manager_staff_id` TINYI
  `address_id` SMALLINT UN
  `last_update` TIMESTAMP
  PRIMARY KEY (`store_id`)
  UNIQUE INDEX `idx_unique
  INDEX `idx_fk_address_id
  CONSTRAINT `fk_store_sta
```

```
CREATE FUNCTION `sakila`.`inventory_in_stock`(p_inventory_id INT) RETURNS BOOLEAN
READS SQL DATA
BEGIN
    DECLARE v_rentals INT;
    DECLARE v_out     INT;

    #AN ITEM IS IN-STOCK IF THERE ARE EITHER NO ROWS IN THE rental TABLE
    #FOR THE ITEM OR ALL ROWS HAVE return_date POPULATED

    SELECT COUNT(*) INTO v_rentals
    FROM rental
    WHERE inventory_id = p_inventory_id;

    IF v_rentals = 0 THEN
```

```
DELIMITER //
-- DROP PROCEDURE IF EXISTS povoaDimCalendario();
CREATE PROCEDURE povoaDimCalendario()
    BEGIN
        SET @DataRef1 := (SELECT MIN(payme
        SET @DataRef2 := (SELECT MAX(payme
        L1: WHILE @DataRef1 <= @DataRef2 D

            INSERT INTO SakilaDW.DimCalend
                SELECT
                    DATE(@DataRef1) AS "Da
                    WEEK(@DataRef1) AS "Se
                    DAYNAME(@DataRef1) AS
                    MONTHNAME(@DataRef1) A
                    QUARTER(@DataRef1) AS
                    NULL AS "Semestre",
                    YEAR(@DataRef1) AS "An
            SET @DataRef1 := ADDDATE(@Data
        END WHILE L1;
    END//
DELIMITER ;
```

```
CREATE  OR REPLACE VIEW sales_by_store
AS
SELECT
CONCAT(c.city, _utf8',', cy.country) AS store
, CONCAT(m.first_name, _utf8' ', m.last_name) AS manager
, SUM(p.amount) AS total_sales
FROM payment AS p
INNER JOIN rental AS r ON p.rental_id = r.rental_id
INNER JOIN inventory AS i ON r.inventory_id = i.inventory_id
INNER JOIN store AS s ON i.store_id = s.store_id
INNER JOIN address AS a ON s.address_id = a.address_id
INNER JOIN city AS c ON a.city_id = c.city_id
INNER JOIN country AS cy ON c.country_id = cy.country_id
INNER JOIN staff AS m ON s.manager_staff_id = m.staff_id
GROUP BY s.store_id
ORDER BY cy.country, c.city;
```

Bases de Dados

- As bases de dados precisam de um local, de um repositório para se instalarem e receberem posteriormente as suas tabelas, dados e restrições. Para se criar e remover uma base de dados de um sistema utilizam-se, respetivamente as seguintes instruções:
  - CREATE DATABASE – criação de uma base de dados de um sistema.
  - DROP DATABASE – remoção de uma base de dados de um sistema.

CREATE DATABASE - Estrutura

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_option] ...
    create_option: [DEFAULT] {
     CHARACTER SET [=] charset_name
    | COLLATE [=] collation_name
    | ENCRYPTION [=] {'Y' | 'N'}}
```

Fonte: MySQL 8.0 Reference Manual, CREATE DATABASE Statement
https://dev.mysql.com/doc/refman/8.0/en/create-database.html

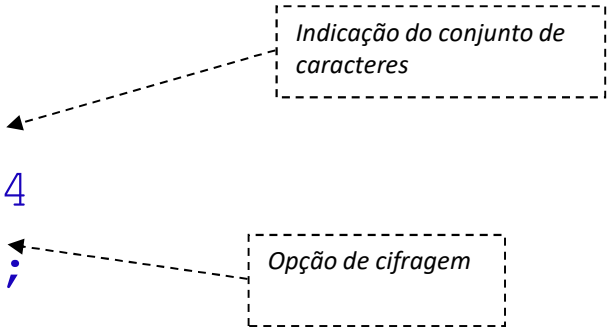
CREATE DATABASE - Exemplos

- Criação de uma base de dados (“Sakila”):

```
CREATE SCHEMA Sakila;
```

- Criação de uma base de dados (“Sakila”) com opções de configuração:

```
CREATE DATABASE Sakila
    DEFAULT CHARSET=utf8mb4
    DEFAULT ENCRYPTION='N';
```



DROP DATABASE - Estrutura

DROP {DATABASE | SCHEMA} [IF EXISTS] db_name

Fonte: MySQL 8.0 Reference Manual, DROP DATABASE Statement
https://dev.mysql.com/doc/refman/8.0/en/drop-database.html

DROP DATABASE - Exemplos

- Remoção de uma base de dados (“Sakila”):

```
DROP DATABASE Sakila;
```

Tabelas

- Nos sistemas de bases de dados relacionais, a informação é guardada em tabelas (ou relações). As operações de descrição de dados sobre tabelas são:
  - CREATE TABLE – criação do esquema de uma tabela na base de dados.
  - ALTER TABLE – alteração do esquema de uma tabela da base de dados.
  - DROP TABLE – remoção de uma tabela da base de dados.

Criação de Tabelas

- O processo de criação de uma tabela não é complicado. Contudo requer alguns cuidados na definição dos seus vários elementos de dados e restrições.
- Em SQL, a criação de uma tabela é realizada com a instrução CREATE TABLE.

CREATE TABLE - Estrutura

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...)
    [table_options]
    [partition_options]
column_definition: { data_type [NOT NULL | NULL] [DEFAULT
{literal | (expr)} ] [VISIBLE | INVISIBLE] [AUTO_INCREMENT]
[UNIQUE [KEY]] [[PRIMARY] KEY] [COMMENT 'string'] [COLLATE
collation_name] [COLUMN_FORMAT {FIXED | DYNAMIC |
DEFAULT}] [ENGINE_ATTRIBUTE [=] 'string']
[SECONDARY_ENGINE_ATTRIBUTE [=] 'string'] [STORAGE {DISK |       MEMORY}]
(...)
```

Obs.: Apenas está apresentada uma parte da estrutura da instrução

Fonte: MySQL 8.0 Reference Manual, CREATE TABLE Statement
https://dev.mysql.com/doc/refman/8.0/en/create-table.html

CREATE TABLE - Exemplos

- Criação de uma tabela:

```
CREATE TABLE Agents (
    Agent_Id INT NOT NULL,
    First_Name VARCHAR(30) NOT NULL,
    Last_Name VARCHAR(30) NOT NULL,
        PRIMARY KEY (Agent_Id);
```

Definição do domínio. Tipo de dados.

Definição de obrigatoriedade de valor

- Criação de uma tabela com um atributo autonumerado:

```
CREATE TABLE Agents (
    Agent_Id INT AUTO_INCREMENT,
    First_Name VARCHAR(30) NOT NULL,
    Last_Name VARCHAR(30) NOT NULL,
        PRIMARY KEY (Agent_Id));
```

Definição de numeração automática

Definição de chave primária

CREATE TABLE - Exemplos

- Criação de uma tabela com definição de chave primária e uma chave estrangeira:

```
CREATE TABLE Promotions (
    Promotion_Nr INT NOT NULL,
    Description VARCHAR(50) NOT NULL,
    Initial_Date DATETIME,
    Final_Date DATETIME,
    Category_Id TINYINT UNSIGNED NOT NULL,
        PRIMARY KEY (Promotion_Nr),
        FOREIGN KEY (Category_Id)
            REFERENCES Category (Category_Id));
```

Não permite valores negativos

Definição de chave primária

Chave de chave estrangeira

CREATE TABLE - Exemplos

- Criação de uma tabela seguida da alteração da sua definição inicial para a introdução de uma chave estrangeira:

```
CREATE TABLE Promotions (
    Promotion_Nr INT NOT NULL,
    Description VARCHAR(50) NOT NULL,
    Initial_Date DATETIME,
    Final_Date DATETIME,
    Category_Id TINYINT UNSIGNED NOT NULL,
        PRIMARY KEY (Promotion_Nr));
ALTER TABLE Promotions
    ADD CONSTRAINT fkPromotionsCategories
        FOREIGN KEY (Category_Id)
        REFERENCES Category (Category_Id);
```

Definição de chave primária

Definição de chave estrangeira

CREATE TABLE - Exemplos

- Criação de uma tabela com definição de restrições ao nível do atributo:

```
CREATE TABLE IF NOT EXISTS Producers (
    Producer_Id INT NOT NULL AUTO_INCREMENT,
    First_Name VARCHAR(30) NOT NULL,
    Last_Name VARCHAR(30) NOT NULL,
    Gender CHAR(1) NOT NULL CHECK (Gender IN ('M', 'F')),
    Country_Id SMALLINT UNSIGNED NULL,
    Date_Of_Birth DATE NULL,
    eMail VARCHAR(150) UNIQUE,
    Facebook VARCHAR(200) NULL,
    Instagram VARCHAR(200) NULL,
    Notes TEXT NULL,
        PRIMARY KEY (Producer_Id) ,
        FOREIGN KEY (Country_Id)
            REFERENCES Country (Country_Id));
```

Definição de numeração automática

Definição de uma restrição de utilização de valores

Não permite valores repetidos

```
39  12:50:40  CREATE TABLE IF NOT EXISTS Producers ( Producer_Id INT NOT NULL AUTO_INCREMENT,...  0 row(s) affected
```

CREATE TABLE - Exemplos

- Criação de uma tabela com a definição de uma chave primária composta e duas chaves estrangeiras:

```
CREATE TABLE IF NOT EXISTS ProducersFilms (
    Producer_Id INT NOT NULL,
    Film_Id SMALLINT UNSIGNED NOT NULL,
        PRIMARY KEY (Producer_Id, Film_Id),
        FOREIGN KEY (Producer_Id)
            REFERENCES Producers (Producer_Id),
        FOREIGN KEY (Film_Id)
            REFERENCES Film (Film_Id));
```

Definição de uma chave primária composta por dois atributos.

Definição das chaves estrangeiras.

Alteração de Tabelas

- Após se ter criado uma tabela é possível modificar as suas características base e as opções assumidas, nomeadamente:
  - Acrescentar, modificar ou remover atributos da tabela, assim como as propriedades dos atributos – nome, tipo de dados, precisão, etc.
  - Acrescentar ou remover restrições sobre as chaves primária e estrangeiras.
  - Acrescentar ou remover restrições sobre as definições UNIQUE e CHECH, e definições DEFAULT.
  - Acrescentar ou remover identificadores de atributos.

ALTER TABLE - Estrutura

```
ALTER TABLE tbl_name
        [alter_option [, alter_option] ...]
        [partition_options]

    alter_option: {
    table_options
    | ADD [COLUMN] col_name column_definition
            [FIRST | AFTER col_name]
    | ADD [COLUMN] (col_name column_definition,...)
    | ADD {INDEX | KEY} [index_name]
            [index_type] (key_part,...) [index_option] ...
    | ADD {FULLTEXT | SPATIAL} [INDEX | KEY] [index_name]
            (key_part,...) [index_option] ...
    | ADD [CONSTRAINT [symbol]] PRIMARY KEY
            [index_type] (key_part,...)
            [index_option] ...
(...)
```

Obs.: Apenas está apresentada uma parte da estrutura da instrução

Fonte: MySQL 8.0 Reference Manual, ALTER TABLE Statement
https://dev.mysql.com/doc/refman/8.0/en/alter-table.html

ALTER TABLE - Exemplos

- Adição de um novo atributo à definição de uma tabela:

```
ALTER TABLE Agents
    ADD Business_Terms VARCHAR(250) NULL;
```

- Adição de novos atributos a uma tabela com indicação de posição:

```
ALTER TABLE Producers
    ADD Royalties DECIMAL(10,2) NOT NULL DEFAULT 0
        AFTER Instagram,
    ADD Contract TEXT
        FIRST;
```

ALTER TABLE - Exemplos

- Remoção de um atributo de uma tabela:

```
ALTER TABLE Agents
    DROP COLUMN Business_Terms;
```

- Modificação da definição de vários atributos de uma tabela:

```
ALTER TABLE Producers
        MODIFY Contract VARCHAR(250) NOT NULL,
        MODIFY Date_Of_Birth DATE NULL AFTER Last_Name;
```

Remoção de Tabelas

- A remoção de uma tabela da base de dados do sistema é realizada com a instrução DROP TABLE.
- Quando se aplica sobre a base de dados uma instrução de remoção de uma tabela, além de se remover da base de dados a sua definição também se removem todos:
  - Registos;
  - Índices e gatilhos;
  - Restrições (*constraints*);
  - Privilégios;
  - (…)
- Caso uma tabela contenha uma chave estrangeira, esta só poderá ser removida da base de dados após a remoção da definição da chave estrangeira da tabela.

DROP TABLE - Estrutura

```
DROP [TEMPORARY] TABLE [IF EXISTS]
        tbl_name [, tbl_name] ...
        [RESTRICT | CASCADE]
```

Fonte: MySQL 8.0 Reference Manual, DROP TABLE Statement
https://dev.mysql.com/doc/refman/8.0/en/drop-table.html

DROP TABLE - Exemplos

- Remoção da tabela “Agents” do sistema:

```
DROP TABLE IF EXISTS Agents;
```

- Remoção simultânea de duas tabelas:

```
DROP TABLE ProducersFilms, Producers;
```

- Remoção de uma tabela temporária:

```
DROP TEMPORARY TABLE tmpBestFilms;
```

Consulta de Metadados

- Todos os objetos que são criados ao longo do tempo numa base de dados possuem um conjunto de metadados que descrevem as suas propriedades.
- Para acedermos aos metadados de uma tabela que foi criada no sistema podemos utilizar a instrução DESC.

DESC - Exemplos

- Consulta dos metadados relativos à tabela “Producers”:

```
DESC Producers;
```

Field	Type	Null	Key	Default	Extra
Producer_Id	int	NO	PRI	NULL	auto_increment
First_Name	varchar(30)	NO		NULL	
Last_Name	varchar(30)	NO		NULL	
Gender	char(1)	NO		NULL	
Country_Id	smallint unsigned	YES	MUL	NULL	
Date_Of_Birth	date	YES		NULL	
eMail	varchar(150)	YES	UNI	NULL	
Facebook	varchar(200)	YES		NULL	
Instagram	varchar(200)	YES		NULL	
Notes	text	YES		NULL	

Descrição de Dados (DDL)

- Base de Dados Exemplo
  - **Sakila -** https://dev.mysql.com/doc/sakila/en/sakila-installation.html'
- Ferramentas
  - **MySQL Workbench -** https://dev.mysql.com/downloads/workbench/
  - **MySQL Community Server -** https://dev.mysql.com/downloads/mysql/
  - **MySQL Command-Line Client**

Exercícios

- Modificação de Dados.
- Remoção de Dados.

4

Manipulação de Dados (DML)

Atualização

Manipulação de Dados

- Na vertente de manipulação de dados da SQL podemos realizar operações de inserção de dados, para povoamento da base de dados, e de modificação e remoção de dados, para atualização dos objetos da base de dados.
- Para realizar as operações de inserção de dados podemos utilizar a instrução INSERT, para as operações de modificação de dados a instrução UPDATE e, por fim, para as operações de remoção DELETE.

INSERT - Estrutura

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name [, partition_name] ...)]
    [(col_name [, col_name] ...)]
    { {VALUES | VALUE} (value_list) [, (value_list)] ... }
    [AS row_alias[(col_alias [, col_alias] ...)]]
    [ON DUPLICATE KEY UPDATE assignment_list]
(...)
```

Obs.: Apenas está apresentada uma parte da estrutura da instrução

Fonte: MySQL 8.0 Reference Manual, INSERT Statement
https://dev.mysql.com/doc/refman/8.0/en/insert.html

INSERT - Exemplos

- Inserção de um registo numa tabela:

```
INSERT INTO Producers
    (Producer_Id, First_Name, Last_Name, Gender,
Country_Id,Date_Of_Birth,
    eMail, Facebook, Instagram, Notes)
    VALUES ('21','Jonh','Well-Done','M','103','1978-01-01',
    'jonhw@email.com',NULL,NULL,NULL);
```

- Inserção de um registo numa tabela, utilizando apenas uma parte do seu esquema da tabela:

```
INSERT INTO Producers
    (Producer_Id, First_Name, Last_Name, Gender)
    VALUES ('22','Anne','Carson','F');
```

INSERT - Exemplos

- Inserção de um registo numa tabela, sem indicação do seu esquema de dados:

```
INSERT INTO Agents
    VALUES ('1','Mary','Smith-James',NULL);
```

- Inserção simultânea de vários registos:

```
INSERT INTO Agents
    (Agent_Id, First_Name, Last_Name)
    VALUES
        ('2','Ralf','Spencer'),
        ('3','Anne','Lopez'),
        ('4','Joseph','Pane');
```

UPDATE - Estrutura

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
        SET assignment_list
        [WHERE where_condition]
        [ORDER BY ...]
        [LIMIT row_count]
        value:
            {expr | DEFAULT}
        assignment:
            col_name = value
        assignment_list:
            assignment [, assignment] ...
```

Fonte: MySQL 8.0 Reference Manual, UPDATE Statement
https://dev.mysql.com/doc/refman/8.0/en/update.html

UPDATE - Exemplos

- Modificação de todos registos de uma tabela:

```
UPDATE Agents
    SET Business_Terms = NULL;
```

- Modificação de dois registos de uma tabela com aplicação de critério de filtragem:

```
UPDATE Agents
    SET Business_Terms = '1 year contract'
    WHERE Agent_Id IN ('1','2');
```

DELETE - Estrutura

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name [[AS] tbl_alias]
    [PARTITION (partition_name [, partition_name] ...)]
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

Fonte: MySQL 8.0 Reference Manual, DELETE Statement
https://dev.mysql.com/doc/refman/8.0/en/delete.html

DELETE - Exemplos de Aplicação

- Remoção de um registo específico de uma tabela:

```
DELETE FROM Agents
    WHERE Agent_Id = '1';
```

- Remoção de vários registos com aplicação de critério de filtragem:

```
DELETE FROM Producers
    WHERE YEAR(Date_Of_Birth) < '2000';
```

- Remoção de todos os registos de uma tabela:

```
DELETE FROM ProducersFilms;
```

```
-- Desativar modo de segurança
SET SQL_SAFE_UPDATES=0;
-- Ativar modo de segurança
SET SQL_SAFE_UPDATES=1;
```

Inserção com Substituição

- Usualmente, a inserção de registos em tabelas de uma base de dados é realizada através da instrução INSERT.
- Por vezes, numa operação de inserção de dados ocorrem erros, reportando, por exemplo, a duplicação de um valor de uma chave primária, o que faz terminar o processo de inserção de dados.
- Em alguns casos específicos, essa situação pode ser evitada isso, fazendo com que o registo que provocou a situação de violação de chave primária fosse substituído pelo "novo" registo. Isso pode ser feito utilizando-se a instrução REPLACE.

A Instrução REPLACE

- O MySQL disponibiliza-nos a instrução REPLACE, que nos permite fazer a atualização dos dados relativos a um registo (ou mais) que já exista na base de dados durante um processo de inserção.
- Esta instrução não faz parte da especificação padrão da SQL. É uma extensão específica do MySQL.
- A instrução REPLACE atua de forma muito semelhante à instrução INSERT, distinguindo-se apenas quando a tabela alvo tem uma chave primária ou um índice único definido.

REPLACE - Exemplos

- Inserção de dois registos e substituição de um registo já existente (Producer_Id = '21'):

```
REPLACE INTO Producers
    (Producer_Id, First_Name, Last_Name, Gender, Country_Id,
    Date_Of_Birth, eMail, Facebook, Instagram, Notes)
    VALUES
        ('20','Stephen','Joker','M','102','1990-02-12',
        'joker@email.com',NULL,NULL,NULL),
        ('21','Jonh','Well-Done','M','102','1978-01-01',
        'jonhw@email.com',NULL,NULL,NULL),
        ('22','Star','Bucks','M','103','1987-06-21',
        'star@email.com',NULL,NULL,NULL);
```

REPLACE - Exemplos

- Atualização de um registo numa tabela:

```
REPLACE INTO Agents
    SET Agent_Id = '2', First_Name = 'Ralf',
    Last_Name = 'Spencer',
    Business_Terms = 'Free Lancer';
```

Manipulação de Dados (DDL)

- Base de Dados Exemplo
  - **Sakila -** https://dev.mysql.com/doc/sakila/en/sakila-installation.html'
- Ferramentas
  - **MySQL Workbench -** https://dev.mysql.com/downloads/workbench/
  - **MySQL Community Server -** https://dev.mysql.com/downloads/mysql/
  - **MySQL Command-Line Client**

Exercícios 

- Interrogações simples.
- Interrogações complexas.
- Subqueries.
- Junções internas e externas.
- Operações de conjuntos.

5

Manipulação de Dados (DML)

Interrogação (*Querying*)

Consultas de Dados

- Após o povoamento inicial de uma base de dados é possível desenvolver processos de consulta de dados para conhecermos aquilo que está contido numa ou noutra tabela.
- Em SQL os processos de consulta de dados são suportados pela instrução SELECT, "provavelmente" a instrução mais poderosa e versátil da linguagem.

SELECT - Estrutura

```
SELECT
        [ALL | DISTINCT | DISTINCTROW ]
        [HIGH_PRIORITY]
        [STRAIGHT_JOIN]
        [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
        [SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
        select_expr [, select_expr] ...
        [into_option]
        [FROM table_references
                [PARTITION partition_list]]
        [WHERE where_condition]
        [GROUP BY {col_name | expr | position}, ... [WITH ROLLUP]]
        [HAVING where_condition]
        [WINDOW window_name AS (window_spec)
                [, window_name AS (window_spec)] ...]
        [ORDER BY {col_name | expr | position}
                [ASC | DESC], ... [WITH ROLLUP]]
        [LIMIT {[offset,] row_count | row_count OFFSET offset}]
        [into_option]
    (…)
```

Obs.: Apenas está apresentada uma parte da estrutura da instrução

Fonte: MySQL 8.0 Reference Manual, SELECT Statement
https://dev.mysql.com/doc/refman/8.0/en/select.html

SELECT - Exemplos

- Consulta de todo o conteúdo (todos os registos) de uma dada tabela:

```
SELECT *
    FROM film;
```

- Consulta do conteúdo de uma tabela, segundo uma dada lista de atributos:

```
SELECT film_id, title, rental_rate
    FROM film;
```

como alternativa, com outra ordem na apresentação dos atributos, podemos ter:

```
SELECT title, film_id, rental_rate
    FROM film;
```

SELECT - Exemplos

- Batizando os atributos no conjunto de resultados:

```
SELECT title AS "Título",
    rental_rate AS "Tx Aluguer”
    FROM film;
```

- Realização de uma pequena ação de cosmética:

```
SELECT title AS "Título",
    rental_rate AS "Tx Aluguer",'______' AS "Coisas"
    FROM film;
```

- Visualização do conteúdo de uma tabela apresentando apenas os valores de um dos atributos, com eliminação de registos repetidos:

```
SELECT DISTINCT
    release_year AS "Ano de Lançamento”
    FROM film;
```

SELECT - Exemplos

- Visualização de toda a informação relativa ao filme ‘1’. Introdução de critérios de filtragem.

```
SELECT *
    FROM Film
    WHERE Film_Id = 1;
```

- Visualização da informação dos filmes com linguagens ‘1’ e ‘2’, cuja duração de aluguer não seja inferior a 6 dias:

```
SELECT *
    FROM Film
    WHERE (Language_Id = 1 OR Language_Id = 2 )
        AND Rental_Duration > 6;
```

SELECT - Exemplos

- Visualização da informação dos funcionários, clientes e datas de devolução de alugueres que sejam superiores a '2005-05-26':

```
SELECT Staff_Id, Customer_Id, Return_Date
    FROM Rental
    WHERE Return_Date >= '2005-05-26';
```

- Visualização da informação dos funcionários, clientes e datas de devolução de alugueres que sejam superiores e iguais a '2005-05-26‘ e inferiores ou iguais a '2005-06-01':

```
SELECT Staff_Id, Customer_Id, Return_Date
    FROM Rental
    WHERE Return_Date >= '2005-05-26'
       AND Return_Date <= '2005-06-01';
```

SELECT - Exemplos

- Visualização do identificador, título do filme e taxa de aluguer de todos os filmes, com exceção dos filmes com os identificadores ‘1’ ou ‘2’.

```
SELECT *
    FROM Film
    WHERE Film_Id NOT IN (1,2);
```

- Alternativamente, poderia ser:

```
SELECT Title, Film_Id, Rental_Rate
    FROM Film
    WHERE Film_Id != 1 AND Film_Id != 2;
```

SELECT - Exemplos

- Visualização do identificador, nome e duração do filme (length), cujo atributo “original_language” não contenha um valor nulo.

```
SELECT  title, film_id, length, original_language_id
   FROM film
   WHERE original_language_id IS NOT NULL;
```

- Visualização dos títulos, descrição e duração dos filmes lançados em ‘2006’ que tenham uma duração entre ’60’ e ‘120’:

```
SELECT title, description, length
   FROM film
   WHERE length BETWEEN 60 AND 120
       AND release_year = 2006;
```

SELECT - Exemplos

- Ordenando a query anterior por duração do filme de forma decrescente.

```
SELECT title, description, length
    FROM film
    WHERE length BETWEEN 60 AND 120
        AND release_year = 2006
    ORDER BY length ASC;
```

- e agora apresentando apenas os dez filmes mais demorados.

```
SELECT title, length
    FROM film
    ORDER BY length DESC
    LIMIT 10;
```

SELECT – Outras Aplicações

- Visualização de um número:

```
SELECT 1;
```

- Fazendo o cálculo de uma expressão numérica:

```
SELECT 1 + (2 * 3) / 4;
```

- Visualizando uma string:

```
SELECT 'Olá Mundo!';
```

SELECT - Utilização de Funções

- Visualização da data e hora atuais do sistema:

```
SELECT NOW();
```

- Visualização do valor da constante matemática pi:

```
SELECT PI();
```

- Visualização dos valores da aplicação de funções trignométricas:

```
SELECT SIN(100), COS(90), TAN(PI());
```

- Conversão de valores em graus e radianos:

```
SELECT RADIANS(120);
SELECT COS(RADIANS(90));
SELECT DEGREES(90);
```

SELECT - Datas e Horas

- Conversão da data atual nos seus respetivos dia da semana, dia, mês, ano e semana:

```
SELECT DAYNAME(NOW()), DAY(NOW()), MONTH(NOW()),
    YEAR(NOW()), WEEK(NOW());
```

- Obtenção da hora atual do sistema:

```
SELECT CURTIME();
```

- Obtenção da data atual do sistema:

```
SELECT CURDATE();
```

- Realização de uma operação de diferença entre dois elementos de dados do tipo DATETIME:

```
SELECT NOW(), Return_Date, DATEDIFF(NOW(), Return_Date)
    FROM Rental;
```

SELECT - Strings

- Concatenação de duas strings, com utilização de um separador:

```
SELECT customer_id, first_name, last_name,
    CONCAT_WS(' ',first_name,last_name) AS "Nome Completo"
    FROM customer
    ORDER BY CONCAT_WS(' ',first_name,last_name);
```

- Conversão para letras maiúsculas:

```
SELECT film_id, UPPER(title), UPPER(description)
    FROM film;
```

- Obter uma parte de uma string (substring):

```
SELECT SUBSTR(description, 1, 10)
    FROM film;
```

SELECT - Strings

- Determinar o número de caracteres de uma string:

```
SELECT last_name, first_name,
    CHAR_LENGTH(concat_ws(' ',first_name,last_name))
    FROM actor;
```

SELECT - Caracteres Especiais

- Selecção dos filmes cujos títulos começam por 'Dr':

```
SELECT *
    FROM Film
    WHERE title LIKE 'Dr%';
```

- Relação dos filmes cujo título obedece à string de pesquisa 'Ba%Da%':

```
SELECT *
    FROM film
    WHERE title LIKE 'Ba%Da%';
```

SELECT - Caracteres Especiais

- Apresentar uma relação de todos os filmes cujos títulos não contêm strings com quatro caracteres, começadas por ‘Jo', seguidas por dois quaisquer caracteres, seguidas por um espaço e por fim por um 'M' e outra coisa qualquer:

```
SELECT *
    FROM film
    WHERE title NOT LIKE '%Jo__ M%';
```

SELECT - Expressões Regulares

- Relação de todos os funcionários cujo primeiro nome começa por 'm':

```
SELECT *
    FROM staff
    WHERE first_name REGEXP '^m';
```

- Relação de todos os funcionários cujo primeiro nome termina em 'e' ou 'n':

```
SELECT *
    FROM staff
    WHERE first_name REGEXP '[en]$';
```

- Relação de todos os filmes cujo título apenas contenha letras de 'x' a 'z':

```
SELECT *
    FROM film
    WHERE title REGEXP '[x-z]';
```

SELECT - Expressões Regulares

- Relação de todos os clientes cujo primeiro nome contenha um 'z' ou um '2' em qualquer posição:

```
SELECT *
    FROM customer
    WHERE first_name REGEXP '[z2]';
```

- Relação dos atores cujo ultimo nome contenha apenas 4 caracteres:

```
SELECT *
    FROM actor
    WHERE last_name REGEXP '^.{4}$';
```

Agrupamento de Resultados

- Utilização da clausula GROUP BY, com limitação de valores e ordenação de resultados.

```
SELECT Customer_Id AS NrCliente,
    COUNT(*) AS NrPagamentos,
    SUM(Amount) AS TotalPagamentos,
    MAX(Amount) AS MaiorPagamento,
    MIN(Amount) AS MenorPagamento,
    AVG(Amount) AS MédiaPagaments
FROM Payment
GROUP BY Customer_Id
ORDER BY SUM(Amount) DESC
LIMIT 10;
```

Agrupamento de valores

Ordenação de valores

Limitação do número de registos no conjunto de resultados

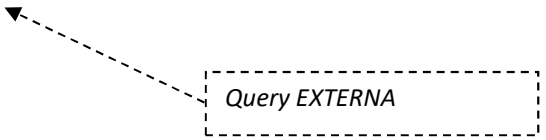
Utilização de Sub-queries

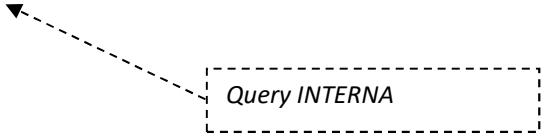
- Por vezes queremos, por exemplo, utilizar os resultados de uma dada query como elementos de trabalho numa outra query. Para isso podemos utilizar sub-queries.
- Quando uma query envolve uma sub-query, designa-se a sub-query por query interna e a query que a acolhe por query externa.

Sub-queries - Exemplos

- Utilização de uma query como sub-query de uma outra, com ordenação de resultados:

```
SELECT Customer_Id AS Identificador,  CONCAT(First_Name,' ',
    Last_Name) AS Nome
FROM Customer
WHERE Customer_ID NOT IN (
    SELECT Customer_ID
        FROM Customer
        WHERE Customer_id IN (1,2,3)
)
ORDER BY Nome ASC;
```





Sub-queries - Exemplos

- Mudança do estado dos clientes que não alugam um filme há mais de 120 dias:

```
UPDATE customer
    SET active = 0
    WHERE customer_id IN (
        SELECT customer_ID
        FROM rental
        WHERE DATEDIFF(curdate(),rental_date) > 120
    );
```

```
-- Desativar modo de segurança
SET SQL_SAFE_UPDATES=0;
-- Ativar modo de segurança
SET SQL_SAFE_UPDATES=1;
```

Combinação de Dados

- A combinação de dados entre uma ou mais tabelas é assegurada pela SQL através de uma instrução de junção de dados (JOIN), que permite, através de uma dado critério de junção fazer a combinação dos dados contidos nessas tabelas.
- Existem vários tipos de junção, com diferentes propósitos e formas de expressão. Todavia, em termos de SQL, podemos organizá-las em dois tipos:
  - Internas – INNER JOIN.
  - Externas – OUTER JOIN (LEFT, RIGHT e FULL).

A Cláusula de Junção - Estrutura

```
table_references:
    escaped_table_reference [, escaped_table_reference] ...
escaped_table_reference: {
    table_reference
  | { OJ table_reference }
}
table_reference: {
    table_factor
  | joined_table
}
table_factor: {
    tbl_name [PARTITION (partition_names)]
        [[AS] alias] [index_hint_list]
(…)
```

Obs.: Apenas está apresentada uma parte da estrutura da clásula

Fonte: MySQL 8.0 Reference Manual, JOIN Clause
https://dev.mysql.com/doc/refman/8.0/en/join.html

Operações de Junção

- As operações de combinação de dados mais frequentes são realizadas com operações de junção interna (INNER JOIN), que envolvem, pelo menos, duas tabelas e um critério de junção.
- As junções internas também são designadas por equijunções, uma vez que o critério de junção se baseia numa igualdade entre atributos de tabelas.
- No caso de uma operação de autojunção apenas se envolve uma única tabela, cujos dados serão combinados com eles mesmos. ☺

Operações de Junção

- As operações de junção (JOIN) são operações bastante úteis no contexto de qualquer sistema de bases de dados, porque:
  - Permitem combinar dados de uma ou mais tabelas dando como resultado um único conjunto de dados (result set);
  - Têm melhor desempenho que as operações envolvendo sub-queries;
  - O resultado de uma operação de junção apenas devolve os registos que satisfaçam uma dada condição (a condição de junção).

Junções Internas - Exemplos

- Lista dos filmes alugados pelos clientes '1' e '5':

```
SELECT RE.Customer_id, IV.film_id
    FROM rental as RE INNER JOIN inventory AS IV
    ON RE.inventory_id=IV.inventory_id
    WHERE RE.customer_id IN (1,2);
```

- Uma forma alternativa para a query anterior:

```
SELECT RE.Customer_id, IV.film_id
    FROM rental as RE, inventory AS IV
    WHERE RE.inventory_id=IV.inventory_id
        AND RE.customer_id IN (1,2);
```

Critério de junção incluído na cláusula WHERE

Junções Internas - Exemplos

- Agora com inclusão do nome do cliente e do título do film:

```
SELECT RE.Customer_id, CL.first_name, CL.last_name,
 IV.film_id, FL.title
 FROM rental as RE
     INNER JOIN inventory AS IV
         ON RE.inventory_id=IV.inventory_id
         INNER JOIN customer AS CL
             ON RE.Customer_id = CL.customer_id
             INNER JOIN film AS FL
                 ON IV.film_id = FL.film_id
 WHERE RE.customer_id IN (1,2);
```

Encadeamento de junções

Junções Internas - Exemplos

- Lista dos pagamentos efetuados pelos pelos clientes:

```
SELECT CONCAT(CL.First_name," ",CL.Last_Name) AS Nome,
     PY.Payment_id AS Pagamento, PY.Amount AS Quantia
FROM Payment AS PY       INNER JOIN Customer AS CL
     ON PY.Customer_Id=Cl.Customer_Id
WHERE CL.Customer_id IN (1,2,3);
```

- Lista dos endereços completos dos clientes:

```
SELECT CL.Customer_id AS Cliente,
     CONCAT(CL.First_name," ",CL.Last_Name) AS Nome,
     AD.address AS Endereço, CI.city AS Cidade,
     CO.country AS País
FROM customer AS CL
     INNER JOIN address AS AD ON CL.address_id=AD.address_id
     INNER JOIN city AS CI ON CI.city_id=AD.city_id
     INNER JOIN country AS CO ON CO.country_id=CI.country_id
 ORDER BY Cliente ASC;
```

Junções Internas - Exemplos

- Quais foram os 10 filmes mais aluguados durante o ano de '2005'?

```
SELECT I.Film_Id, F.Title,
    COUNT(R.Rental_Id) AS NrALugueres
FROM Rental AS R INNER JOIN Inventory AS I
       ON R.Inventory_Id=I.Inventory_Id
    INNER JOIN Film AS F
       ON I.Film_Id=F.Film_id
 WHERE YEAR(R.Rental_Date) = 2005
 GROUP BY I.Film_Id
 ORDER BY NrALugueres DESC
 LIMIT 10;
```

Junções Internas - Exemplos

- Quais foram os 10 filmes (Id e Título) mais alugados? Indicar o correspondente número de alugueres e valor recebido:

```
SELECT   FL.Film_id AS IdFilme,  FL.Title AS Titulo,
 COUNT(*) AS NrALugueres,    SUM(PY.Total) AS ValorTotal
FROM Rental AS RT INNER JOIN Inventory AS IV
     ON RT.Inventory_Id=IV.Inventory_Id
 INNER JOIN Film AS FL
     ON IV.Film_Id = FL.Film_Id
 INNER JOIN (
     SELECT Rental_Id, SUM(Amount) AS Total
         FROM Payment
         WHERE Rental_Id IS NOT NULL
         GROUP BY Rental_Id )    AS PY
     ON PY.Rental_Id=RT.Rental_id
GROUP BY FL.Film_id
ORDER BY NrALugueres DESC
LIMIT 10;
```

Junções Internas - Exemplos

- Quais os nomes dos atores dos filmes que foram alugados por clientes do país 'Peru' durante o mês de 'Maio' de 2006:

```
SELECT DISTINCT CONCAT(AC.First_Name,' ',AC.Last_Name) AS NomeAtor
FROM Country as C
    INNER JOIN City AS I ON C.Country_Id=I.Country_Id
    INNER JOIN Address AS A ON I.City_Id=A.City_Id
    INNER JOIN Customer AS U ON U.Address_Id=A.Address_Id
    INNER JOIN Rental AS R ON U.Customer_Id=R.Customer_Id
    INNER JOIN Inventory AS N ON N.Inventory_Id=R.Inventory_Id
    INNER JOIN Film AS F ON N.Film_Id=F.Film_Id
    INNER JOIN Film_Actor AS FA ON FA.Film_Id=F.Film_Id
    INNER JOIN Actor AS AC ON AC.Actor_Id=FA.Actor_Id
WHERE C.Country = 'Peru'
    AND YEAR(R.Rental_Date) = '2005'
    AND MONTHNAME(R.Rental_Date) = 'May';
```

As Junções Externas

- De forma semelhante às junções internas, as junções externas permitem combinar dados entre duas ou mais tabelas com base num dado critério de junção.
- As junções externas incluem, contudo, nos resultados finais todos os registos da tabela à esquerda ou à direita, conforme o tipo de junção, mesmo quando estes não tenham qualquer possibilidade de combinação com qualquer registo da outra tabela.
- As junções externas podem ser à esquerda (LEFT OUTER JOIN), à direita (RIGHT OUTER JOIN), ou completas (à esquerda e à direita).

Junções Externas - Exemplos

- Exemplo de uma junção externa à esquerda, revelando nos resultados todos os registos de clientes, mesmo quando estes não realizaram qualquer aluguer:

```
SELECT *
    FROM customer AS C LEFT OUTER JOIN rental AS R
        ON C.customer_id=R.customer_id;
```

- Exemplo de uma junção externa à direita, revelando nos resultados todos os registos de filmes, mesmo quando estes não tiveram qualquer aluguer:

```
SELECT *
    FROM rental AS RE RIGHT OUTER JOIN inventory AS IV
        ON RE.inventory_id=IV.inventory_id;
```

Junções Externas - Exemplos

```
SELECT C.Customer_Id, C.First_Name, C.Last_Name, T.Title, T.Description
FROM Customer AS C LEFT OUTER JOIN (
          SELECT R.Customer_Id, F.Title, F.Description, R.Rental_Id
     FROM Rental AS R INNER JOIN Inventory AS I
                    ON R.Inventory_Id=I.Inventory_Id
               INNER JOIN Film AS F
                    ON I.Film_Id=F.Film_Id
          WHERE YEAR(R.Rental_Date) = '2005'
               AND WEEK(R.Rental_Date) IN ('20','21')) AS T
               ON C.Customer_Id=T.Customer_Id
UNION
SELECT C.Customer_Id, C.First_Name, C.Last_Name, T.Title, T.Description
FROM Customer AS C RIGHT OUTER JOIN (
          SELECT R.Customer_Id, F.Title, F.Description, R.Rental_Id
          FROM Rental AS R INNER JOIN Inventory AS I
                    ON R.Inventory_Id=I.Inventory_Id
               INNER JOIN Film AS F
                    ON I.Film_Id=F.Film_Id
          WHERE YEAR(R.Rental_Date) = '2005'
               AND WEEK(R.Rental_Date) IN ('20','21')) AS T
     ON C.Customer_Id=T.Customer_Id;
```

Obs.: Uma junção externa completa, utilizando a união (UNION) de uma junção externa à esquerda (LEFT OUTER JOIN) e uma junção externa à direita (RIGHT OUTER JOIN).

Auto-junções

- Uma auto-junção (*self join*) é uma operação de junção que utiliza apenas uma tabela, isto é, combina os dados de uma tabela com os dados da mesma tabela.
- Usualmente, as auto-junções são utilizadas para consultar informação hierárquica ou para comparar um registo com outros registos da mesma tabela.

Auto-junções - Exemplos

- Consulta dos funcionários da loja e do seu respetivo chefe:

```
SELECT S1.Store_Id AS Store,
    CONCAT(S2.First_Name,' ',S2.Last_Name) AS Employee,
    CONCAT(S1.First_Name,' ',S1.Last_Name) AS Manager
    FROM Staff S1 INNER JOIN Staff S2
        ON S1.Staff_Id = S2.Manager_Id
    ORDER BY Store ASC, Employee ASC;
```

Obs.: Para desenvolvermos este exemplo tivemos que acrescentar à tabela "staff" um novo atributo ("manager_id") e de seguida inserir um novo elemento na tabela (um gestor) e atualizar os registos que nela já estava contidos relativos aos outros funcionários de forma a que estes o tivessem como gestor.

Semi-junção

- A operação de semi-junção (*semi join*) permite reduzir o número de registos envolvidos numa operação de junção entre duas tabelas, aplicando sobre o resultado da junção uma projeção envolvendo todos os atributos da primeira tabela (semi-junção à esquerda) ou todos os atributos da segunda tabela (semi-junção à direita).
- Na prática é uma operação bastante frequente.

Semi-junção - Exemplos

- Semi-junção à esquerda – Lista dos clientes com o código de endreço '5' que fizeram alugueres de filmes:

```
SELECT C.*
    FROM Customer AS C INNER JOIN Rental AS R
        ON C.Customer_Id=R.Customer_id
    WHERE C.Address_Id = '5';
```

- Semi-junção à direita – Lista dos clientes com o código de endreço '5' que fizeram pagamentos de alugueres~:

```
SELECT C.*
    FROM Payment AS P INNER JOIN Customer AS C
        ON P.Customer_Id=C.Customer_Id
    WHERE C.Address_Id = '5';
```

Anti-junção

- A operação de anti-junção dá-nos um resultado que contém todos os registos que não satisfaçam a condição de junção de uma operação de junção entre duas tabelas.

Anti-junção - Exemplos

- Listar os clientes que não realizaram qualquer aluguer durante o mês de ‘Janeiro’, 'Fevereiro' e 'Março' de ‘2005’:

```
SELECT *
    FROM Customer
    WHERE NOT EXISTS (
        SELECT *
            FROM Rental
            WHERE YEAR(Rental_Date) = '2005'
                AND MONTHNAME(Rental_Date)
                        IN ('January','February','March'));
```

Operações de Conjuntos

- Em processos de combinação de dados também podemos utilizar as operações tradicionais de conjuntos.
- Em MySQL apenas estão implementadas o produto cartesiano e a união (UNION). As restantes operações sobre conjuntos, diferença e quociente, têm que ser desenvolvidas com uma implementação específica utilizando a instrução SELECT.

O Produto Cartesiano

- O produto cartesiano relaciona os dados de duas tabelas, combinando todos os registos da primeira tabela (n) com todos os registos da segunda tabela (m), dando como resultado uma relação com todos (união) os atributos das duas tabelas e todos os seus registos (n*m).

Produto Cartesiano - Exemplos

- Combinando todos os registos da tabela actor com todos os registos da tabela film:

```
SELECT CONCAT(AC.first_name,' ',AC.last_name) AS Ator,
    FI.title AS Filme
    FROM actor AS AC, film AS FI;
```

- Um produto cartesiano menos exigente:

```
SELECT CONCAT(AC.first_name,' ',AC.last_name) AS Ator,
    FI.title AS Filme
    FROM actor AS AC, film AS FI
    WHERE AC.actor_id IN (198,199,200)
        AND FI.film_id IN (1,2,3);
```

Produto cartesiano entre as tabelas actor e film.

A Operação de União

- A operação de união é realizada entre duas tabelas e permite-nos reunir no conjunto de resultados finais os registos das duas tabelas sem a ocorrência de registos duplicados.
- Para realizar uma operação de união o MySQL disponibiliza-nos a instrução UNION, que tem uma variante UNION ALL para incluir, se desejarmos, os registos duplicados.

União - Exemplos

- Consulta de uma lista de eMails de clientes - "customer" - e de funcionários - "staff" - simultneamente.

```
SELECT customer_id, first_name, last_name, email
    FROM Customer
UNION
SELECT staff_id, first_name, last_name, email
    FROM Staff;
```

- Complemento da query anterior com alguma "cosmética" para realce dos resultados.

```
SELECT 'C' AS Tipo, customer_id AS "Nr Cliente",
    first_name AS Nome, last_name AS Apelido,
    email, '______' AS Observacoes
    FROM Customer
UNION
SELECT 'F', staff_id, first_name, last_name,
    email, '______'
    FROM Staff;
```

União - Exemplos

- Uma operação de união de resultados com utilização de sub-queries:

```
SELECT *
    FROM (
        SELECT 'C' AS Tipo, customer_id AS Nr,
            first_name AS Nome, last_name AS Apelido,
            email, '______' AS Observacoes
            FROM Customer AS C
        UNION
        SELECT 'F', staff_id, first_name, last_name,
            email, '______'
            FROM Staff AS S) AS T
    ORDER BY Nome, Apelido ASC;
```

A Operação de Interseção

- A operação de interseção permite-nos identificar os registos que são iguais em duas tabelas distintas.
- O MySQL não suporta a instrução INTERSECT, que outros sistemas integram nos seus dialetos SQL. Como tal, esta operação apenas pode ser realizada através de uma implementação específica da instrução SELECT. O mesmo acontece para as operações de diferença e quociente.

Interseção - Exemplos

- Identificação dos clientes da cidade de '1' que fizeram alugueres durante o ano de '2005‘ e ‘2006’:

```
SELECT CL.customer_id, CL.first_name, CL.last_name, AD.city_id
     FROM customer AS CL
          INNER JOIN address AS AD ON CL.address_id=AD.address_id
 WHERE customer_id IN (
     SELECT DISTINCT CL.customer_id
          FROM rental AS RE
               INNER JOIN inventory AS IV ON RE.inventory_id=IV.inventory_id
               INNER JOIN film AS FI ON FI.film_Id=IV.film_id
               INNER JOIN customer AS CL ON CL.customer_id=RE.customer_id
               INNER JOIN address AS AD ON AD.address_id = CL.address_id
          WHERE (YEAR(RE.rental_date) = '2005'
               OR YEAR(RE.rental_date) = '2006')
               AND AD.city_id = '1');
```

Diferença - Exemplos

- Identificação dos filmes que não tiveram qualquer aluguer durante o ano de '2005':

```
SELECT film_id, title
FROM film
WHERE film_id NOT IN (
    SELECT DISTINCT FI.film_id
        FROM rental AS RE
            INNER JOIN inventory AS IV
                ON RE.inventory_id=IV.inventory_id

            INNER JOIN film AS FI
                ON FI.film_Id=IV.film_id
        WHERE YEAR(RE.rental_date) = '2005');
```

Quociente - Exemplos

- Lista dos clientes que já alugaram pelo menos um filme de todas as categorias de filmes existentes na base de dados:

```
SELECT DISTINCT V1.Cliente AS Cliente
    FROM vwClientesCategoriasFilmes AS V1
    WHERE NOT EXISTS (
        SELECT V3.Categoria
            FROM vwCategories AS V3
            WHERE V3.Categoria NOT IN (
                SELECT V2.Categoria
                FROM vwClientesCategoriasFilmes AS V2
                WHERE V2.Cliente = V1.Cliente)
    );
```

Tabela virtual (vista) contendo uma lista com os nomes dos clientes e os nomes das categorias de filmes que eles já alugaram..

Tabela virtual (vista) contendo uma lista com os nomes de todas as categorias de filmes.

Manipulação de Dados (DDL)

- Base de Dados Exemplo
  - **Sakila -** https://dev.mysql.com/doc/sakila/en/sakila-installation.html'
- Ferramentas
  - **MySQL Workbench -** https://dev.mysql.com/downloads/workbench/
  - **MySQL Community Server -** https://dev.mysql.com/downloads/mysql/
  - **MySQL Command-Line Client**

Exercícios

- Gestão de Utilizadores
- Atribuição e Remoção de privilégios
- Grupos de privilégios

6

Controlo de Dados (DCL)

Controlo de Dados

- A vertente de controlo de dados (DCL) da SQL incorpora um conjunto de instruções especificamente orientados para controlar o acesso aos dados (e metadados) que estejam armazenados numa base de dados.
- As instruções de control de dados permitem atribuir (GRANT) ou retirar (REVOKE) aos utilizadores (e grupos) permissões de trabalho sobre os diversos objetos da base de dados.

Gestão de Utilizadores

- A gestão de utilizadores nun sistema de bases de dados é realizada através de instruções de descrição de dados (DDL).
- Para que um utilizador possa aceder a uma base de dados é necessário que tenha um perfil de utilização criado no sistema de acolhimento dessa base, bem como possuir um conjunto de permissões (privilégios) que lhe defina aquilo que pode (ou não) realizar com a base de dados.
- Para fazermos a criação de um utilizador (perfil de utilização) num sistema utilizamos a instrução CREATE USER. Para alterarmos ou removermos a definição de um utilizador do sistema utiliza mos, respetivamente, as instruções ALTER USER e DROP USER.

Criação de Utilizadores

- Criação de um utilizador com acesso a partir de qualquer sistema:

```
CREATE USER 'james'@'localhost'
    IDENTIFIED BY 'passJames';
```

- Criação de um utilizar num sistema remoto com IP ’11-0-1-1’:

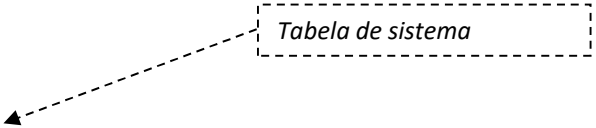
```
CREATE USER ‘gloria'@'11.0.1.1'
    IDENTIFIED BY 'passGloria';
```

- Alteração da password de um utilizador no sistema local (localhost):

```
ALTER USER 'james'@'localhost'
    IDENTIFIED BY 'newJames';
```

Consulta de Utilizadores

- Consulta da informação relativa a todos os utilizadores criados no sistema:



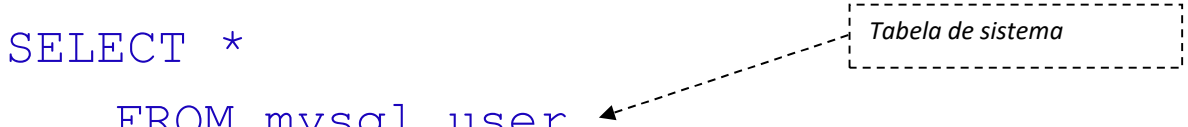

```
SELECT *
    FROM mysql.user;
```

- Consulta dos nomes dos utilizadores, sistemas de acesso e número máximo de conexões, ordenada pela identificação dos utilizadores:

```
SELECT USER, HOST, MAX_CONNECTIONS
    FROM mysql.user
    ORDER BY USER;
```

Consulta de Utilizadores

- Consulta da informação relativa a um conjunto de utilizadores no sistema local:




```
SELECT *
    FROM mysql.user
    WHERE USER IN ('james','ruth','gloria')
       AND HOST = 'localhost';
```

Remoção de Utilizadores

- Remoção de um utilizador no sistema local:

```
DROP USER 'james'@'localhost';
```

Atribuição de Privilégios

- Os privilégios de um utilizador determinam aquilo que ele pode realizar no sistema no qual foi criado.
- Os privilégios mais vulgares são:
  - ALL PRIVILEGES – que concede todos os privilégios a um dado utilizador
  - CREATE – que permite criar base de dados e tabelas.
  - DROP – que permite remover bases de dados e tabelas.
  - DELETE – que permite remover registos de uma tabela.
  - INSERT – que permite inserir registos numa tabela.
  - SELECT – que permite consultar os objetos de uma base de dados.
  - UPDATE – permite modificar registos numa tabela.

Atribuição de Privilégios

- Os privilégios podem ser atribuídos ou retirados. Para atribuir ou retirar privilégios a um utilizador (ou a um grupo) utilizamos, respetivamente, a instrução GRANT ou a instrução REVOKE.

Atribuição de Privilégios - Exemplos

- Atribuição de todos os tipos de privilégios a um utilizador sobre todas as bases de dados e todas as suas tabelas que estejam no sistema local:

```
GRANT ALL PRIVILEGES
    ON *.*
    TO 'gloria'@'localhost';
```

- Carregamento dos privilégios no sistema. Deve ser executado sempre que se modifica os privilégios de um utilizador ou de um grupo:

```
FLUSH PRIVILEGES;
```

- Atribuição de todos os tipos de privilégios a um utilizador sobre uma base de dados e todas as suas tabelas:

```
GRANT ALL PRIVILEGES
    ON sakila.*
    TO 'james'@'localhost';
```

Atribuição de Privilégios - Exemplos

- Atribuição de todos os tipos de privilégios sobre uma tabela de uma base de dados:

```
GRANT ALL PRIVILEGES
    ON sakila.customer
    TO 'james'@'localhost';
```

- Atribuição de privilégios de leitura sobre todas as tabelas de uma base de dados:

```
GRANT SELECT
    ON sakila.rental
    TO 'james'@'localhost';
```

Atribuição de Privilégios - Exemplos

- Atribuição múltipla de privilégios. Atribuição de privilégios de leitura, inserção e remoção sobre todas as tabelas de uma base de dados:

```
GRANT SELECT, INSERT, DELETE
    ON sakila.*
    TO 'james'@'localhost';
```

- Atribuição de privilégios para executar procedimentos:

```
GRANT EXECUTE
    ON PROCEDURE mydb.myproc
    TO 'user'@'host';
```

Atribuição de Privilégios - Exemplos

- Atribuição de privilégios de leitura dos valores de dois atributos de uma tabela:

```
GRANT SELECT (email, username)
    ON sakila.staff
    TO 'james'@'localhost';
```

- Atribuição de privilégios de inserção de valores para um conjunto de atributos de uma tabela:

```
GRANT INSERT (first_name, last_name, job_title,
address_id, store_id, username)
    ON sakila.staff
    TO 'james'@'localhost';
```

Atribuição de Privilégios - Exemplos

- Atribuição do privilégio GRANT a um utilizador de um sistema:

```
GRANT USAGE
    ON *.*
    TO 'user'@'host'
    WITH GRANT OPTION;
```

Outro tipo de Privilégios

- Também podemos fazer a delimitação do acesso dos utilizadores a recursos do sistema – limitação das atuações das contas do sistema.
- A delimitação de recursos aos utilizadores pode ser feito sobre:
  - o número de queries que um utilizador pode executar por hora;
  - o número de modificações que um utilizador pode executar por hora;
  - o número de ligações a um servidor por hora;
  - o número de ligações simultâneas de um utilizador.

Outro tipo de Privilégios - Exemplos

- Delimitação do acesso de um utilizador a alguns recursos do sistema, aquando da sua criação no sistema:

```
CREATE USER 'anne'@'localhost'
 IDENTIFIED BY 'passAnne'
 WITH
     MAX_QUERIES_PER_HOUR 100
     MAX_UPDATES_PER_HOUR 10
     MAX_CONNECTIONS_PER_HOUR 10
     MAX_USER_CONNECTIONS 1;
```

- Alteração dos limites de atuação de um utilizador:

```
ALTER USER 'anne'@'localhost'
     WITH MAX_QUERIES_PER_HOUR 50;
```

Remoção de Privilégios - Exemplos

- Remoção de todos os privilégios a todos os utilizadores no sistema:

```
REVOKE ALL PRIVILEGES, GRANT OPTION
    FROM '%'@'%';
```

- Remoção de todos os privilégios de dois utilizadores em todos os sistemas MySQL.

```
REVOKE ALL PRIVILEGES, GRANT OPTION
    FROM 'james'@'%', 'glory'@'%';
```

- Remoção de privilégios de consulta e de inserção a um grupo:

```
REVOKE SELECT, INSERT    ON sakila.*
    FROM 'managers';
```

Remoção de Privilégios - Exemplos

- Remoção dos privilégios de dois grupos a dois utilizadores:

```
REVOKE 'managers', 'cachiers'
    FROM 'james'@'localhost', 'glory'@'localhost';
```

Refrescamento de Privilégios

- Para que o sistema assuma a atribuição ou remoção de privilégios de um utilizador o grupo devemos realizar o seu refrescamento.
- O refrescamento de privilégios é realizado com a instrução FLUSH PRIVILEGES.

Grupos

- Num sistema é usual um dado conjunto de utilizadores possuir os mesmos privilégios.
- Para se definir um conjunto de privilégios para um dado grupo de utilizadores podemos criar grupos (roles).
- A definição de grupos simplifica o processo de definição de privilégios, reduzindo o tempo de gestão do administrador.
- Um grupo é um conjunto de privilégios com uma dada designação. A criação de um grupo faz através da instrução CREATE ROLE.

Gestão de Grupos - Exemplos

- Criação de um grupo designado por ‘Managers’:

```
CREATE ROLE Managers;
```

- Definição de privilégios para o grupo ‘Managers’:

```
GRANT SELECT, INSERT ON sakila.* TO Managers;
```

- Atribuição dos privilégios do grupo ‘managers’ a um utilizador:

```
GRANT Managers TO james;
```

- Remoção do grupo ‘Managers’ do sistema:

```
DROP ROLE Managers;
```

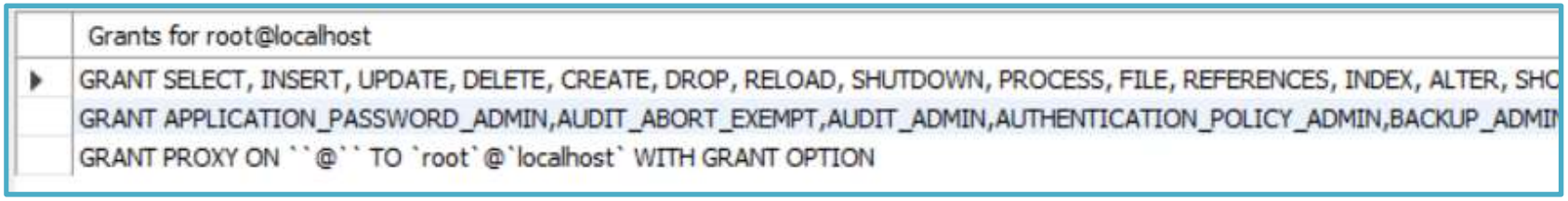
Consulta de Privilégios - Exemplos

- Consulta dos privilégios para o utilizar corrente:

```
SHOW GRANTS;
SHOW GRANTS FOR CURRENT_USER;
SHOW GRANTS FOR CURRENT_USER();
```

- Consulta dos privilégios atribuídos ao utilizador 'root':

```
SHOW GRANTS FOR root@localhost;
```



Grants for root@localhost
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, SHUTDOWN, PROCESS, FILE, REFERENCES, INDEX, ALTER, SHO
GRANT APPLICATION_PASSWORD_ADMIN,AUDIT_ABORT_EXEMPT,AUDIT_ADMIN,AUTHENTICATION_POLICY_ADMIN,BACKUP_ADMI
GRANT PROXY ON ``@`` TO `root`@`localhost` WITH GRANT OPTION

- Consulta dos privilégios atribuídos ao utilizador 'james':

```
SHOW GRANTS FOR james@localhost;
```

Consulta de Privilégios - Exemplos

- Consulta dos privilégios de um utilizador específico no sistema local:

```
SELECT *
    FROM mysql.user
    WHERE host = 'localhost'
       AND user = 'james';
```

Host	User	Select_priv	Insert_priv	Update_priv	Delete_priv	Create_priv	Drop_priv	Reload_priv	Shutdown_priv	Process
localhost	james	N	N	N	N	N	N	N	N	N
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Consulta dos privilégios de um grupo:

```
SHOW GRANTS FOR Managers;
```

Controlo de Dados (DDL)

- Base de Dados Exemplo
  - **Sakila -** https://dev.mysql.com/doc/sakila/en/sakila-installation.html'
- Ferramentas
  - **MySQL Workbench -** https://dev.mysql.com/downloads/workbench/
  - **MySQL Community Server -** https://dev.mysql.com/downloads/mysql/
  - **MySQL Command-Line Client**

Exercícios

- Vistas
- Gestão de vistas
- Utilização de vistas

7

Utilização de Vistas (DDL)

Vistas

- Basicamente, uma vista é uma tabela virtual cujo conteúdo é definido por uma dada consulta (query), aplicada sobre uma ou mais tabelas (incluindo vistas) que nos fornece um conjunto de dados específico estruturado de acordo com um conjunte de atributos e um conjunto de registos.
- Tanto os atributos como os registos provêm das tabelas que estão referenciadas na query que sustenta a vista, sendo revelados quando consultamos a vista em causa.

Utilidade das Vistas

- As vistas são úteis para suportarem a combinação de dados de diferentes tabelas, simplificando, por exemplo, a imagem que os utilizadores têm sobre a bases de dados ou personalizando a forma como eles a podem ver.
- As vistas também podem ser utilizadas como meios de segurança, delimitando o acesso dos utilizadores a dados privados, não os deixando aceder diretamente às tabelas base sobre as quais a vista foi definida.
- As vistas são criadas através da instrução CREATE VIEW.

Criação de Vistas - Exemplos

- Criação de uma vista para fornecer os contactos de clientes:

```
CREATE VIEW vwClientesEMail AS
    SELECT customer_id, first_name, last_name, email
        FROM customer;
```

- Criação de uma vista para fornecer dados para a gestão de alugueres:

```
CREATE VIEW vwGestaoAlugueres AS
    SELECT customer_id AS NrCliente,
        COUNT(*) AS NrPagamentos,SUM(amount) AS TotalPagamentos,
        MAX(amount) AS MaiorPagamento,
        MIN(amount) AS MenorPagamento,
        AVG(amount) AS MédiaPagamento
    FROM payment
    GROUP BY customer_id
    ORDER BY SUM(amount) DESC;
```

Criação de Vistas - Exemplos

- Criação de uma vista com os nomes de todas as categorias de filmes existentes na “Sakila”:

```
CREATE VIEW vwCategories AS
    SELECT Name AS "Categoria"    FROM Category;
```

- Criação de uma vista com os nomes dos clientes e as categorias de filmes que eles já viram:

```
CREATE VIEW vwClientesCategoriasFilmes AS
    SELECT DISTINCT CONCAT(C.First_Name,' ',C.Last_Name)
    AS Cliente, A.Name AS Categoria
    FROM Rental AS R INNER JOIN Customer AS C
        ON R.Customer_Id=C.Customer_Id
        INNER JOIN Inventory AS I
            ON R.Inventory_Id=I.Inventory_Id
            INNER JOIN Film AS F
                ON I.Film_Id=F.Film_Id
                INNER JOIN Film_Category AS FC
                    ON F.Film_Id=FC.Film_Id
                    INNER JOIN Category AS A
                        ON FC.Category_Id=A.Category_Id
    ORDER BY Cliente, Categoria ASC;
```

Obs.: Estas duas vistas foram utilizadas anteriormente no exemplo de demonstração da aplicação de uma operação de quociente entre duas tabelas.

Utilização de Vistas - Exemplos

- Podemos utilizar as vistas que criámos, realizando sobre elas processos de consulta convencionais. Vejamos os seguintes exemplos:

```
SELECT * FROM vwClientesEMail;
SELECT * FROM vwClientesEMail
   WHERE first_name LIKE 'M%';
```

Remoção de Vistas - Exemplos

- Remoção de duas vistas:

```
DROP VIEW vwClientesEMail;
DROP VIEW vwGestaoAlugueres
```

Utilização de Vistas

- Base de Dados Exemplo
  - **Sakila -** https://dev.mysql.com/doc/sakila/en/sakila-installation.html'
- Ferramentas
  - **MySQL Workbench -** https://dev.mysql.com/downloads/workbench/
  - **MySQL Community Server -** https://dev.mysql.com/downloads/mysql/
  - **MySQL Command-Line Client**

Exercícios

- Referências Web

8

Lista de Recursos

Lista de Recursos

- https://m.fca.pt/pt/catalogo/informatica/bases-de-dados-sistemas-inteligentes/bases-de-dados-relacionais/
- *https://www.thoughtco.com/sql-fundamentals-1019780*
- *https://www.iso.org/standard/63555.html*
- https://sigmodrecord.org/publications/sigmodRecord/1203/pdfs/10.industry.zemke.pdf
- https://towardsdatascience.com/coding-and-implementing-a-relational-database-using-mysql-d9bc69be90f5
- https://dev.mysql.com/doc/refman/8.0/en/
- https://dev.mysql.com/doc/refman/8.0/en/tutorial.html

Lista de Recursos

- https://www.mysqltutorial.org/
- https://www.youtube.com/watch?v=7S_tz1z_5bA
- https://www.tutorialspoint.com/mysql/index.htm
- (...)

Sistemas de Bases de Dados
Notas de Leitura

05 >> A Linguagem SQL

Orlando Belo
Departamento de Informática, Escola de Engenharia, Universidade do Minho
PORTUGAL

> www.di.uminho.pt/~omb
> algoritmi.uminho.pt/orlandobelo
> https://orcid.org/0000-0003-2157-8891
> www.researchgate.net/profile/Orlando_Belo
> https://www.linkedin.com/in/orlando-belo-9431942a/?originalSubdomain=pt

Rev. 2022

Fim

05