ODBC——开放数据库连接

- ODBC定义了一套基于SQL的、公共的、与数据库无关的API（应用程序设计接口），使每个应用程序利用相同的源代码就可访问不同的数据库系统，存取多个数据库中的数据，从而使得应用程序与数据库管理系统（DBMS）之间在逻辑上的独立性，使应用程序具有数据库无关性。
- 层次：应用程序→ODBC应用程序接口→ODBC驱动程序管理器→驱动程序→数据源→DBMS→数据库

JDBC——Java数据库连接

- JDBC是一种用于执行SQL语句的Java API，它由一组用Java编程语言编写的类和接口组成，为数据库开发人员提供了一个标准的API，使他们能够用纯Java API 来编写数据库应用程序。
- 层次：应用→JDBC API→JDBC驱动程序→数据库
- 步骤：导入java.sql包→加载注册驱动→创建Connection对象→创建Statement对象→执行SQL语句→使用ResultSet对象返回结果→关闭各个对象

```
import java.sql.*;
public class JDBCDemo{
        public static void main(String[] args) throws Exception{
                //加载驱动
                Class.forName("com.postgresql.Driver");
                //创建连接
                Connection conn =
DriverManager.getConnection("jdbc:postgresql://localhost:5432/testdb","postgres","123456"
);
                //创建Statement对象
                Statement stmt = conn.createStatement();
                //执行SQL语句
                ResultSet rs = stmt.executeQuery("select * from student");
                //处理结果集
                while(rs.next()){
                        System.out.println(rs.getString("sno")+"
"+rs.getString("sname"));
                }
                //关闭各个对象
                rs.close();
                stmt.close();
                conn.close();
        }
}
```

NOSQL概述

关系数据库面临的挑战

- 数据库高并发读写需求
- 海量数据的高效存储和处理
- 数据库高扩展性和高可用性需求
- 大数据处理（海量数据+复杂计算）方面的要求
  - 大数据（海量数据+复杂计算）
    - 5V特征
      - 超量Volume、高速Velocity、异构Variety、真实Veracity、价值Value
    - 挑战
      - 数据的异构性和不完备性
      - 数据处理的时效性
      - 数据的安全与隐私保护
      - 大数据的能耗问题
      - 大数据管理易用性问题

NoSQL理论基础

- CAP理论
  - 在分布式环境下的三个核心需求CAP（一致性Consistency，可用性Availability、分区容忍性Partition Tolerance）
    - 一致性：数据修改后所有结点在同一时间具有相同数据
    - 可用性：每个操作无论失败系统都会在一定时间内返回结果
    - 分区容忍性：系统中信息丢失或失败不会影响系统继续运行，在网络被分割成若干独立区域时仍可工作
  - 不可能同时满足三个，最多满足两个
  - CA——单点集群，可扩展性不大
  - CP——性能不高
  - AP——一致性低
- BASE模型
  - BASE含义
    - Basically Available——基本可用；系统能够基本运行， 一直提供服务。
    - Soft-state——软状态。 "Soft state" 可以理解为"无连接"的, 而 "Hard state" 是"面向连接"的；系统不要求一直保持强一致状态。
    - Eventual Consistency——最终一致性，系统在某个时刻达到最终一致性
  - BASE定义为CAP中AP的衍生，在分布式环境下， BASE是数据的属性，BASE强调基本的可用性，按照功能划分数据库
- 最终一致性理论
  - 一致性的分类
    - 强一致性——要求无论更新操作实在哪一个副本执行，之后所有的读操作都要能获得最新的数据
    - 弱一致性——用户读到某一操作对系统特定数据的更新需要一段时间，称这段时间为“不一致性窗口”

- 最终一致性——弱一致性的一种特例，保证用户最终能够读取到某操作对系统特定数据的更新
    - 因果一致性
    - 读一致性
    - 会话一致性
    - 单调读一致性
    - 单调写一致性

NoSQL基本概念

- NoSQL是Not Only SQL的缩写,意即“不仅仅是SQL”，即对关系型SQL数据库系统的补充。
- NoSQL的共同特征
    - 不用预定义模式：不需要预定义表结构
    - 无共享架构：数据划分存储在各个服务器上
    - 弹性可扩展
    - 分区
    - 异步复制：基于日志的异步复制，不总能保证一致性
    - BASE：保证事务的最终一致性和软事务
- NoSQL普遍采用的技术
    - 简单数据类型——一次获得单个记录的约束，不支持外键和跨记录关联，数据操作可在单机执行
    - 元数据和应用数据分离——元数据用于管理结点和副本
    - 弱一致性——减少同步开销
- NoSQL数据库的存储模型
    - 列存储
    - 键值对
    - 文档
    - 图

分类	举例	典型应用场景	数据模型	优点	缺点
键值对	Redis	内容缓存；消息队列；排行榜/计数器	Key 指向 Value 的键值对，通常用hash table来实现	查找速度快	数据无结构化，通常只被当作字符串或者二进制数据
列存储	HBase	对象存储，如新闻；推荐画像（稀疏矩阵）；时空数据；订单	以列簇式存储，将同一列数据存在一起	查找速度快，可扩展性强，更容易进行分布式扩展	功能相对局限

分类	举例	典型应用场景	数据模型	优点	缺点
文档	MongoDb	对象及JSON数据存储；缓存；大尺寸、低价值数据	Key-Value对应的键值对，Value为结构化数据	数据结构要求不严格，表结构可变，不需要像关系型数据库一样需要预先定义表结构	查询性能不高，而且缺乏统一的查询语法。
图	Neo4J	社交网络，推荐系统等。专注于构建关系图谱	图结构	利用图结构相关算法。比如最短路径寻址，N度关系查找等	很多时候需要对整个图做计算才能得出需要的信息，而且这种结构不太好做分布式的集群方案。

列存储数据库

一、列存储数据库简介

- 列式数据库把一列中的数据值串在一起存储起来，然后再存储下一列的数据，以此类推。
- 存储的效果是字符串：*C001，C002,C003,数据库原理及应用,操作系统基础,面向对象程序设计,学科基础，学科基础，学科基础，64，64，48，4，4，4*
- 查询中的选择规则是通过列来定义的，列式存储数据库是自动索引化的；数据压缩比高，查询速度高