南京大学
NANJING UNIVERSITY

MYSQL数据库

吴震

南京大学人工智能学院

2024年7月

目录

南京大学
NANJING UNIVERSITY

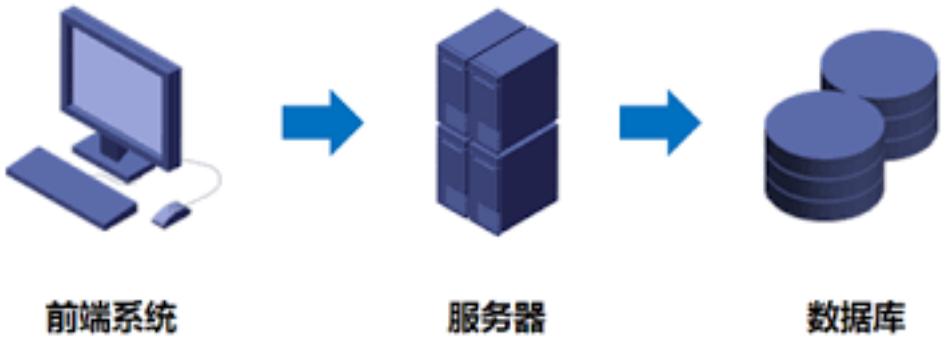
01

数据库基础

DATABASE BASICS

数据库

- 按照数据结构来组织、存储和管理数据的建立在计算机存储设备上的仓库
- 对数据进行新增、截取、更新、删除等操作
- 电子化的文件柜——存储电子文件的处所



为什么需要数据库？

- 数据量越来越大，读写文件速度慢
- 读写文件并解析出数据需要大量重复代码
- 从成千上万的数据中快速查询出指定数据需要复杂的逻辑

数据库分类

南京大学 NANJING UNIVERSITY

关系数据库

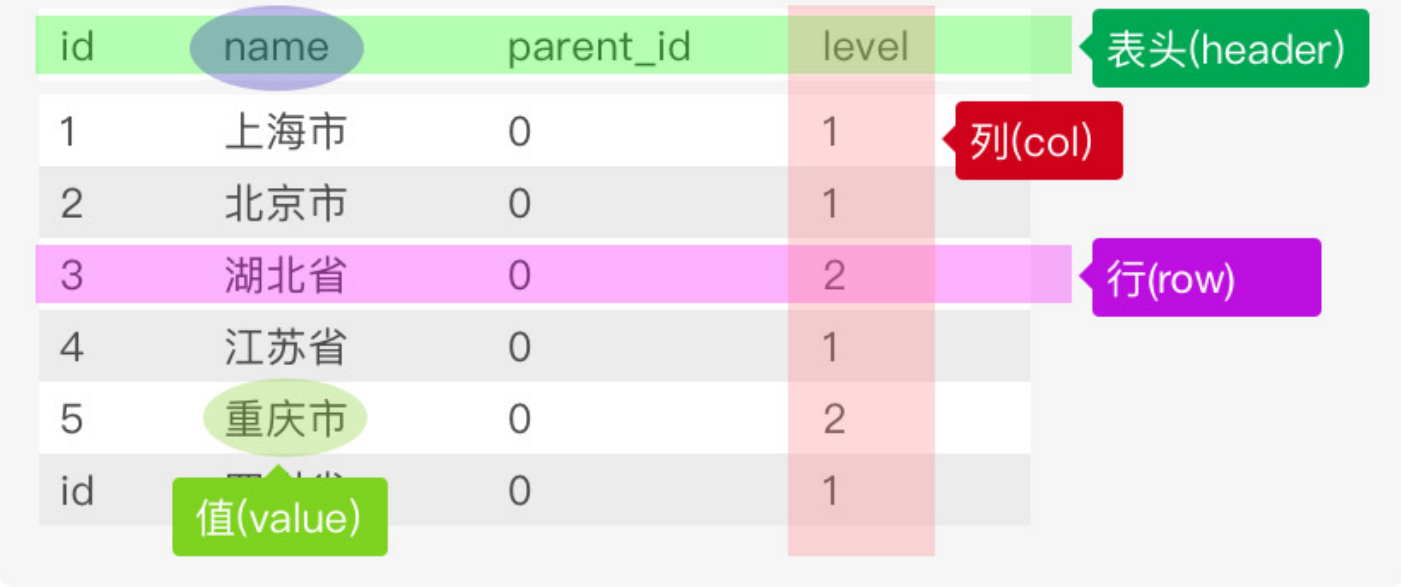
- 创建在关系模型基础上的数据库
- 借助于集合代数等数学概念和方法来处理数据库中的数据
- 主要被用于执行规模小而读写频繁，或者大批量极少写访问的事务

非关系型数据库（NoSQL）

- 轻量、开源、不提供SQL功能的数据库

数据表

- 关系数据库中一个非常重要的对象，是其它对象的基础
- 一系列二维数组的集合，用来存储、操作数据的逻辑结构
- 设有关键字、主键、索引等
- 表中的每一行数据被称为一条记录



数据类型

- 代表不同的信息类型
- 数据类型决定了数据在计算机中的存储格式

名称	类型	说明
INT	整型	4字节整数类型，范围约+/-21亿
BIGINT	长整型	8字节整数类型，范围约+/-922亿亿
REAL	浮点型	4字节浮点数，范围约+/-10^{38}
DOUBLE	浮点型	8字节浮点数，范围约+/-10^{308}
DECIMAL(M,N)	高精度小数	由用户指定精度的小数，例如，DECIMAL(20,10)表示一共20位，其中小数10位，通常用于财务计算
CHAR(N)	定长字符串	存储指定长度的字符串，例如，CHAR(100)总是存储100个字符的字符串
VARCHAR(N)	变长字符串	存储可变长度的字符串，例如，VARCHAR(100)可以存储0~100个字符的字符串
BOOLEAN	布尔类型	存储True或者False
DATE	日期类型	存储日期，例如，2018-06-22
TIME	时间类型	存储时间，例如，12:20:59
DATETIME	日期和时间类型	存储日期+时间，例如，2018-06-22 12:20:59

主键

- primary key，又称主码，用于唯一标识表中每条记录的一个或多个字段的组合
- 主键的取值不能缺失，即不能为空值（Null）
- 主键下的每个值都唯一，任意两行都不具有相同的主键值

id	name	gender	score
1	小明	M	90
2	小红	F	95

字段

主键

- 记录一旦插入到表中，主键最好不要再修改
- 不使用任何业务相关的字段作为主键
- 身份证号、手机号、邮箱地址这些看上去可以唯一的字段，均不可用作主键
- 主键最好是完全业务无关的字段
- 主键常见的数据类型有：
  - 自增整数类型：数据库会在插入数据时自动为每一条记录分配一个自增整数。INT自增类型，表的记录数最多2147483647（约21亿），BIGINT自增类型则可以最多约922亿亿条记录
  - 全局唯一GUID类型：使用一全局唯一的字符串作为主键，类似8f55d96b-8acc-4636-8cb8-76bf8abc2f5

主键

id	name	gender	score
1	小明	M	90
2	小红	F	95

外键

一对多关系

- 一个班级可以有多个学生

如何确定students表的一条记录，例如，id=1的小明，属于哪个班级？

id	name	other columns...
1	小明	...
2	小红	...

students表

id	name	other columns...
1	一班	...
2	二班	...

classes表

外键

一对多关系

- 在students表中加入一列class_id，让它的值与classes表的某条记录相对应

students表中，通过class_id字段，把students表与另classes表关联起来，这种列称为**外键**

id	class_id	name	other columns...
1	1	小明	...
2	1	小红	...
5	2	小白	...

students表

id	name	other columns...
1	一班	...
2	二班	...

classes表

外键

- foreign key，在关系型数据库中，会有一些属性来连系多个数据表之间的关系，这些属性即可作为外键
- 父数据表（Parent Entity）的主键（primary key）会放在另一个数据表，当做属性以创建彼此的关系，这个属性就是外键

外键

id	class_id	name	other columns...
1	1	小明	...
2	1	小红	...
5	2	小白	...

students表

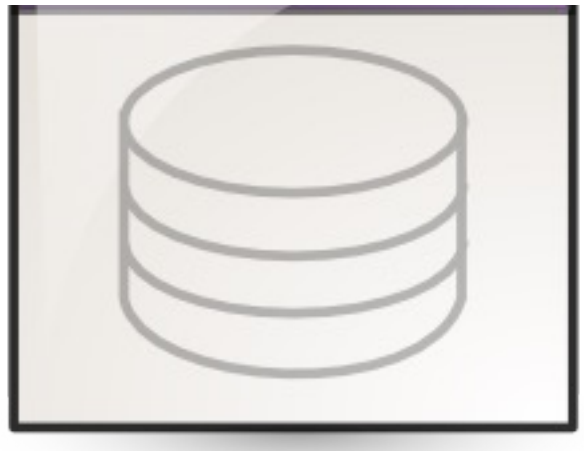
id	name	other columns...
1	一班	...
2	二班	...

classes表

什么是SQL？

- Structured Query Language，结构化查询语言
- 访问和处理关系型数据库的计算机标准语言
- 用于管理关系数据库管理系统
- 可以访问和处理数据库，包括数据表创建、数据插入、查询、更新和删除。

SQL



SQL语言的功能

- 数据定义语言（Data Definition Language，DDL）：DROP、CREATE、ALTER等语句；数据库定义语言。主要用于定义数据库、表、视图、索引和触发器等。CREATE语句主要用于创建数据库、创建表、创建视图。ALTER语句主要用于修改表的定义，修改视图的定义。DROP语句主要用于删除数据库、删除表和删除视图等。

- 数据操作语言（Data Manipulation Language，DML）：INSERT、UPDATE、DELETE语句；数据库操作语言。主要用于插入数据，更新数据，删除数据。INSERT语句用于插入数据，UPDATE语句用于更新数据，DELETE语句用于删除数据。

- 数据查询语言（Data Query Language，DQL）：SELECT语句。主要用于查询数据。

- 数据控制语言（Data Control Language，DCL）：数据库控制语言。主要用于控制用户的访问权限。其中GRANT语句用于给用户增加权限，REVOKE语句用于收回用户的权限。

02

MySQL

MYSQL INTRODUCTION

什么是MYSQL？

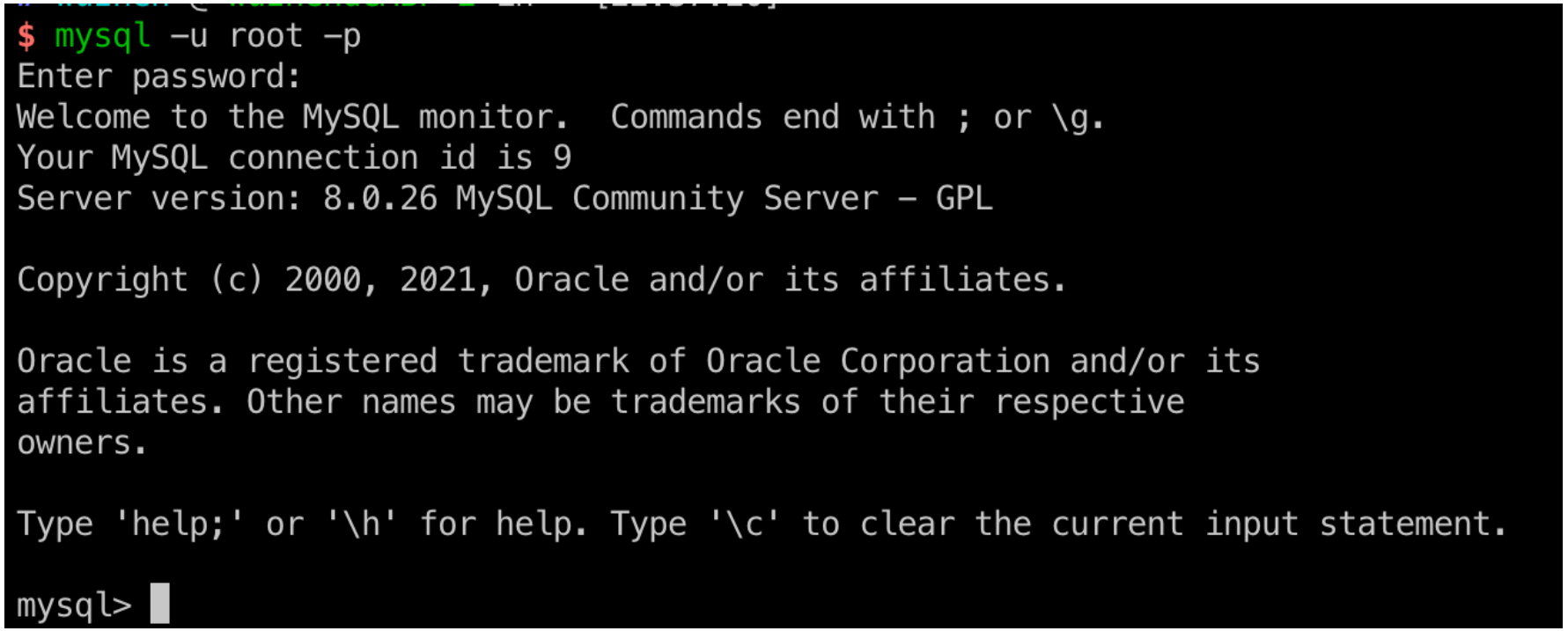
南京大学 NANJING UNIVERSITY

- 一个开放源代码的关系型数据库管理系统（DBMS），支持SQL语言
- 目前应用最广泛的开源关系数据库，性能高、成本低、可靠性好
- 由瑞典的MySQL AB公司开发，2008年被SUN收购，2009年被Oracle收购
- 使用C和C++编写，并使用了多种编译器进行测试，保证源代码的可移植性
- 为多种编程语言提供了API，如C、C++、Java、Python等

MYSQL安装

- Windows系统MySQL安装教程
- Ubuntu系统MySQL安装教程
- Mac系统MySQL安装教程

登录MYSQL


$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.26 MySQL Community Server - GPL
Copyright (c) 2000, 2021, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>

查看数据库

使用SQL语句来查看当前所有存在的数据库

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)
```

创建、删除数据库

```
mysql> CREATE DATABASE test;
Query OK, 1 row affected (0.01 sec)
```

创建数据库test

```
mysql> DROP DATABASE test;
Query OK, 0 rows affected (0.01 sec)
```

删除数据库test

```
mysql> USE test;
 Database changed
```

使用数据库test（需要创建好）

创建、删除数据表

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL,
    birthdate DATE,
    is_active BOOLEAN DEFAULT TRUE
);
```

在数据库test中创建一张数据表users

- id: 用户 id，整数类型，自增长，作为主键。
- username: 用户名，变长字符串，不允许为空。
- email: 用户邮箱，变长字符串，不允许为空。
- birthdate: 用户的生日，日期类型。
- is_active: 用户是否已经激活，布尔类型，默认值为 true。

```
DROP TABLE users;
```

删除数据表users

设置外键

```
ALTER TABLE students
ADD CONSTRAINT fk_class_id
FOREIGN KEY (class_id)
REFERENCES classes (id);
```

- 外键约束的名称fk_class_id可以任意
- FOREIGN KEY (class_id)指定了class_id作为外键
- REFERENCES classes (id)指定了这个外键将关联到classes表的id列（即classes表的主键）

向数据表添加数据

```
INSERT INTO users (username, email, birthdate, is_active)
VALUES ('test', 'test@runoob.com', '1990-01-01', true);
```

添加单条记录

```
INSERT INTO users (username, email, birthdate, is_active)
VALUES
    ('test1', 'test1@runoob.com', '1985-07-10', true),
    ('test2', 'test2@runoob.com', '1988-11-25', false),
    ('test3', 'test3@runoob.com', '1993-05-03', true);
```

添加多条记录

查询数据


```
-- 选择所有列的所有行
SELECT * FROM users;

-- 选择特定列的所有行
SELECT username, email FROM users;

-- 添加 WHERE 子句，选择满足条件的行
SELECT * FROM users WHERE is_active = TRUE;

-- 添加 ORDER BY 子句，按照某列的升序排序
SELECT * FROM users ORDER BY birthdate;

-- 添加 ORDER BY 子句，按照某列的降序排序
SELECT * FROM users ORDER BY birthdate DESC;

-- 添加 LIMIT 子句，限制返回的行数
SELECT * FROM users LIMIT 10;
```

WHERE子句

南京大学 NANJING UNIVERSITY

WHERE 子句可以有条件地从表中选取数据

```
-- 等于条件
SELECT * FROM users WHERE username = 'test';

-- 组合条件（AND、OR）
SELECT * FROM products WHERE category = 'Electronics' AND price > 100.00;

-- 模糊匹配条件（LIKE）
SELECT * FROM customers WHERE first_name LIKE 'J%';

-- IN 条件
SELECT * FROM countries WHERE country_code IN ('US', 'CA', 'MX');

-- BETWEEN 条件
SELECT * FROM orders WHERE order_date BETWEEN '2023-01-01' AND '2023-12-31';

-- IS NOT NULL 条件
SELECT * FROM customers WHERE email IS NOT NULL;
```

更新数据表记录


```
-- 更新单个列的值
UPDATE employees
SET salary = 60000
WHERE employee_id = 101;

-- 更新多个列的值
UPDATE orders
SET status = 'Shipped', ship_date = '2023-03-01'
WHERE order_id = 1001;

--更新使用子查询的值
UPDATE customers
SET total_purchases = (
    SELECT SUM(amount)
    FROM orders
    WHERE orders.customer_id = customers.customer_id
)
WHERE customer_type = 'Premium';
```

删除数据表记录


```
--删除符合条件的行
DELETE FROM students
WHERE graduation_year = 2021;

--删除所有行
DELETE FROM orders;

--使用子查询删除符合条件的行
DELETE FROM customers
WHERE customer_id IN (
    SELECT customer_id
    FROM orders
    WHERE order_date < '2023-01-01'
);
```

南京大学
NANJING UNIVERSITY

03

Python连接MySQL

USING MYSQL IN PYTHON

PYMYSQL

南京大学 NANJING UNIVERSITY

- PyMySQL 是在 Python3.x 版本中用于连接 MySQL 服务器的一个库

```
pip3 install PyMySQL
```

PyMySQL安装

数据库连接

```
#!/usr/bin/python3

import pymysql

# 打开数据库连接
db = pymysql.connect(host='localhost', user='root', password='****', database='test')

# 使用 cursor() 方法创建一个游标对象 cursor
cursor = db.cursor()

# 使用 execute()  方法执行 SQL 查询
cursor.execute("SELECT VERSION()")

# 使用 fetchone() 方法获取单条数据.
data = cursor.fetchone()

print ("Database version : %s " % data)

# 关闭数据库连接
db.close()
```

创建数据表

```
#!/usr/bin/python3

import pymysql

# 打开数据库连接
db = pymysql.connect(host='localhost', user='root', password='****', database='test')

# 使用 cursor() 方法创建一个游标对象 cursor
cursor = db.cursor()

# 使用 execute() 方法执行 SQL，如果表存在则删除
cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")

# 使用预处理语句创建表
sql = """CREATE TABLE EMPLOYEE (
         FIRST_NAME  CHAR(20) NOT NULL,
         LAST_NAME  CHAR(20),
         AGE INT,
         SEX CHAR(1),
         INCOME FLOAT )"""

cursor.execute(sql)

# 关闭数据库连接
db.close()
```

创建数据表

```
#!/usr/bin/python3

import pymysql

# 打开数据库连接
db = pymysql.connect(host='localhost', user='root', password='****', database='test')

# 使用 cursor() 方法创建一个游标对象 cursor
cursor = db.cursor()

# 使用 execute() 方法执行 SQL，如果表存在则删除
cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")

# 使用预处理语句创建表
sql = """CREATE TABLE EMPLOYEE (
         FIRST_NAME  CHAR(20) NOT NULL,
         LAST_NAME  CHAR(20),
         AGE INT,
         SEX CHAR(1),
         INCOME FLOAT )"""

cursor.execute(sql)

# 关闭数据库连接
db.close()
```

添加数据


```
#!/usr/bin/python3

import pymysql

# 打开数据库连接
db = pymysql.connect(host='localhost', user='root', password='****', database='test')

# 使用 cursor() 方法创建一个游标对象 cursor
cursor = db.cursor()

# SQL 插入语句
sql = """INSERT INTO EMPLOYEE(FIRST_NAME,
         LAST_NAME, AGE, SEX, INCOME)
         VALUES ('Mac', 'Mohan', 20, 'M', 2000)"""
try:
   # 执行sql语句
   cursor.execute(sql)
   # 提交到数据库执行
   db.commit()
except:
   # 如果发生错误则回滚
   db.rollback()

# 关闭数据库连接
db.close()
```

查询数据

```
#!/usr/bin/python3

import pymysql

# 打开数据库连接
db = pymysql.connect(host='localhost', user='root', password='****', database='test')

# 使用 cursor() 方法创建一个游标对象 cursor
cursor = db.cursor()

# SQL 查询语句
sql = "SELECT * FROM EMPLOYEE \
       WHERE INCOME > %s" % (1000)
try:
   # 执行SQL语句
   cursor.execute(sql)
   # 获取所有记录列表
   results = cursor.fetchall()
   for row in results:
      fname = row[0]
      lname = row[1]
      age = row[2]
      sex = row[3]
      income = row[4]
       # 打印结果
      print ("fname=%s,lname=%s,age=%s,sex=%s,income=%s" % \
             (fname, lname, age, sex, income ))
except:
   print ("Error: unable to fetch data")

# 关闭数据库连接
db.close()
```

豆瓣爬虫数据库设计建议

- 电影信息表movies
- 电影评论表comments
- 电影信息表movies的主键作为电影评论表comments的外键

参考资料

- RUNOOB MySQL教程
- 廖雪峰博客 MySQL教程
- RUNOOB Python操作MySQL教程