



NEW PLAN

软件高级架构师

· 一 段 新 征 程 ·





01

嵌入式技术应用



- 本章节在历年考试过程中的分值占比大概是**2-4分**，但是在最近的一次考试过程中竟然一分没考，近三次考试分值从**4分到0分**，不知道会不会是一种趋势，还得再观察**1-2期**才能得到答案，所以同学们还是按照正常节奏做复习
- 本章节在新版教材里所对应的是**2.4**这一个小章以及下篇八大架构里面的嵌入式架构，原本在之前的考试过程中，嵌入式技术就是一个容易出偏硬件超纲题的考试内容，在改版之后不仅没有增加硬件这块的内容，反而比之前版本的内容更少了，而是在嵌入式架构里面增加了很多理论概念性的东西，考试中如果仍然碰到不会的超纲题，心态放好即可
- 被考到知识点有：
 - 嵌入式系统组成及特点
 - 嵌入式系统分类
 - 嵌入式软件组成及特点
 - 安全软件设计
- 在改版之后的考试中，分别考察的知识点为：
 - 2023年11月：M2M（智能化机器+硬件+网络+中间件+应用）、D0178、NPU、嵌入式系统设计考虑因素
 - 2024年05月：嵌入式系统屏蔽复杂性、嵌入式常见架构模式
 - 2024年11月：无

嵌入式系统概念

- 嵌入式系统是以**应用为中心、以计算机技术为基础，并将可配置与可裁剪的软、硬件集成于一体的专用计算机系统**，需要满足应用对功能、可靠性、成本、体积和功耗等方面的要求。
- 从计算机角度看，**嵌入式系统是指嵌入各种设备及应用产品内部的计算机系统**。它主要完成信号控制的功能，体积小、结构紧凑，可作为一个部件埋藏于所控制的装置中。
- 一般嵌入式系统由**嵌入式处理器、相关支撑硬件、嵌入式操作系统、支撑软件以及应用软件**组成。

嵌入式技术广泛应用于各个行业，例如：

- 汽车电子：发动机控制、**ABS**防抱死、天窗控制等。
- 家电产品：空调控制、洗衣机控制、冰箱控制等。
- 通信设备：路由器、交换机、手机、**base station**等。
- 安防监控：监控摄像头、生物识别等。
- 工业自动化：**PLC**控制器、机器人控制等。
- 医疗设备：心电监护仪、B超仪器、血糖仪等。



从传统意义上讲，嵌入式系统主要由以下部件组成：

- 嵌入式微处理器(MCU)
- 存储器(RAM/ROM)
- 内(外)总线逻辑
- 定时/计数器
- **看门狗电路：定时器溢出则中断，系统复位处理。**
- I/O接口（串口、网络、USB、JTAG接口-用来进行CPU调试的常用接口）
- 外部设备(UART、LED等)
- 其他部件

嵌入式系统的发展大致经历了五个阶段：

- 第一阶段：单片微型计算机(SCM)阶段，即单片机时代。
- 第二阶段：微控制器(MCU)阶段。
- 第三阶段：片上系统(SoC)。
- 第四阶段：以Internet为基础的嵌入式系统。
- 第五阶段：在智能化、云技术推动下的嵌入式系统。

| 特性 | 冯·诺依曼架构 | 哈佛架构 | 混合架构 |
|---------|----------------------|-------------------|-------------------------|
| 存储器结构 | 程序指令和数据存储在同一存储器中 | 程序指令和数据存储在不同的存储器中 | 部分层级分离（如缓存），部分层级共享（如主存） |
| 总线设计 | 单一总线，用于传输指令和数据 | 独立总线，分别传输指令和数据 | 结合单一总线和独立总线设计 |
| 指令与数据访问 | 顺序访问，存在“冯·诺依曼瓶颈” | 并行访问，指令和数据可同时读取 | 在缓存层级并行，在主存层级顺序 |
| 性能 | 较低，受限于总线带宽和访问冲突 | 较高，指令和数据并行访问提升效率 | 较高，通过缓存优化减少瓶颈 |
| 硬件复杂度 | 简单，成本低 | 复杂，成本较高 | 中等，介于冯·诺依曼和哈佛之间 |
| 灵活性 | 高，易于编程和扩展 | 较低，指令和数据分离增加复杂性 | 较高，兼顾灵活性和性能 |
| 应用场景 | 通用计算机（如PC、服务器） | 嵌入式系统、DSP、实时处理系统 | 现代处理器（如ARM、x86） |
| 典型代表 | Intel x86（早期）、AMD处理器 | 8051单片机、DSP处理器 | ARM Cortex系列、现代x86处理器 |
| 优点 | 设计简单，成本低，灵活性高 | 高性能，适合实时处理 | 兼顾高性能和灵活性 |
| 缺点 | 存在“冯·诺依曼瓶颈”，性能受限 | 硬件复杂，成本高，灵活性较低 | 设计复杂度较高 |

| 分类维度 | 类型 | 描述 | 典型代表 | 拓展说明 |
|--------|--------------|---------------------------------|----------------------------|------------------------------------|
| 按字长宽度 | 4位 | 一次处理4位数据，适合简单控制任务 | 早期计算器、玩具控制器 | 成本低，功耗低，但性能有限，已逐渐被淘汰。 |
| | 8位 | 一次处理8位数据，适合低复杂度控制任务 | 51单片机（如Intel 8051）、AVR、PIC | 广泛应用于家电控制、工业控制等领域，性价比高。 |
| | 16位 | 一次处理16位数据，性能优于8位，适合中等复杂度任务 | TI MSP430、Intel 8096 | 在汽车电子、医疗设备等领域有较多应用。 |
| | 32位 | 一次处理32位数据，性能强大，适合复杂任务 | ARM Cortex-M系列、PowerPC | 主流嵌入式处理器，广泛用于智能手机、物联网设备等。 |
| | 64位 | 一次处理64位数据，性能极高，适合高性能计算任务 | ARM Cortex-A系列、Intel Atom | 主要用于服务器、高性能嵌入式设备和复杂计算任务。 |
| 按系统集成度 | 一般用途型微处理器 | 仅包含CPU，需外接存储器、I/O等部件 | Intel x86、AMD Ryzen | 灵活性高，但设计复杂，适合通用计算任务。 |
| | 单芯片微控制器(MCU) | 将CPU、ROM、RAM、I/O等集成在同一芯片上 | 51单片机、ARM Cortex-M系列 | 集成度高，成本低，适合嵌入式控制任务。 |
| 按用途 | 微控制器(MCU) | 资源有限，适合简单控制与接口应用 | 51单片机、AVR、PIC | 广泛应用于家电、工业控制、汽车电子等领域。 |
| | 微处理器(MPU) | 性能较高，适合复杂控制与高级应用 | ARM Cortex-A系列、PowerPC | 用于智能手机、平板电脑、路由器等高性能设备。 |
| | 数字信号处理器(DSP) | 专为信号处理设计，适合语音/音频/图像处理 | TI TMS320系列、ADI Blackfin | 在通信、音频处理、图像处理等领域有广泛应用。 |
| | 图形处理器(GPU) | 专为图形渲染和并行计算设计 | NVIDIA GeForce、AMD Radeon | 用于游戏、AI计算、科学计算等高性能任务。 |
| | 片上系统(SoC) | 集成MCU/MPU核、内存、外设等，适合便携设备和高集成度应用 | 高通Snapdragon、华为麒麟、三星Exynos | 广泛应用于智能手机、物联网设备、智能硬件等领域，集成度高，性能强大。 |

人工智能(Artificial Intelligence, AI)芯片的定义：从广义上讲只要能够运行人工智能算法的芯片都叫作AI芯片。但是通常意义上的AI芯片指的是针对人工智能算法做了特殊加速设计的芯片，现阶段，这些人工智能算法一般以深度学习算法为主，也可以包括其它机器学习算法。

人工智能芯片四大类 (按技术架构分类)：

- GPU
- FPGA(现场可编程门阵列)
- ASC(专用集成电路)
- 类脑芯片

AI芯片的关键特征：

- 新型的计算范式：AI计算既不脱离传统计算，也具有新的计算特质
- 训练和推断：AI系统通常涉及训练和推断过程
- 大数据处理能力：满足高效能机器学习的数据处理要求
- 数据精度：降低精度的设计
- 可重构的能力：针对特定领域而不针对特定应用的设计，可以通过重新配置，适应新的AI算法、架构和任务
- 开发工具：AI芯片需要软件工具链的支持



真题



嵌入式处理器是嵌入式系统的核心部件，一般可分为嵌入式微处理器(MPU)、微控制器(MCU)、数字信号处理器(DSP)和片上系统(SoC)。以下叙述中，错误的是()。

- A.MPU在安全性和可靠性等方面进行增强，适用于运算量大的智能系统
- B.MCU典型代表是单片机，体积小从而使功耗和成本下降
- C.DSP处理器对系统结构和指令进行了特殊设计，适合数字信号处理
- D.SoC是一个有专用目标的集成电路，其中包括完整系统并有嵌入式软件的全部内容

嵌入式软件是指应用在嵌入式计算机系统当中的各种软件，除了具有通用软件的一般特性，还具有一些与嵌入式系统相关的特点，包括：规模较小、开发难度大、实时性和可靠性要求高、要求固化存储。

- **系统软件（底层驱动和操作系统）**：控制和管理嵌入式系统资源，为嵌入式应用提供支持的各种软件，如设备驱动程序、嵌入式操作系统、嵌入式中间件等。
- **应用软件**：嵌入式系统中的上层软件，定义了嵌入式设备的主要功能和用途，并负责与用户交互，一般面向特定的应用领域，如飞行控制软件、手机软件、地图等。
- **支撑软件**：辅助软件开发的工具软件，如系统分析设计工具、在线仿真工具、交叉编译器等。

板级支持包(BSP)是属于底层驱动和操作系统的一部分，它是针对特定硬件平台的软件包，包含了与硬件相关的驱动程序、引导程序、硬件抽象层等，用于支持特定的硬件平台和操作系统。**BSP**主要负责底层硬件资源的管理和控制，提供给上层软件使用。

主要具有以下两个特点。

- **硬件相关性：**BSP是基于特定的硬件平台(如CPU架构、外设组成等)开发的，它需要对该平台的每一个硬件模块(如CPU、内存、外设)进行详细的配置与编程，以实现对整个系统的初始化与控制。所以，不同的硬件平台需要不同的BSP软件包支持。
- **操作系统相关性：**BSP需要为上层的嵌入式操作系统提供统一的软件接口和硬件运行环境。所以，相同硬件平台的不同操作系统也需要不同的BSP软件包支持。

BSP主要包括两个方面的内容：

- 引导加载程序BootLoader
- 设备驱动程序

Bootloader是嵌入式系统加电后运行的第一段软件代码，是在操作系统内核运行之前运行的一小段程序，通过这段程序，可以初始化硬件设备、建立内存空间的映射图，从而将系统的软硬件环境设置到一个合适的状态，以便为最终调用操作系统内核做好准备。一般包括以下功能：

- ✓ **片级初始化**: 主要完成**微处理器的初始化**，包括设置微处理器的核心寄存器和控制寄存器、微处理器的核心工作模式及其局部总线模式等。片级初始化把微处理器从上电时的默认状态逐步设置成系统所要求的工作状态。**这是一个纯硬件的初始化过程。**
- ✓ **板级初始化**: 通过正确地设置各种寄存器的内容来完成**微处理器以外的其他硬件设备的初始化**。例如，初始化LED显示设备、初始化定时器、设置中断控制寄存器、初始化串口通信、初始化内存控制器、建立内存空间的地址映射等。在此过程中，除了要设置各种硬件寄存器以外，还要设置某些软件的数据结构和参数。因此，**这是一个同时包含有软件和硬件在内的初始化过程。**
- ✓ **加载内核(系统级初始化)**: 将操作系统和应用程序的映像从**Flash存储器**复制到系统的内存当中，然后跳转到系统内核的第一条指令处继续执行。

所以，**BootLoader**为操作系统的正常运行奠定硬件基础，总的来说**BootLoader**的主要任务是初始化硬件设备、加载操作系统内核到内存中，并将控制权转移给内核，从而启动整个系统。

嵌入式操作系统EOS与通用操作系统相比，EOS主要有以下特点：

- **微型化**。EOS的运行平台不是通用计算机，而是嵌入式系统。这类系统一般没有大容量的内存，几乎没有外存，因此，EOS必须做得小巧，以占用尽量少的系统资源。
- **代码质量高**。在大多数嵌入式应用中，存储空间依然是宝贵的资源，这就要求程序代码的质量要高，代码要尽量精简。
- **专业化**。嵌入式系统的硬件平台多种多样，处理器的更新速度快，每种处理器都是针对不同的应用领域而专门设计的。因此，EOS要有很好适应性和移植性，还要支持多种开发平台。
- **实时性强**。嵌入式系统广泛应用于过程控制、数据采集、通信、多媒体信息处理等要求实时响应的场合，因此，实时性成为EOS的又一特点。
- **可裁减和可配置**。应用的多样性要求EOS具有较强的适应能力，能够根据应用的特点和具体要求进行灵活配置和合理裁减，以适应微型化和专业化的要求。

嵌入式实时系统（RTOS）是一种完全嵌入受控器件内部，为特定应用而设计的专用计算机系统。在嵌入式实时系统中，要求系统在投入运行前即具有确定性和可预测性。

- 可预测性：是指系统在运行之前，其功能、响应特性和执行结果是可预测的；
- 确定性是：指系统在给定的初始状态和输入条件下，在确定的时间内给出确定的结果。

实时操作系统的特征：

- ✓ 高精度计时系统

在实时应用系统中，经常需要精确确定实时地操作某个设备或执行某个任务，或精确的计算一个时间函数。这些不仅依赖于一些硬件提供的时钟精度，也依赖于实时操作系统实现的高精度计时功能。

- ✓ 多级中断机制

一个实时应用系统通常需要处理多种外部信息或事件，但处理的紧迫程度有轻重缓急之分。有的必须立即作出反应，有的则可以延后处理。因此，需要建立多级中断嵌套处理机制，以确保对紧迫程度较高的实时事件进行及时响应和处理。

- ✓ 实时调度机制

实时操作系统不仅要及时响应实时事件中断，同时也要及时调度运行实时任务。实时调度机制包括两个方面，一是在调度策略和算法上保证优先调度实时任务；二是建立更多“安全切换”时间点，保证及时调度实时任务。

以下关于RTOS(实时操作系统)的叙述中，不正确的是()。

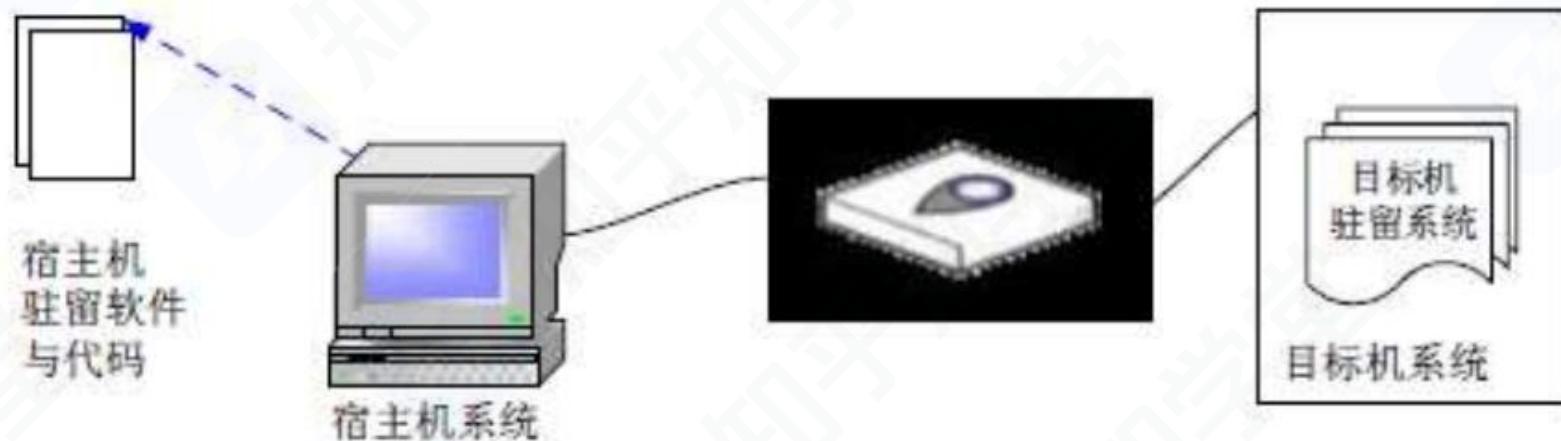
- A.RTOS不能针对硬件变化进行结构与功能上的配置及裁剪
- B.RTOS可以根据应用环境的要求对内核进行裁剪和重配
- C.RTOS的首要任务是调度一切可利用的资源来完成实时控制任务
- D.RTOS实质上就是一个计算机资源管理程序，需要及时响应实时事件和中断

以下描述中，()不是嵌入式操作系统的特点。

- A.面向应用，可以进行裁剪和移植
- B.用于特定领域，不需要支持多任务
- C.可靠性高，无需人工干预独立运行，并处理各类事件和故障
- D.要求编码体积小，能够在嵌入式系统的有效存储空间内运行

一个典型的交叉平台开发环境，包含三个高度集成的部分：

1. 运行在宿主机的强有力的交叉开发工具和实用程序。
2. 运行在目标机上的高性能、可裁剪的实时操作系统。
3. 连接宿主机和目标机的多种通信方式，例如，以太网、USB、串口等。



嵌入式软件的开发可以分为几个阶段：编码、交叉编译、交叉调试。

- **交叉编译 (gcc编译器)**：交叉编译就是在在一个平台上生成可以在另一个平台上执行的代码。嵌入式系统的开发需要借助宿主机(通用计算机)来编译出目标机的可执行代码。
- **交叉调试 (gdb调试器)**：在宿主机和目标机之间进行的交叉调试，调试器仍然运行在宿主机的通用操作系统之上，但被调试的进程却是运行在基于特定硬件平台的嵌入式操作系统中，调试器和被调试进程通过串口或者网络进行通信



真题

以下关于嵌入式系统开发的叙述，正确的是()。

- A.宿主机与目标机之间只需要建立逻辑连接
- B.宿主机与目标机之间只能采用串口通信方式
- C.在宿主机上必须采用交叉编译器来生成目标机的可执行代码
- D.调试器与被调试程序必须安装在同一台机器上

【2018】目前处理器市场中存在CPU和DSP两种类型处理器，分别用于不同场景，这两种处理器具有不同的体系结构，DSP采用（）。

- (A) 冯•诺伊曼结构
- (B) 哈佛结构
- (C) FPGA结构
- (D) 与GPU相同结构

【2018】嵌入式系统设计一般要考虑低功耗，软件设计也要考虑低功耗设计，软件低功耗设计一般采用（）。

- (A) 结构优化、编译优化和代码优化
- (B) 软硬件协同设计、开发过程优化和环境设计优化
- (C) 轻量级操作系统、算法优化和仿真实验
- (D) 编译优化技术、软硬件协同设计和算法优化



真题



【2019】某嵌入式实时操作系统采用了某种调度算法，当某任务执行接近自己的截止期（deadline）时，调度算法将把该任务的优先级调整到系统最高优先级，让该任务获取CPU资源运行。请问此类调度算法是（ ）。

- (A) 优先级调度算法
- (B) 抢占式优先级调度算法
- (C) 最晚截止期调度算法
- (D) 最早截止期调度算法

【2020】实时操作系统主要用于有实时要求的过程控制等领域。因此，在实时操作系统中，对于来自外部的事件必须在（）。

- (A) 一个时间片内进行处理
- (B) 一个周转时间内进行处理
- (C) 一个机器周期内进行处理
- (D) 被控对象允许的时间范围内进行处理



真题



2023年11月第35-36题：M2M全称Machine-to-Machine，是指数据从一台终端传送到另一台终端，也就是机器与机器的对话。

M2M应用系统构成有（），其中（）对获得的数据进行加工分析，提供决策依据。

- A. 智能化机器、M2M硬件、通信网络、中间件、应用
- B. 智能化机器、M2M硬件、工业总线、中间件、工业软件

- C. 传感器、M2M硬件、工业总线、中间件、应用
- D. 传感器、M2M硬件、通信网络、中间件、应用

A. 传感器

B. M2M硬件

C. 工业总线

D. 应用

2023年11月第38题：DO-178B的目的在于为制造机载系统和设备的机载软件提供指导，使其能够提供在满足符合适航要求的安全性水平下完成预期的功能的信心。在DO-178B中，（）是软件适航的基本要求。

- A. 目标、过程、数据
- B. 指南、目标、过程
- C. 目标、过程、结果
- D. 目标、顺序、数据



“民航标准体系”的适航认证，是不可以飞行的。而这个“民航标准体系”中，针对机载软件适航认证的，就是DO-178B标准。经过再次完善和补充，2011年形成了DO-178C标准，它将工具鉴定、基于模型的开发验证技术、面向对象的技术和形式化验证技术纳入适航验证中。

1. DO-178B 的目的和内容

DO-178B的目的是为制造机载系统和设备的机载软件提供指导，使其能够提供在满足符合适航要求的安全性水平下完成预期功能。为了满足该目标，DO-178B给予了以下3方面的指导。

- (1) 软件生命周期过程的目标。
- (2) 为满足上述目标要进行的活动。
- (3) 证明上述目标已经达到的证据，也即软件生命周期数据。

在DO-178B中，目标、过程、数据是软件适航的基本要求。这三方面适航要求是辩证统一的关系，即一旦选择了DO-178B标准作为符合性方法，就必须满足该标准所定义的所有适航目标，而满足这些适航目标的途径则是执行该标准所建议的过程和活动，为证明这些适航目标被满足，应按照该标准所定义的软件生命周期数据来组织相关证据。DO-178B的主要内容就是介绍目标、过程、数据这三个方面的适航要求。目标、过程和数据三个因素是DO-178B的精髓，它贯穿在整个软件生命周期各个过程之中。

(1) 目标。DO-178B标准规定了软件整个生命周期需要达到的66个目标。在DO-178B中，根据软件在系统中的重要程度将软件的安全等级分为A~E五级，不同安全等级的软件，需要达到目标要求不同，其分布详见表2-4。

表2-4软件安全等级与目标关系

| 等级 | 失效状态 | 简要说明 | 目标数量 |
|-----|-------|--------------------------------------|------|
| A 级 | 灾难性的 | 软件异常会导致的后果是：航空器无法安全飞行和着陆 | 6 6 |
| B 级 | 危害性的 | 软件异常会导致的后果是：严重降低了航空器或机组在克服不利运行情况时的能力 | 6 5 |
| C 级 | 严重的 | 软件异常会导致的后果是：显著降低了航空器或机组在克服不利运行情况时的能力 | 5 6 |
| D 级 | 不严重的 | 软件异常会导致的后果是：轻微降低了航空器或机组在克服不利运行情况时的能力 | 2 8 |
| E 级 | 没有影响的 | 软件异常会导致的后果是：不会影响航空器或机组任何能力 | 0 |

- (2) 过程。DO-178B标准把软件生命周期分为“软件计划过程”“软件开发过程”和“软件综合过程”，其中软件开发过程和软件综合过程又分别被细分成4个子过程。
- (3) 数据。DO-178B把软件生命周期中产生的文档、代码、报表、记录等所有产品统称为软件生命周期数据。

DO-178B仅仅定义的是软件生命周期过程，在嵌入式系统中，软件的需求是来自系统分解给它的需求，二者是密不可分的。图2-15展示了软件生存周期与系统生存周期的关系。



真题



2023年11月第46题：嵌入式系统是以应用为中心，以现代计算机技术为基础，能够根据用户需求（）灵活裁剪软硬件模块的专用计算机系统。

- A. 功能、性能、可靠性、成本、体积和功耗
- B. 功能、性能、可靠性、安全、体积和功耗
- C. 功能、性能、可靠性、实时、体积和功耗
- D. 功能、性能、可靠性、稳定、体积和功耗

2023年11月第48题：NPU是国产嵌入式神经网络处理器，是一种专门应用于网络应用数据包的处理器，以下关于NPU的说法中正确的是（）

- A. 采用“数据驱动并行计算”的架构，特别擅长处理视频、图像类的海量多媒体数据。
- B. NPU也是集成电路的一种，是特殊用途集成电路（ASIC）的单一功能
- C. NPU更关注延迟而不是吞吐量
- D. NPU不能并行运行多个线程



真题



2024年05月第69题：以下关于嵌入式系统的说法中，正确的是（）

- A. 嵌入式系统是软硬一体，以软件为主，且通用的
- B. 嵌入式系统开发和编译调试可以在一台设备上完成
- C. 嵌入式系统通常屏蔽操作系统的复杂性，直接运行特定的应用程序
- D. 嵌入式系统分为硬件层、抽象层、操作系统层，应用层等4层

2024年05月第70题：嵌入式系统常见架构模式包括（）

- A. 事件驱动架构模式、数据流架构模式
- B. 层次化架构模式、实时架构模式
- C. 实时架构模式、事件驱动架构模式
- D. 层次化架构模式、递归模式架构



02

系 统 性 能

- 本章节在历年考试中分值为1-2分，但是已经连续3次考试没考到一分了，但是老师仍然保留，因为这里面的知识都是一些常识，学起来并不难，保不准哪次就会在选择题的选项中出现了，所以各位同学正常学习本章内容
- 本章节在新版教材里所对应的是2.9这一小章，内容比较少，也比较简单，在之前考试中有时候会考，有时候不考，如果被考到，一般就是1道题，考察的就是性能指标参数和评价方法，容易拿分
- 考试涉及知识点有：
 - 性能指标介绍
 - 性能计算
 - 性能设计、评估
- 在改版之后的考试中，分别考察的知识点为：
 - 2023年11月：无
 - 2024年05月：无
 - 2024年11月：无

| 分类 | | 指标 |
|----|---------|--|
| 硬件 | 计算机 | 时钟频率(主频)、运算速度与精度、内存的存储容量、存储器的存取周期、数据处理速率PDR、吞吐率、各种响应时间、各种利用率、RASIS特性(即：可靠性Reliability、可用性Availability、可维护性Sericeability、完整性和安全性Integrity and Security)、平均故障响应时间、兼容性、可扩充性、性能价格比 |
| | 路由器 | 设备吞吐量、端口吞吐量、全双工线速转发能力、丢包率、时延、时延抖动、VPN支持能力、端口硬件队列数、基于Web的管理、网管类型等。 |
| | 交换机 | 交换机类型、配置、支持的网络类型、最大ATM端口数、支持协议和标准等 |
| | 网络 | 设备级性能指标、网络级性能指标、应用级性能指标、用户级性能指标、吞吐量。 |
| 软件 | 操作系统 | 系统的可靠性、系统的吞吐率(量)、系统响应时间、系统资源利用率、可移植性。 |
| | 数据库管理系统 | 衡量数据库管理系统的主要性能指标包括数据库本身和管理系统两部分，有：数据库的大小、数据库中表的数量、单个表的大小、表中允许的记录(行)数量、单个记录(行)的大小、表上所允许的索引数量、数据库所允许的索引数量、最大并发事务处理能力、负载均衡能力、最大连接数等。 |
| | WEB服务器 | 最大并发连接数、响应延迟、吞吐量。常见的Web服务器性能评测方法有基准性能测试、压力测试和可靠性测试 |

1、计算机

对计算机评价的主要性能指标有：时钟频率(主频)、运算速度、运算精度、内存的存储容量、存储器的存取周期、数据处理速率PDR(processingdatarate)、吞吐率、响应时间、各种利用率、RASIS特性(即：可靠性Reliability、可用性Availability、可维护性Serviceability、完整性和安全性Integrity and Security)、平均故障响应时间、兼容性、可扩充性、性能价格比。

2、路由器

对路由器评价的主要性能指标有：设备吞吐量、端口吞吐量、全双工线速转发能力、背靠背帧数、路由表能力、背板能力、丢包率、时延、时延抖动、VPN支持能力、内部时钟精度、队列管理机制、端口硬件队列数、分类业务带宽保证、RSVP、IP Diff Serv、CAR支持、冗余、热插拔组件、路由器冗余协议、网管、基于Web的管理、网管类型、带外网管支持、网管粒度、计费能力/协议、分组语音支持方式、协议支持、语音压缩能力、端口密度、信令支持。



3、交换机

对交换机评价的主要性能指标有：交换机类型、配置、支持的网络类型、最大ATM端口数、最大SONET端口数、最大FDDI端口数、背板吞吐量、缓冲区大小、最大MAC地址表大小、最大电源数、支持协议和标准、路由信息协议RIP、RIP2、开放式最短路径优先第2版、边界网关协议BGP、无类域间路由CIDR、互联网成组管理协议IGMP、距离矢量多播路由协议DVMRP、开放式最短路径优先多播路由协议MOSPF、协议无关的多播协议PIM、资源预留协议RSVP、802.1p优先级标记，多队列、路由、支持第3层交换、支持多层(4到7层交换、支持多协议路由、支持路由缓存、可支持最大路由表数、VLAN、最大VLAN数量、网管、支持网管类型、支持端口镜像、QoS、支持基于策略的第2层交换、每端口最大优先级队列数、支持基于策略的第3层交换、支持基于策略的应用级QoS、支持最小/最大带宽分配、冗余、热交换组件(管理卡，交换结构，接口模块，电源，冷却系统、支持端口链路聚集协议、负载均衡。

4、网络

评价网络的性能指标有：设备级性能指标；网络级性能指标；应用级性能指标；用户级性能指标；吞吐量。

5、操作系统

评价操作系统的性能指标有：系统的可靠性、系统的吞吐率(量)、系统响应时间、系统资源利用率、可移植性。



6、数据库管理系统

衡量数据库管理系统的主要性能指标包括数据库本身和管理系统两部分，有：数据库的大小、数据库中表的数量、单个表的大小、表中允许的记录(行)数量、单个记录(行)的大小、表上所允许的索引数量、数据库所允许的索引数量、最大并发事务处理能力、负载均衡能力、最大连接数等等。

7、WEB服务器

评价Web服务器的主要性能指标有：最大并发连接数、响应延迟、吞吐量。

对计算机评价的主要性能指标有时钟频率、()、运算精度和内存容量等。对数据库管理系统评价的主要性能指标有()、数据库所允许的索引数量和最大并发事务处理能力等。

- | | | | |
|--------|-----------|---------|----------|
| A.丢包率 | B.端口吞吐量 | C.可移植性 | D.数据处理速率 |
| A.MIPS | B.支持协议和标准 | C.最大连接数 | D.时延抖动 |

为了优化系统的性能，有时需要对系统进行调整。对于不同的系统，其调整参数也不尽相同。例如，对于数据库系统，主要包括CPU/内存使用状况、()、进程/线程使用状态、日志文件大小等。对于应用系统，主要包括应用系统的可用性、响应时间、()、特定应用资源占用等。

- | | | | |
|---------|-----------|----------|----------|
| A.数据丢包率 | B.端口吞吐量 | C.数据处理速率 | D.查询语句性能 |
| A.并发用户数 | B.支持协议和标准 | C.最大连接数 | D.时延抖动 |

基准程序法(Benchmark): 把应用程序中用得最多、最频繁的那部分核心程序作为评价计算机性能的标准程序，称为基准测试程序(benchmark)。是目前被用户一致承认的测试性能的较好方法，有多种多样的基准程序，包括：

- 整数测试程序。同一厂家的机器，采用相同的体系结构，用相同的基准程序测试，得到的MIPS值越大，一般说明机器速度越快。
- 浮点测试程序。指标MFLOPS(理论峰值浮点速度)。
- SPEC基准程序(SPEC Benchmark)。重点面向处理器性能的基准程序集，将被测计算机的执行时间标准化，即将被测计算机的执行时间除以一个参考处理器的执行时间，
- TPC基准程序。用于评测计算机在事务处理、数据库处理、企业管理与决策支持系统等方面的性能。其中，TPC-C是在线事务处理(On-line Transaction Processing, OLTP)的基准程序，TPC-D是决策支持的基准程序。TPC-E作为大型企业信息服务的基准程序。

大多数情况下，为测试新系统的性能，用户必须依靠评价程序来评价机器的性能。下面列出了4种评价程序，它们评测的准确程度依次递减：真实的程序、核心程序、小型基准程序、合成基准程序

把应用程序中应用最频繁的那部分核心程序作为评价计算机性能的标准程序，称为()程序。()不是对Web服务器进行性能评估的主要指标。

- | | | | |
|--------|-----------|--------|--------|
| A.仿真测试 | B.核心测试 | C.基准测试 | D.标准测试 |
| A.丢包率 | B.最大并发连接数 | C.响应延迟 | D.吞吐量 |

在实际应用中，用户通常依靠评价程序来测试系统的性能。以下评价程序中，()的评测准确程度最低。事务处理性能委员会(Transaction Processing Performance Council, TPC)是制定商务应用基准程序(benchmark)标准规范、性能和价格度量，并管理测试结果发布的非营利组织，其发布的TPC-C是()的基准程序。

- | | | | |
|--------|----------|----------|----------|
| A.核心程序 | B.真实程序 | C.合成基准程序 | D.小型基准程序 |
| A.决策支持 | B.在线事务处理 | C.企业信息服务 | D.联机分析处理 |

阿姆达尔定律描述了并行处理系统中性能改进的程度与系统中可并行化部分所占比例之间的关系。它指出，系统中对某一部件采用更快执行方式所能获得的系统性能改进程度，取决于这种执行方式被使用的频率，或所占总执行时间的比例。

阿姆达尔定律的公式为：

$$S = \frac{1}{(1-P) + \frac{P}{N}}$$

其中：

- S 是系统整体性能的提升倍数。
- P 是可以并行化的部分所占的比例。
- N 是改进后的组件性能提升倍数。



阿姆达尔(Amdahl)定律量化定义了通过改进系统中某个组件的性能，使系统整体性能提高的程度。假设某一功能的处理时间为整个系统运行时间的60%，若使该功能的处理速度提高至原来的5倍，则根据阿姆达尔定律，整个系统的处理速度可提高至原来的 ()倍。

- A.1.333
- B.1.923
- C.1.5
- D.1.829



03

信息系统基础知识



- 本章节在历年考试中分值为0-1分，有一些知识点和下篇八大架构里面的理论知识有一定的重叠，比如EAI，如果把它算成下篇内容（15章面向服务架构）的话，那本章节已经连续3次没考了，但是本章是对整个信息系统的基础知识做了解的章节，里面都是一些信息化建设的常识，还是很建议同学们做个了解的，有助于你理清整个软件系统的脉络，EAI老师保留在对应的脑图里了，但是我们做复习的话，放在下篇的《信息系统架构》部分专门讲。
- 本章节在新版教材里所对应的是第三章，这一章的这个版本跟上个版本有比较大的出入，所以同学们在做改版前的历年真题时，会发现参考答案和书本上有些对不上，我们还是以新版本的说法为主，另外，这一章之前考的多的时候超纲率也比较高，2-3个题目，会有1~2个超纲，比较考验同学们的知识储备以及对名词的理解，因为只有对里面的各种名称理解了，才方便进行使用排除法
- 考试涉及知识点有：
 - 信息系统概述、生命周期、开发方法
 - 业务处理系统TPS、管理信息系统MIS、决策支持系统DSS、专家系统ES、办公自动化系统OAS、企业资源规划ERP
 - 典型信息系统架构模型、电子政务和电子商务
- 在改版之后的考试中，分别考察的知识点为：
 - 2023年11月：无
 - 2024年05月：无（考了EAI（企业应用集成）的概念）
 - 2024年11月：无（也是考了1分的EAI）



诺兰模型：信息系统进化的阶段模型。将计算机信息系统的发展道路划分为**6个阶段**：

- 初始阶段：计算机刚进入企业时只作为办公设备使用，应用非常少。一般仅用于财务部门。
- 传播阶段：企业对计算机有了一定了解，想利用计算机解决工作中的问题，比如进行更多的数据处理，给管理工作和业务带来便利。会大幅度增加软件投入，盲目投入产生问题，效率低。
- 控制阶段：从整体上控制计算机信息系统的发展，在客观上要求组织协调、解决数据共享问题。信息系统呈现单点、分散的特点，系统和资源利用率不高。是计算机管理逐渐变为数据管理的关键。
- 集成阶段：在控制的基础上，企业开始重新进行规划设计，建立基础数据库，并建成统一的信息管理系统。使人、财、物等资源信息能够在企业集成共享，更有效地利用现有的IT系统和资源。
- 数据管理阶段：企业高层意识到信息战略的重要，信息成为企业的重要资源，企业的信息化建设也真正进入到数据处理阶段。使用统一平台，各部门、各系统基本实现资源整合和信息共享。
- 成熟阶段：信息系统已经可以满足企业各个层次的需求，从简单的事务处理到支持高效管理的决策。企业真正把IT与管理过程结合起来，将组织内部、外部的资源充分整合和利用。

业务 (数据) 处理系统(TPS/DPS)：随着企业业务需求的增长和技术条件的发展，人们逐步将计算机应用于企业局部业务(数据)的管理，某个领域具有专家水平的大量知识与经验，能够利用人类专家的知识和解决问题的方法来处理财会管理、销售管理、物资管理和生产管理等，即计算机应用发展到对企业的局部事务的管理。

管理信息系统(MIS)：由人和计算机等组成的，能进行管理信息的收集、传输、存储、加工、维护和使用的系统。形成了对企业全局性的、整体性的计算机应用。能提供企业各级领导从事管理需要的信息，但其收集信息的范围还更多地侧重于企业内部。

决策支持系统(DSS)：是在管理信息系统基础上发展起来的系统，帮助决策者利用数据和模型去解决半结构化决策问题和非结构化决策问题的交互式系统。服务于高层决策的管理信息系统，按功能可分为专用DSS、DSS工具和DSS生成器。

专家系统(ES)：一个智能计算机程序系统，其内部包含某个领域具有专业水平的大量只是和经验。是一种模拟人类专家解决领域问题的计算机程序系统。比如医疗诊断、金融规划等

办公自动化系统(OAS)：人机结合的综合性的办公事务管理系统，或称办公事务处理系统。该系统将当代各种先进技术和设备应用于办公室的办公活动中，使办公活动实现科学化、自动化以达到改善工作环境、最大限度地提高办公事务工作质量和工作效率。



企业资源规划是指建立在信息技术基础上，以系统化的管理思想，为企业提供决策和运营手段的管理平台。ERP系统是将企业所有资源进行集成整合，并进行全面、一体化管理的信息系统。

演变过程：物料需求计划(物料单系统)→制造资源计划(增加库存、分销等)→企业资源计划(打通了供应链，加入财务、人力资源、销售管理等)。

企业有三大资源：物流（物流管理）、资金流（财务管理）、信息流（生产控制管理），现在一般认为人力资源（人力资源管理）是企业第四大资源。

企业的资源计划可从下面三点来理解：

- 管理思想：ERP首先是一种管理思想，将企业资源分类管理，是管理思想的变革。
- 软件产品：其次，ERP是个软件产品，为企业用户提供一体化的解决方案，不是买来直接用的，需要个性化的开发和部署。
- 管理系统：ERP最后是一个管理系统，存在众多的子系统，这些子系统有统一的规划，是互联互通的，便于事前事中监控。



ERP的功能

- 财会管理：会计核算、财务管理。
- 生产控制管理：主生产计划、物料需求计划、能力需求计划、车间控制、制造标准。
- 物流管理：销售管理、库存控制、采购管理。
- 人力资源管理：人力资源规划的辅助决策、招聘管理、工资核算、工时管理、差旅核算。

ERP的五个层次

- 生产计划大纲：是根据经营计划的生产目标制定的，是对企业经营计划的细化，用以描述企业在可用资源的条件下，在一定时期的产量计划。
- 主生产计划：是对企业生产计划大纲的细化，说明在一定时期的如下计划：生产什么，生产多少和什么时候交货。
- 物料需求计划：是对主生产计划的各个项目所需的全部制造件和全部采购件的网络支持计划和时间进度计划。
- 能力需求计划：是对物料需求计划所需能力进行核算的一种计划管理方法。旨在通过分析比较MRP的需求和企业现有生产能力，及早发现能力的瓶颈所在。
- 车间作业计划：是在MRP所产生的加工制造订单（即自制零部件生产计划）的基础上，按照交货期的前后和生产优先级选择原则以及车间的生产资源情况（如设备、人员、物料的可用性、加工能力的大小等），将零部件的生产计划以订单的形式下达给适当的车间。



第一阶段：信息系统的产生阶段，也是信息系统的概念阶段或者是信息系统的需求分析阶段。这一阶段又分为两个过程，一是概念的产生过程，即根据企业经营管理的需要，提出建设信息系统的初步想法；二是需求分析过程，即对企业信息系统的需求进行深入地调研和分析，并形成需求分析报告。

第二阶段：信息系统的开发阶段：最重要、关键的阶段。包括总体规划、系统分析、系统设计、系统实施和系统验收这**5个阶段**。

- 总体规划阶段。信息系统总体规划是系统开发的起始阶段，它的基础是需求分析。作用主要有：指明信息系统在企业经营战略中的作用和地位；指导信息系统的开发；优化配置和利用各种资源，包括内部资源和外部资源。总体规划产出包括信息系统的开发目标、信息系统的总体架构、信息系统的组织结构和管理流程、信息系统的实施计划、信息系统的技术规范等。
- 系统分析阶段。目标是为系统设计阶段提供系统的逻辑模型。以企业的业务流程分析为基础，规划即将建设的信息系统的基本架构，它是企业的管理流程和信息流程的交汇点。内容主要包括组织结构及功能分析、业务流程分析、数据和数据流程分析、系统初步方案等。
- 系统设计阶段。根据系统分析的结果，设计出信息系统的实施方案。主要内容包括系统架构设计、数据库设计、处理流程设计、功能模块设计、安全控制方案设计、系统组织和队伍设计、系统管理流程设计等。
- 系统实施阶段。将设计阶段的结果在计算机和网络上具体实现，也就是将设计文本变成能在计算机上运行的软件系统。由于系统实施阶段是对以前的全部工作的检验，因此，系统实施阶段用户的参与特别重要。系统实施阶段以后，用户逐步变为系统的主导地位。
- 系统验收阶段。信息系统实施阶段结束以后，系统就要进入试运行。通过试运行，系统性能的优劣以及是否做到了用户友好等问题都会暴露在用户面前，这时就进入了系统验收阶段。



第三阶段：信息系统的运行阶段：当信息系统通过验收，正式移交给用户以后，系统就进入了运行阶段。系统维护包括即排错性维护、适应性维护、完善性维护和预防性维护。

第四阶段：信息系统的消亡阶段：在信息系统建设的初期企业就应当注意系统的消亡条件和时机，以及由此而花费的成本。

信息系统建设的原则：高层管理人员介入原则、用户参与开发原则、自顶向下规划原则、工程化原则、其他原则（创新性，整体性，发展性，经济性等）。



结构化方法：结构是指系统内各个组成要素之间的相互联系、相互作用的框架。结构化开发方法就是把软件开发过程划分成若干个阶段和步骤，每个阶段有明确的输入和输出，并采用一定的技术或表示方式来描述各个阶段的工作成果，结构化方法是一种传统的信息系统开发方法，由**结构化分析 (SA)、结构化设计(SD)和结构化程序设计(SP)**三部分有机组合而成，其精髓是自顶向下、逐步求精和模块化设计。

结构化方法的主要阶段：

- 需求分析阶段：与用户沟通，明确软件需求和业务流程，绘制需求模型图。
- 概要设计阶段：根据需求，设计软件的总体结构和模块，描绘系统结构图。
- 详细设计阶段：对各个模块进行详细的接口设计、数据库设计、业务逻辑设计，画出详细设计图。
- 编码实现阶段：根据详细设计文档，选择编程语言编写程序代码。
- 测试阶段：对编写的代码进行测试，确保软件符合需求。
- 部署阶段：将测试通过的代码安装部署到服务器，推向产品环境。

结构化方法的主要特点：

- 开发目标清晰化。结构化方法的系统开发遵循“用户第一”的原则。
- 开发工作阶段化。每个阶段工作完成后，要根据阶段工作目标和要求进行审查，这使各阶段工作有条不紊地进行，便于项目管理与控制。
- 开发文档规范化。结构化方法每个阶段工作完成后，要按照要求完成相应的文档，以保证各个工作阶段的衔接与系统维护工作的遍历。
- 设计方法结构化。在系统分析与设计时，从整体和全局考虑，自顶向下地分解；在系统实现时，根据设计的要求，先编写各个具体的功能模块，然后自底向上逐步实现整个系统。

结构化方法的不足和局限:

- 开发周期长：按顺序经历各个阶段，直到实施阶段结束后，用户才能使用系统。
- 难以适应需求变化：不适用于需求不明确或经常变更的项目。
- 很少考虑数据结构：结构化方法是一种面向数据流的开发方法，很少考虑数据结构。

结构化方法一般利用图形表达用户需求，常用工具有数据流图、数据字典、结构化语言、判定表以及判定树等。



面向对象方法：面向对象(OO)方法认为，客观世界是由各种对象组成的，任何事物都是对象每一个对象都有自己的运动规律和内部状态，都属于某个对象类，是该对象类的一个元素。复杂的对象可由相对简单的各种对象以某种方式而构成，不同对象的组合及相互作用就构成了系统，有以下特点：

- 使用OO方法构造的系统具有更好的复用性，其关键在于建立一个全面、合理、统一的模型。OO方法也划分阶段，但其中的系统分析、系统设计和系统实现三个阶段之间已经没有“缝隙”也就是说，这三个阶段的界限变得不明确，某项工作既可以在前一个阶段完成，也可以在后一个阶段完成；前一个阶段工作做得不够细，在后一个阶段可以补充。
- 面向对象方法可以普遍适用于各类信息系统的开发。
- 面向对象方法的不足之处：必须依靠一定的面向对象技术支持，在大型项目的开发上具有一定的局限性，不能涉足系统分析以前的开发环节。

当前，一些大型信息系统的开发，通常是将结构化方法和OO方法结合起来。首先，使用结构化方法进行自顶向下的整体划分；然后，自底向上地采用OO方法进行开发。因此，结构化方法和OO方法仍是两种在系统开发领域中相互依存的、不可替代的方法



原型化方法：也称为快速原型法，或者简称为原型法。它是一种根据用户初步需求，利用系统开发工具，快速地建立一个系统模型展示给用户，在此基础上与用户交流，最终实现用户需求的信息系统快速开发的方法，有以下分类和特点：

- 按是否实现功能分类：分为水平原型(行为原型，功能的导航)、垂直原型 (结构化原型，实现了部分功能)。
- 按最终结果分类：分为抛弃式原型、演化式原型。
- 原型法可以使系统开发的周期缩短、成本和风险降低、速度加快，获得较高的综合开发效益
- 原型法是以用户为中心来开发系统的，用户参与的程度大大提高，开发的系统符合用户的需求，因而增加了用户的满意度，提高了系统开发的成功率。
- 由于用户参与了系统开发的全过程，对系统的功能和结构容易理解和接受，有利于系统的移交，有利于系统的运行与维护。
- 原型法的不足之处：开发的环境要求高。管理水平要求高。

由以上的分析可以看出，原型法的优点主要在于能更有效地确认用户需求。从直观上来看，原型法适用于那些需求不明确的系统开发。事实上，对于分析层面难度大、技术层面难度不大的系统，适合于原型法开发



面向服务的方法。面向服务(**Service Oriented Architecture**,简称**SOA**)的方法：进一步将接口的定义与实现进行解耦，则催生了服务和面向服务的开发方法

举例：面向服务开发方法，就像我们日常生活中使用的各种服务一样。比如叫外卖，我们可以选择用美团外卖、饿了么等应用，它们都提供了外卖订餐服务。这些外卖应用就是调用了餐馆提供的“外卖服务”。再比如坐车，我们可以选择用滴滴、微信打车等应用叫车，它们调用了出租车公司提供的“叫车服务”。可以看到，餐馆和出租车公司将自己的服务以标准接口的形式对外提供，各种应用可以灵活调用这些服务。如果餐馆或者出租车公司要改进服务流程，对外部调用者没有影响，因为服务接口没有变。这就实现了服务提供者和服务调用者的松耦合。服务调用者只关心服务接口，不关心服务的内部实现。面向服务开发就是这样一种思想，要把系统分割成不同的服务，每个服务完成一项业务功能，服务之间通过开放接口进行交互。这样可以提高服务的重用性和系统的灵活性。比如新增业务时，可以调用已有服务，无需从头开发。

从应用的角度来看，组织内部、组织之间各种应用系统的互相通信和互操作性直接影响着组织对信息的掌握程度和处理速度。如何使信息系统快速响应需求与环境变化，提高系统可复用性、信息资源共享和系统之间的互操作性，成为影响信息化建设效率的关键问题，而**SOA**的思维方式恰好满足了这种需求。



政府到政府 (**Government-to-Government, G2G**) 、政府到企业 (**Government-to-Business, G2B**) 、政府到公众 (**Government-to-Citizen, G2C**) 、企业到政府 (**Business-to-Government, B2G**) 和公众到政府 (**Citizen-to-Government, C2G**) :

G2G (政府到政府) :

- 双边合作协议：两个国家政府之间签署协议，以促进贸易、合作项目、科技研发等领域的合作。例如，两国政府签署的关于环保合作的协议，共同应对气候变化和环境污染。
 - 政府间数据共享：两个政府部门之间共享数据，以加强安全合作、犯罪打击、边境管理等。例如，两国情报部门之间分享情报信息，以便预防跨国恐怖主义活动。

G2B (政府到企业) :

- 政府招标和采购：政府向企业发布招标公告，邀请企业竞标政府项目或服务。例如，政府部门发布建设一所新学校的招标公告，邀请建筑公司投标竞争。
 - 政府发放补贴和奖励：政府向企业提供财政支持，以鼓励特定产业的发展或实现特定目标。例如，政府向可再生能源公司提供津贴，以促进清洁能源的发展。



G2C (政府到公众) :

- 电子政务服务：政府通过互联网平台向公众提供各种在线服务，例如在线申请身份证件、驾驶执照或社会福利等。例如，公民可以通过政府网站在线申请护照。
- 警示和通知：政府向公众发布警示和通知，以提醒市民注意安全或宣传重要信息。例如，政府发布台风预警，提醒市民采取必要的防护措施。

B2G (企业到政府) :

- 税务申报和缴税：企业向政府部门申报税务信息，并缴纳相关税款。例如，企业根据税法规定向税务局报告年度财务状况，并缴纳应纳税款。
- 合规申报和监管报告：企业根据政府法规和监管要求向相关政府部门提交合规申报和监管报告。例如，某制药公司向药品监管机构提交新药研发和临床试验报告。

C2G (公众到政府) :

- 公众意见和建议：公众向政府表达意见、建议或对政策的看法。例如，市民可以通过公开听证会或信件向政府表达对特定法规的意见。
- 投诉和申诉：公众向政府部门提出投诉或申诉，寻求问题解决或寻求公平裁决。例如，消费者向市场监管部门投诉某家企业的不当商业行为。

18.企业数字化转型的五个发展阶段依次是(18)

- A.初始级发展阶段、单元级发展阶段、流程级发展阶段、网络级发展段、生态级发展阶段
- B.初始级发展阶段、单元级发展阶段、系统级发展阶段、网络级发展阶段、生态级发展阶段
- C.初始级发展阶段、单元级发展阶段、流程级发展阶段、网络服发展输段、优化级发展阶段
- D.初始级发展阶段、流程级发展阶段、系统级发展险段、网络级发展阶段、生态级发展阶段

19.从信息化建设的角度出发，以下说法错误的是(19)

- A.有效开发利用信息资源
- B.大力发展信息产业
- C.充分建设信息化政策法规和标准规范
- D.信息化的主体是程序员和项目经理

20.政府、企业等对信息化的需求是能织信息化的原动力，它决定了组织信息化的价值取向和成果效益水平，而需求本身又是极为复杂的，它是一个系统的、多层次的目和体系、组织信息化需求通常包含三个层次，即(20)，三个层次的需求并不是相互孤立的，而是有着内在的联系。

- A.战略需求，运作需求，功能需求
- B.战略需术，运作需求，技术需求
- C.市场需求，技术需求，用户需求
- D.市场需求，技术需求，领域需求

ERP(Enterprise Resource Planning)是建立在信息技术的基础上，利用现代企业的先进管理思想，对企业的物流、资金流和()流进行全面集成管理的管理信息系统，为企业提供决策、计划、控制与经营业绩评估的全方位和系统化的管理平台。在ERP系统中，()管理模块主要是对企业物料的进、出、存进行管理。

- (20)A.产品 B.人力资源 C.信息 D.加工
(21)A.库存 B.物料 C.采购 D.销售

与电子政务相关的行为主体主要有三类，即政府、企(事)业单位及居民。因此，政府的业务活动也主要围绕着这三类行为主体展开。政府与政府、政府与企(事)业单位以及政府与居民之间的互动构成了5种不同的、却又相互关联的领域。其中人口信息采集、处理和利用业务属于(18)领域：营业执照的颁发业务属于(19)领域：户籍管理业务属于(20)领域：参加政府工程投标活动属于(21)领域。

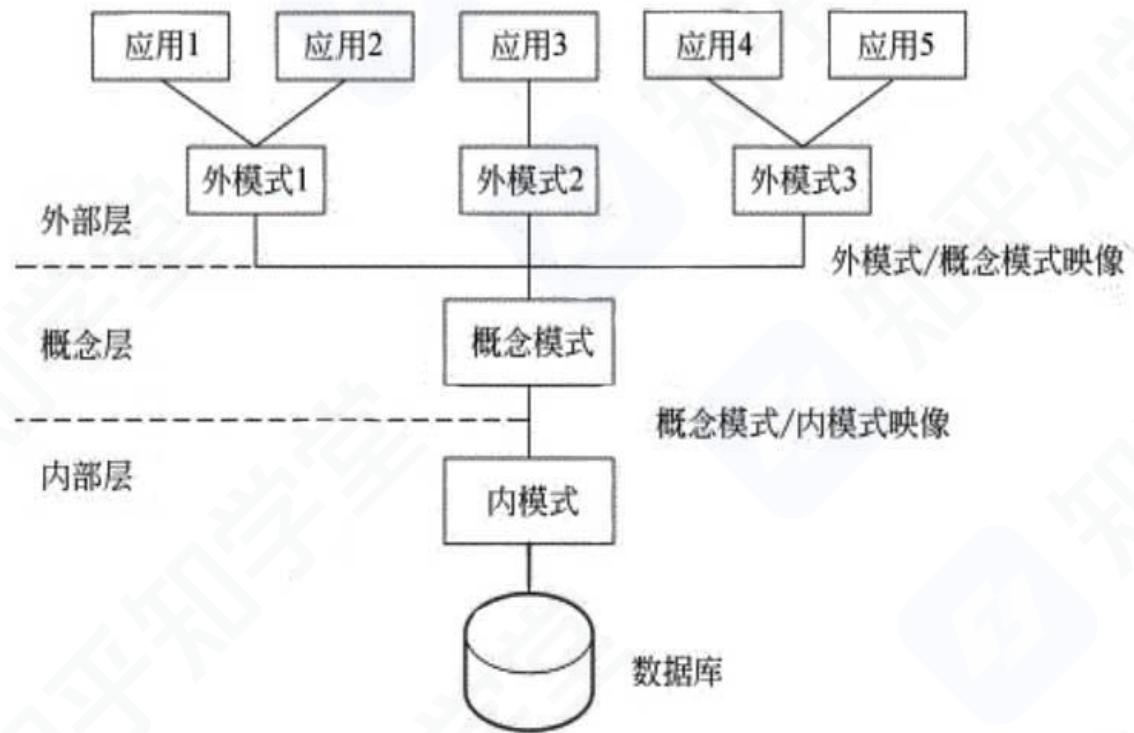
- A.政府对企(事)业单位(G2B) B.政府与政府(G2G) C.企业对政府(B2G) D.政府对居民(G2C)
A.政府对企(事)业单位(G2B) B.政府与政府(G2G) C.企业对政府(B2G) D.政府对居民(G2C)
A.政府对企(事)业单位(G2B) B.政府与政府(G2G) C.企业对政府(B2G) D.政府对居民(G2C)
A.政府对企(事)业单位(G2B) B.政府与政府(G2G) C.企业对政府(B2G) D.政府对居民(G2C)



04

数 据 库

- 本章节在历年考试过程中的分值占比大概是4-6分，考试的内容在选择题方向的题目难度不大，偶尔会出现1分的超纲，在2024年下半年的考试过程中，总共出了6分，但是没有一分超纲，总体上来说这部分的分数还是好拿的
- 被考到知识点有：
 - 基本概念：三级模式-两级映像、数据库设计
 - 数据库模型：E-R模型、关系模型、关系代数、SQL
 - 规范化：函数依赖、键与约束、范式、模式分解
 - 事务并发：并发三种问题、三级封锁协议
 - 数据库新技术：数据库安全与备份、反规范化、分布式数据库、缓存数据库、数据库集群，NoSql
- 在改版之后的考试中，分别考察的知识点为：
 - 2023年11月：范式（传递依赖和2NF,多值依赖和4NF）、数据库语句(having+group by)、三级模式
 - 2024年05月：范式（2NF和部分依赖）、笛卡尔积、事务四大特性、关系代数换算、反规范化设计
 - 2024年11月：约束、SQL注入、自然连接、函数依赖、三级模式、还有一道概念题



三级模式是指数据库管理系统从三个层次来管理数据，分别是外部层（External Level）、概念层（Conceptual Level）和内部层（Internal Level）。这三个层次分别对应三种不同类型的模式，分别是外模式（External Schema）、概念模式（Conceptual Schema）和内模式（Internal Schema）。在外模式与概念模式之间，以及概念模式与内模式之间，还存在映像，即二级映像。

- 外模式：面向应用程序，描述用户的数据视图（View）；
- 内模式（又称为物理模式、存储模式）：面向物理上的数据库，描述数据在磁盘中如何存储；
- 概念模式（又称为模式、逻辑模式）：面向数据库设计人员，描述数据的整体逻辑结构。

数据库关系的三种类型:

- 基本表: 实际存在的表, 实际存储数据的逻辑表示。
- 查询表: 查询结果对应的表
- 视图表: 由基表或其他视图表导出的表, 本身不独立存储, 数据库只存放它的定义, 常称为**虚表**。

视图的优点:

- 视图能简化用户操作
- 视图使用户能以多种角度看待同一数据
- 视图对重构数据库提供了一定程度的逻辑独立性
- 视图可以对机密数据提供安全保护

分布式数据库（Distributed Database）是指数据存储和处理在多台计算机上分布式管理的数据库系统。它通过网络将多个数据库节点连接在一起，形成一个协同工作的整体，旨在提供数据的高可用性、可扩展性以及可靠性。

分布式数据库的特点：

- **数据分布**：数据不是集中存储在单个服务器上，而是分布在多个物理节点上。每个节点可以存储数据的一个子集，或者可能存储数据库的完整副本（如在高可用性系统中）。
- **透明性**：分布式数据库通常对用户和应用程序透明，用户在访问数据库时，不需要知道数据是存储在哪个节点上。这种透明性包括：
 - 位置透明性：用户不关心数据存储的具体位置。
 - 访问透明性：用户可以通过统一的接口访问所有数据。
 - 故障透明性：系统能够自动处理部分节点故障，确保系统的连续性。
- **数据冗余和复制**：为了增强系统的容错能力，分布式数据库可以通过数据复制的方式，在多个节点上保存相同的数据副本，确保数据的高可用性。如果某个节点故障，其他节点的副本可以继续提供服务。
- **数据一致性**：在分布式环境中，如何保证数据的一致性是一个挑战。常见的解决方案包括使用一致性协议（如两阶段提交协议（2PC）或Paxos协议），以及通过CAP定理来做出权衡：一致性、可用性和分区容忍性。
- **可扩展性、高可用性和容错性**：通过冗余和故障转移机制，分布式数据库能够确保系统在部分节点或网络故障的情况下仍能正常工作。

优点:

- 高可用性: 通过冗余和故障恢复机制, 分布式数据库可以提供比传统集中式数据库更高的可用性。
- 可扩展性: 随着数据量增长, 能够通过增加节点进行水平扩展, 轻松应对大规模数据存储和处理的需求。
- 容错性: 系统能够容忍单点故障, 保证数据的可靠性和系统的持续可用性。

缺点:

- 一致性问题: 分布式系统需要处理不同节点之间的数据一致性问题, 特别是在存在网络延迟或节点故障时。解决方案如分布式事务或一致性协议增加了系统的复杂性。
- 性能瓶颈: 虽然可以水平扩展, 但由于网络延迟、节点间的数据同步等因素, 分布式数据库的性能可能不如单节点的集中式数据库。
- 管理复杂性: 分布式数据库的管理需要处理多个节点的协调、故障恢复、数据备份、负载均衡等问题, 管理起来较为复杂。



(1)需求分析: 即分析数据存储的要求，产出物有数据流图、数据字典、需求说明书。获得用户对系统的三个要求：信息要求、处理要求、系统要求。

(2)概念结构设计: 就是设计E-R图，也即实体-联系图。工作步骤包括：选择局部应用、逐一设计分E-R图、E-R图合并。分E-R图进行合并时，它们之间存在的冲突主要有以下3类。

- 属性冲突。同一属性可能会存在于不同的分E-R图中。
- 命名冲突。相同意义的属性，在不同的分E-R图上有着不同的命名，或是名称相同的属性在不同的分E-R图中代表着不同的意义。
- 结构冲突。同一实体在不同的分E-R图中有不同的属性，同一对象在某一分E-R图中被抽象为实体而在另一分E-R图中又被抽象为属性。

(3)逻辑结构设计: 将E-R图，转换成关系模式。工作步骤包括：确定数据模型、将E-R图转换成为指定的数据模型、确定完整性约束和确定用户视图。

(4)物理设计: 步骤包括确定数据分布、存储结构和访问方式。

(5)数据库实施阶段: 根据逻辑设计和物理设计阶段的结果建立数据库，编制与调试应用程序，组织数据入库，并进行试运行。

(6)数据库运行和维护阶段: 数据库应用系统经过试运行即可投入运行，但该阶段需要不断地对系统进行评价、调整与修改。

数据模型的四种分类：

- **关系模型**：是二维表的形式表示的实体-联系模型，是将实体-联系模型转换而来的，经过开发人员设计的；
- **概念模型**：是从用户的角度进行建模的，是现实世界到信息世界的第一抽象，是真正的实体-联系模型。
- **网状模型**：表示实体类型及其实体之间的联系，一个事物和另外几个都有联系，形成一张网。
- **面向对象模型**：是采用面向对象的方法设计数据库，以对象为单位，每个对象包括属性和方法，具有类和继承等特点

数据模型三要素：

- **数据结构**：所研究的对象类型的集合
- **数据操作**：对数据库中各种对象的实例允许执行的操作的集合
- **数据的约束条件**：一组完整性规则的集合

用E-R图来描述概念数据模型，世界是由一组称作实体的基本对象和这些对象之间的联系构成的。

在E-R模型中，使用椭圆表示属性(一般没有)、长方形表示实体、菱形表示联系，联系的两端要填写联系类型，示例如下图：



图 9-9 学校教学管理系统的 E-R 模型

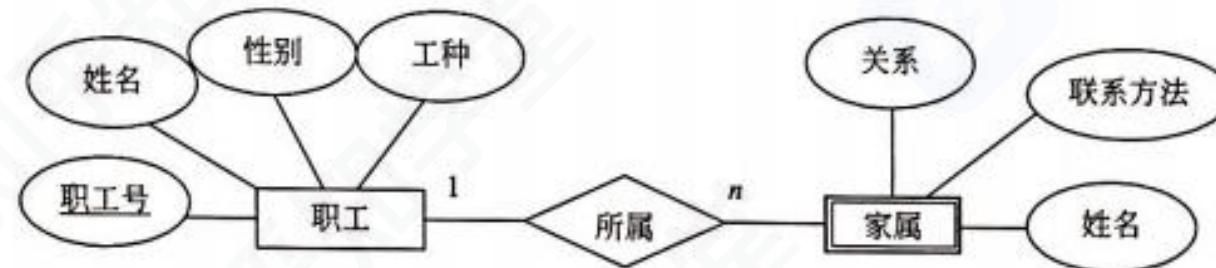


图 9-10 弱实体与依赖联系

关系模型中数据的逻辑结构是一张二维表，由行列组成。用表格结构表达实体集，用外键标识实体间的联系。

- 优点：建立在严格的数学概念基础上；概念单一、结构简单、清晰，用户易懂易用；存取路径对用户透明，从而数据独立性、安全性好，简化数据库开发工作。
- 缺点：由于存取路径透明，查询效率往往不如非关系数据模型。

| S 学生关系 | | | | | T 教师关系 | | | |
|--------|-------|----|-----|-----|--------|-------|-----|-----|
| Sno | Sname | SD | Age | Sex | Tno | Tname | Age | Sex |
| 01001 | 贾皓昕 | IS | 20 | 男 | 001 | 方铭 | 34 | 女 |
| 01002 | 姚勇 | IS | 20 | 男 | 002 | 章雨敬 | 58 | 男 |
| 03001 | 李晓红 | CS | 19 | 女 | 003 | 王平 | 48 | 女 |

| SC 选课 | | | C 课程关系 | | |
|-------|------|-------|--------|-------|------|
| Sno | Cno | Grade | Cno | Cname | Pcno |
| 01001 | C001 | 90 | C001 | MS | |
| 01001 | C002 | 91 | C002 | IC | |
| 01002 | C001 | 95 | C003 | C++ | C002 |
| 01002 | C003 | 89 | C004 | OS | C002 |
| 03001 | C001 | 91 | C005 | DBMS | C004 |

图 9-12 关系模型的实例

并：结果是两张表中所有记录数合并，相同记录只显示一次。

交：结果是两张表中相同的记录。

差： $S1 - S2$ ，结果是 $S1$ 表中有而 $S2$ 表中没有的那些记录。

| 关系S1 | | |
|--------|-------|-------|
| Sno | Sname | Sdept |
| No0001 | Mary | IS |
| No0003 | Candy | IS |
| No0004 | Jam | IS |

| 关系S2 | | |
|--------|--------|-------|
| Sno | Sname | Sdept |
| No0001 | Mary | IS |
| No0008 | Katter | IS |
| No0021 | Tom | IS |

| S1∩S2 (交) | | |
|-----------|-------|-------|
| Sno | Sname | Sdept |
| No0001 | Mary | IS |

| S1 ∪ S2 (并) | | |
|-------------|--------|-------|
| Sno | Sname | Sdept |
| No0001 | Mary | IS |
| No0003 | Candy | IS |
| No0004 | Jam | IS |
| No0008 | Katter | IS |
| No0021 | Tom | IS |

| S1 - S2 (差) | | |
|-------------|-------|-------|
| Sno | Sname | Sdept |
| No0003 | Candy | IS |
| No0004 | Jam | IS |

笛卡尔积: $S1 \times S2$, 产生的结果包括 $S1$ 和 $S2$ 的所有属性列, 并且 $S1$ 中每条记录依次和 $S2$ 中所有记录组合成一条记录, 最终属性列为 $S1 + S2$ 属性列, 记录数为 $S1 * S2$ 记录数。

投影(π): 实际是按条件选择某关系模式中的某列, 列也可以用数字表示。

选择(σ): 实际是按条件选择某关系模式中的某条记录。

| 关系S1 | | |
|--------|-------|-------|
| Sno | Sname | Sdept |
| No0001 | Mary | IS |
| No0003 | Candy | IS |
| No0004 | Jam | IS |

| 关系S2 | | |
|--------|--------|-------|
| Sno | Sname | Sdept |
| No0001 | Mary | IS |
| No0008 | Katter | IS |
| No0021 | Tom | IS |

| S1 × S2 (笛卡尔积) | | | | | |
|----------------|-------|-------|--------|--------|-------|
| Sno | Sname | Sdept | Sno | Sname | Sdept |
| No0001 | Mary | IS | No0001 | Mary | IS |
| No0001 | Mary | IS | No0008 | Katter | IS |
| No0001 | Mary | IS | No0021 | Tom | IS |
| No0003 | Candy | IS | No0001 | Mary | IS |
| No0003 | Candy | IS | No0008 | Katter | IS |
| No0003 | Candy | IS | No0021 | Tom | IS |
| No0004 | Jam | IS | No0001 | Mary | IS |
| No0004 | Jam | IS | No0008 | Katter | IS |
| No0004 | Jam | IS | No0021 | Tom | IS |

| (投影) | |
|--------|-------|
| Sno | Sname |
| No0001 | Mary |
| No0003 | Candy |
| No0004 | Jam |

| (选择) | | |
|--------|-------|-------|
| Sno | Sname | Sdept |
| No0003 | Candy | IS |

自然连接：显示全部的属性列，但是相同属性列只显示一次，显示两个关系模式中属性相同且值相同的记录。

设有关系R、S如下左图所示，自然连接结果如下右图所示：

| A | B | C |
|---|---|---|
| a | b | c |
| b | a | d |
| c | d | e |
| d | f | g |

(a) 关系 R

| A | C | D |
|---|---|---|
| a | c | d |
| d | f | g |
| b | d | g |

(b) 关系 S

| A | B | C | D |
|---|---|---|---|
| a | b | c | d |
| b | a | d | g |

$R \bowtie S$

表 9-2 关系代数运算符

| 运 算 符 | | 含 义 | 运 算 符 | | 含 义 |
|-----------------------|---------------------------------------|---------------------|-----------------------|---|---------------------------------------|
| 集 合 运 算 符 | \cup $-$ \cap \times | 并 差 交 笛卡儿积 | 比 较 运 算 符 | $>$ \geq $<$ \leq $=$ \neq | 大于 大于等于 小于 小于等于 等于 不等于 |
| 专 门 的 关 系 运 算 符 | σ π \bowtie $+$ | 选择 投影 连接 除 | 逻 辑 运 算 符 | \neg \wedge \vee | 非 与 或 |

【例 9.4】设有关系 R 、 S 如下所示，请求出 $R \cup S$ 、 $R - S$ 、 $R \times S$ 、 $\pi_{A,C}(R)$ 、 $\sigma_{A>B}(R)$ 和 $\sigma_{3<4}(R \times S)$ 。

| A | B | C |
|-----|-----|-----|
| a | b | c |
| b | a | d |
| c | d | e |
| d | f | g |

关系 R

| A | B | C |
|-----|-----|-----|
| b | a | d |
| d | f | g |
| f | h | k |

关系 S

| $R \cup S$ | | |
|------------|----------|----------|
| <i>A</i> | <i>B</i> | <i>C</i> |
| a | b | c |
| b | a | d |
| c | d | e |
| d | f | g |
| f | h | k |

| $R-S$ | | |
|----------|----------|----------|
| <i>A</i> | <i>B</i> | <i>C</i> |
| a | b | c |
| c | d | e |

| $\pi_{A,C}(R)$ | |
|----------------|----------|
| <i>A</i> | <i>C</i> |
| a | c |
| b | d |
| c | e |
| d | g |

| $\sigma_{A>B}(R)$ | |
|-------------------|----------|
| <i>A</i> | <i>B</i> |
| b | a |
| | D |

| $R \times S$ | | | | | |
|--------------|------------|------------|------------|------------|------------|
| <i>R.A</i> | <i>R.B</i> | <i>R.C</i> | <i>S.A</i> | <i>S.B</i> | <i>S.C</i> |
| a | b | c | b | a | d |
| a | b | c | d | f | g |
| a | b | c | f | h | k |
| b | a | d | b | a | d |
| b | a | d | d | f | g |
| b | a | d | f | h | k |
| c | d | e | b | a | d |
| c | d | e | d | f | g |
| c | d | e | f | h | k |
| d | f | g | b | a | d |
| d | f | g | d | f | g |
| d | f | g | f | h | k |

| $\sigma_{3 < 4}(R \times S)$ | | | | | |
|------------------------------|------------|------------|------------|------------|------------|
| <i>R.A</i> | <i>R.B</i> | <i>R.C</i> | <i>S.A</i> | <i>S.B</i> | <i>S.C</i> |
| a | b | c | d | f | g |
| a | b | c | f | h | k |
| b | a | d | f | h | k |
| c | d | e | f | h | k |

图 9-15 运算结果

给定关系 $R(A, B, C, D)$ 和关系 $S(C, D, E)$ ，对其进行自然连接运算 $R \bowtie S$ 后的属性列为（ ）个；

与 $\sigma_{R.B>S.E}(R \bowtie S)$ 等价的关系代数表达式为（ ）。 |

- A. 4 B. 5 C. 6 D. 7 ↓
-

- A. $\sigma_{2>7}(R \times S)$ B. $\pi_{1, 2, 3, 4, 7}(\sigma_{‘2’ > ‘7’ \Delta 3=5 \Delta 4=6}(R \times S)) \downarrow$
C. $\sigma_{‘2’ > ‘7’}(R \times S)$ D. $\pi_{1, 2, 3, 4, 7}(\sigma_{2>7 \Delta 3=5 \Delta 4=6}(R \times S)) \downarrow$

超键：能唯一标识记录的属性集合(可能不止一个)

候选键：最小的超键，用于作为主键(通常只有一个)

主属性：除候选键外的具有代表意义的非重复且非衍生属性

主键：任选一个候选键，即可作为主键。

外键：其他表中的主键。

实体完整性约束：即主键约束，主键值不能为空，也不能重复。

参照完整性约束：即外键约束，外键必须是其他表中已经存在的主键的值，或者为空。

用户自定义完整性约束：自定义表达式约束，如设定年龄属性的值必须在0到180之间。

数据库系统-键与约束



超键：能够唯一标识一条记录的属性或属性集

| 学号 | 姓名 | 性别 | 年龄 | 系别 | 专业 |
|----------|-----|----|----|-----|------|
| 20020612 | 李辉 | 男 | 20 | 计算机 | 软件开发 |
| 20060613 | 张明 | 男 | 18 | 计算机 | 软件开发 |
| 20060614 | 王小玉 | 女 | 19 | 物理 | 力学 |
| 20060615 | 李淑华 | 女 | 17 | 生物 | 动物学 |
| 20060616 | 赵静 | 男 | 21 | 化学 | 食品化学 |

| 学号 | 姓名 | 性别 | 年龄 | 系别 | 专业 |
|----------|-----|----|----|-----|------|
| 20020612 | 李辉 | 男 | 20 | 计算机 | 软件开发 |
| 20060613 | 张明 | 男 | 18 | 计算机 | 软件开发 |
| 20060614 | 王小玉 | 女 | 19 | 物理 | 力学 |
| 20060615 | 李淑华 | 女 | 17 | 生物 | 动物学 |
| 20060616 | 赵静 | 男 | 21 | 化学 | 食品化学 |

| 学号 | 姓名 | 性别 | 年龄 | 系别 | 专业 |
|----------|-----|----|----|-----|------|
| 20020612 | 李辉 | 男 | 20 | 计算机 | 软件开发 |
| 20060613 | 张明 | 男 | 18 | 计算机 | 软件开发 |
| 20060614 | 王小玉 | 女 | 19 | 物理 | 力学 |
| 20060615 | 李淑华 | 女 | 17 | 生物 | 动物学 |
| 20060616 | 赵静 | 男 | 21 | 化学 | 食品化学 |

| 学号 | 姓名 | 性别 | 年龄 | 系别 | 专业 |
|----------|-----|----|----|-----|------|
| 20020612 | 李辉 | 男 | 20 | 计算机 | 软件开发 |
| 20060613 | 张明 | 男 | 18 | 计算机 | 软件开发 |
| 20060614 | 王小玉 | 女 | 19 | 物理 | 力学 |
| 20060615 | 李淑华 | 女 | 17 | 生物 | 动物学 |
| 20060616 | 赵静 | 男 | 21 | 化学 | 食品化学 |

| 学号 | 姓名 | 性别 | 年龄 | 系别 | 专业 |
|----------|-----|----|----|-----|------|
| 20020612 | 李辉 | 男 | 20 | 计算机 | 软件开发 |
| 20060613 | 张明 | 男 | 18 | 计算机 | 软件开发 |
| 20060614 | 王小玉 | 女 | 19 | 物理 | 力学 |
| 20060615 | 李淑华 | 女 | 17 | 生物 | 动物学 |
| 20060616 | 赵静 | 男 | 21 | 化学 | 食品化学 |

候选键：能够唯一标识一条记录的最小属性集

| 学号 | 姓名 | 性别 | 年龄 | 系别 | 专业 |
|----------|-----|----|----|-----|------|
| 20020612 | 李辉 | 男 | 20 | 计算机 | 软件开发 |
| 20060613 | 张明 | 男 | 18 | 计算机 | 软件开发 |
| 20060614 | 王小玉 | 女 | 19 | 物理 | 力学 |
| 20060615 | 李淑华 | 女 | 17 | 生物 | 动物学 |
| 20060616 | 赵静 | 男 | 21 | 化学 | 食品化学 |

| 学号 | 姓名 | 性别 | 年龄 | 系别 | 专业 |
|----------|-----|----|----|-----|------|
| 20020612 | 李辉 | 男 | 20 | 计算机 | 软件开发 |
| 20060613 | 张明 | 男 | 18 | 计算机 | 软件开发 |
| 20060614 | 王小玉 | 女 | 19 | 物理 | 力学 |
| 20060615 | 李淑华 | 女 | 17 | 生物 | 动物学 |
| 20060616 | 赵静 | 男 | 21 | 化学 | 食品化学 |

| 学号 | 姓名 | 性别 | 年龄 | 系别 | 专业 |
|----------|-----|----|----|-----|------|
| 20020612 | 李辉 | 男 | 20 | 计算机 | 软件开发 |
| 20060613 | 张明 | 男 | 18 | 计算机 | 软件开发 |
| 20060614 | 王小玉 | 女 | 19 | 物理 | 力学 |
| 20060615 | 李淑华 | 女 | 17 | 生物 | 动物学 |
| 20060616 | 赵静 | 男 | 21 | 化学 | 食品化学 |

| 学号 | 姓名 | 性别 | 年龄 | 系别 | 专业 |
|----------|-----|----|----|-----|------|
| 20020612 | 李辉 | 男 | 20 | 计算机 | 软件开发 |
| 20060613 | 张明 | 男 | 18 | 计算机 | 软件开发 |
| 20060614 | 王小玉 | 女 | 19 | 物理 | 力学 |
| 20060615 | 李淑华 | 女 | 17 | 生物 | 动物学 |
| 20060616 | 赵静 | 男 | 21 | 化学 | 食品化学 |

主属性：候选码所有属性的并集

| 学号 | 姓名 | 性别 | 年龄 | 系别 | 专业 |
|----------|-----|----|----|-----|------|
| 20020612 | 李辉 | 男 | 20 | 计算机 | 软件开发 |
| 20060613 | 张明 | 男 | 18 | 计算机 | 软件开发 |
| 20060614 | 王小玉 | 女 | 19 | 物理 | 力学 |
| 20060615 | 李淑华 | 女 | 17 | 生物 | 动物学 |
| 20060616 | 赵静 | 男 | 21 | 化学 | 食品化学 |

| 学号 | | | | | |
|----|--|--|--|--|--|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

| 学号 | 姓名 | 性别 | 年龄 | 系别 | 专业 |
|----------|-----|----|----|-----|------|
| 20020612 | 李辉 | 男 | 20 | 计算机 | 软件开发 |
| 20060613 | 张明 | 男 | 18 | 计算机 | 软件开发 |
| 20060614 | 王小玉 | 女 | 19 | 物理 | 力学 |
| 20060615 | 李淑华 | 女 | 17 | 生物 | 动物学 |
| 20060616 | 赵静 | 男 | 21 | 化学 | 食品化学 |

| 姓名 | | | | | |
|----|--|--|--|--|--|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

| | | |
|------|----|----|
| 主属性： | 学号 | 姓名 |
| | | |
| | | |
| | | |
| | | |

函数依赖：给定一个X，能唯一确定一个Y，就称X决定(确定)Y，或者说Y依赖于X。

- 例如： $Y=X*X$ 函数，此时X能确定Y的值，但是Y无法确定X的值，比如 $x=2, y=4$ ，但是 $y=4$ 无法确定 $x=2$ 。

函数依赖又可扩展以下两种规则：

- 部分函数依赖：部分函数依赖是指在一个关系模式中，一个非主属性（即非码属性）依赖于关系模式中的某个码的一部分，而不是整个码。
- 传递函数依赖：指的是当一个非主属性依赖于另一个非主属性，而这两个非主属性之间又通过主属性之间的依赖关系联系起来时，就存在传递依赖。

部分函数依赖举例：假设有一个关系模式 $R = \{A, B, C\}$ ，其中 $\{A, B\}$ 是 R 的码，如果某个非码属性（比如 C ）只依赖于 R 的码中的一部分（比如 A ），而不是整个码（即 $\{A, B\}$ ），那么我们说 C 对 A 部分函数依赖（记为 $A \Rightarrow\!\!> C$ ）。

传递函数依赖举例：假设有一个订单表（Orders），包含字段{订单号, 客户ID, 客户姓名, 订单日期}，并且存在以下函数依赖：订单号 → 客户ID、客户ID → 客户姓名、订单号 → 订单日期，在这个例子中，“订单号”是“客户ID”和“订单日期”的候选键。存在传递函数依赖：订单号 → 客户ID → 客户姓名。这意味着如果我们知道订单号，我们可以确定客户ID，进而确定客户姓名。

第一范式 (1NF)：若关系模式R的每一个分量是不可再分的数据项，则关系模式R属于第一范式。通俗地说，第一范式就是表中不允许有小表的存在。比如，对于如下的员工表，就不属于第一范式：

| 员工编号 | 员工姓名 | 出生日期 | 薪资/月 | | 所属部门 |
|------|------|----------|--------|------|------|
| | | | 基本工资/月 | 补贴/月 | |
| 1 | 王红 | 19900908 | 9000 | 1000 | 101 |
| ... | ... | ... | ... | ... | ... |

例如，供应商和它所提供的零件信息，关系模式FIRST和函数依赖集F如下：

- FIRST(Sno,Sname,Status,City,Pno,Qty)
- F={Sno→Sname,Sno→Status,Status→City,(Sno,Pno)→Qty}

表 9-8 FIRST

| Sno | Sname | Status | City | Pno | Qty |
|-----|-------|--------|------|-----|-----|
| S1 | 精益 | 20 | 天津 | P1 | 200 |
| S1 | 精益 | 20 | 天津 | P2 | 300 |
| S1 | 精益 | 20 | 天津 | P3 | 480 |
| S2 | 盛锡 | 10 | 北京 | P2 | 168 |
| S2 | 盛锡 | 10 | 北京 | P3 | 500 |
| S3 | 东方红 | 30 | 北京 | P1 | 300 |
| S3 | 东方红 | 30 | 北京 | P2 | 280 |
| S4 | 泰达 | 40 | 上海 | P2 | 460 |

第二范式 (2NF)：在1NF的基础上，要求数据库表中的每个非主属性完全依赖于码，例如，FIRST关系中的码是Sno、Pno，而 $Sno \rightarrow Status$ ，因此非主属性Status部分函数依赖于码，故非2NF的。

表 9-8 FIRST

| Sno | Sname | Status | City | Pno | Qty |
|-----|-------|--------|------|-----|-----|
| S1 | 精益 | 20 | 天津 | P1 | 200 |
| S1 | 精益 | 20 | 天津 | P2 | 300 |
| S1 | 精益 | 20 | 天津 | P3 | 480 |
| S2 | 盛锡 | 10 | 北京 | P2 | 168 |
| S2 | 盛锡 | 10 | 北京 | P3 | 500 |
| S3 | 东方红 | 30 | 北京 | P1 | 300 |
| S3 | 东方红 | 30 | 北京 | P2 | 280 |
| S4 | 泰达 | 40 | 上海 | P2 | 460 |

若此时将FIRST关系分解为FIRST1(Sno,Sname,Status,City)和FIRST2(Sno,Pno, Qty)。分解后的关系模式FIRST1的码为Sno,非主属性Sname、Status、City完全依赖于码Sno,所以属于2NF；关系模式FIRST2的码为Sno、Pno,非主属性Qty完全依赖于码，所以也属于2NF。

第三范式（3NF）：在2NF的基础上，消除了非主属性对码的传递函数依赖，则称为3NF。例如，FIRST1不属于3NF，因为在分解后的关系模式FIRST1中有 $Sno \rightarrow Status$, $Status \rightarrow City$ ，存在着非主属性City传递依赖于码Sno。若此时将FIRST1继续分解为：FIRST1.1 (Sno,Sname,Status)、FIRST1.2 (Status,City)，通过上述分解，数据库模式FIRST转换为FIRST1.1(Sno,Sname,Status)、FIRST1.2(Status,City)和FIRST2(Sno,Pno,Qty)3个子模式。由于这3个子模式都达到了3NF，因此称分解后的数据库模式达到了3NF。

可以证明，3NF的模式必是2NF的模式。产生冗余和异常的两个重要原因是部分依赖和传递依赖。因为3NF模式中不存在非主属性对码的部分函数依赖和传递函数依赖，所以具有较好的性能。对于非3NF的1NF、2NF其性能弱，一般不宜作为数据库模式，通常要将它们变换成为3NF或更高级别的范式，这个变换过程称为“关系模式的规范化处理”。

第四范式：在**3NF**的基础上，要求一个表的主键只对应一个多值。例如，学生信息表(学生ID, 住址, 电话号码)，这个表中住址和电话号码表示学生可以有多个，它们与学生存在多值依赖关系。这个表不满足4NF，因为在这个表中，住址和电话号码是独立的多值依赖，这意味着它们各自都直接依赖于学生ID，并且彼此之间是独立的。解决办法有两种，一种是通过程序来控制，二种是拆成两张表，每张表里面只拥有一个值。

BC范式(BCNF): 规范化数据库设计的一种方法，它对关系型数据库中的表进行分解，其符合第三范式 (3NF) ，同时尽量避免数据冗余和不一致性，提高数据的可靠性和完整性。

假设仓库管理关系表(仓库ID, 存储物品ID, 管理员ID, 数量)，且有一个管理员只在一个仓库工作；一个仓库可以存储多种物品。此关系模式已经属于了3NF，那么这个关系模式是否存在问题呢？我们来看以下几种操作：

- 删除异常：当仓库被清空后，所有”存储物品ID”和”数量”信息被删除的同时，”仓库ID”和”管理员ID”信息也被删除了。
- 插入异常：当仓库没有存储任何物品时，无法给仓库分配管理员。
- 更新异常：如果仓库换了管理员，则表中所有行的管理员ID都要修改。

解决方案：把仓库管理关系表分解为二个关系表：

- 仓库管理：(仓库ID, 管理员ID);
- 仓库：(仓库ID, 存储物品ID, 数量)。

这样的数据库表是符合BCNF范式的，消除了删除异常、插入异常和更新异常。

➤ 函数依赖的公理系统(Armstrong阿姆斯特朗公理)是一组在关系数据库理论中用于推导属性依赖的基本规则。提供了一种形式化的方法，用于推导关系数据库中的所有属性依赖。通过使用这套公理，可以理解和掌握一个数据库的所有潜在的属性依赖，从而帮助设计合理的数据库模式，确保数据的一致性与完整性。

➤ Armstrong公理包括三条基本规则，它们分别是：

- 自反律：若属性集Y是属性集X的子集 ($Y \subseteq X \subseteq U$)，则 $X \rightarrow Y$ 成立。这表示如果一个属性集合Y是另一个属性集合X的子集，那么X可以决定Y。
- 增广律：若 $X \rightarrow Y$ 成立，且属性集Z是属性集U的子集 ($Z \subseteq U$)，则 $XZ \rightarrow YZ$ 也成立。这表示如果X可以决定Y，那么在任何集合Z加入到X和Y两侧的情况下，依赖关系仍然成立。
- 传递律：若 $X \rightarrow Y$ 及 $Y \rightarrow Z$ 成立，则 $X \rightarrow Z$ 也成立。这表示如果X可以决定Y，且Y可以决定Z，那么X也可以决定Z。

➤ 根据上述3条推理规则又可推出下述3条推理规则：

- 合并律：对于任意属性集合X、Y和Z，如果 $X \rightarrow Y$ 和 $X \rightarrow Z$ ，那么 $X \rightarrow YZ$ 。
- 分解律：对于任意属性集合X、Y和Z，如果 $X \rightarrow YZ$ ，则 $X \rightarrow Y$ 和 $X \rightarrow Z$ 。
- 伪传递规则：若 $X \rightarrow Y$ 成立，且 $WY \rightarrow Z$ 成立，则 $XW \rightarrow Z$ 也成立。这表示如果X可以决定Y，且WY可以决定Z，那么XW也可以决定Z。这条规则适用于处理那些不仅依赖于单个属性集合的复杂函数依赖关系。

反规范化技术：规范化设计后，数据库设计者希望牺牲部分规范化来提高性能。

- 采用反规范化技术的益处：降低连接操作的需求、降低外码和索引的数目，还可能减少表的数目，能够提高查询效率。
- 可能带来的问题：数据的重复存储，浪费了磁盘空间；可能出现数据的完整性问题，为了保障数据的一致性，增加了数据维护的复杂性，会降低修改速度。

具体方式：

- 增加冗余列：在多个表中保留相同的列，通过增加数据冗余减少或避免查询时的连接操作。
- 增加派生列：在表中增加可以由本表或其它表中数据计算生成的列，减少查询时的连接操作并避免计算或使用集合函数。
- 重新组表：如果许多用户需要查看两个表连接出来的结果数据，则把这两个表重新组成一个表来减少连接而提高性能。
- 水平分割表：根据一列或多列数据的值，把数据放到多个独立的表中，主要用于表数据规模很大、表中数据相对独立或数据需要存放到多个介质上时使用。
- 垂直分割表：对表进行分割，将主键与部分列放到一个表中，主键与其它列放到另一个表中，在查询时减少I/O次数。

模式分解：是关系数据库规范化设计中的一个重要概念。它指的是通过对关系模式进行拆分，来消除模式中的混合组合依赖，达到将模式分解为更小的模式的过程。一般分为以下两种：

- 是否保持函数依赖分解：对于关系模式R，有依赖集F，若对R进行分解，分解出来的多个关系模式，保持原来的依赖集不变，则为保持函数依赖的分解。另外，注意要消除掉冗余依赖(如传递依赖)
- 有损无损分解：分解后的关系模式能够还原出原关系模式，就是无损分解，不能还原就是有损

是否保持函数依赖：设原关系模式 $R(A, B, C)$ ，依赖集 $F(A \rightarrow B, B \rightarrow C, A \rightarrow C)$ ，将其分解为两个关系模式 $R_1(A, B)$ 和 $R_2(B, C)$ ，此时 R_1 中保持依赖 $A \rightarrow B$ ， R_2 保持依赖 $B \rightarrow C$ ，说明分解后的 R_1 和 R_2 是保持函数依赖的分解，因为 $A \rightarrow C$ 这个函数依赖实际是一个冗余依赖，可以由前两个依赖传递得到，因此不需要管。

是否无损分解：如果R的分解为 $p=\{R_1, R_2\}$ ， F 为R所满足的函数依赖集合，分解 p 具有无损连接性的充分必要条件是 $R_1 \cap R_2 \rightarrow (R_1 - R_2)$ 或者 $R_1 \cap R_2 \rightarrow (R_2 - R_1)$ 。

假设关系模式 $R(U, F)$, 属性集 $U=\{A, B, C\}$, 函数依赖集 $F=\{A \rightarrow B, B \rightarrow C\}$ 。若将其分解为 $p=\{R1(U1, F1), R2(U2, F2)\}$, 其中 $U1=\{A, B\}$, $U2=\{A, C\}$ 。那么, 分解 p ()。

- A.有损连接但保持函数依赖
- B.既无损连接又保持函数依赖
- C.有损连接且不保持函数依赖
- D.无损连接但不保持函数依赖

给定关系模式 $R<U, F>$, $U=\{A, B, C, D, E\}$, $F=\{B \rightarrow A, D \rightarrow A, A \rightarrow E, AC \rightarrow B\}$, 则 R 的候选关键字为(), 分解 $p=\{R1(ABCE), R2(CD)\}()$ 。

- A.CD
 - B.ABD
 - C.ACD
 - D.ADE
-
- A.具有无损连接性, 且保持函数依赖
 - B.不具有无损连接性, 但保持函数依赖
 - C.具有无损连接性, 但不保持函数依赖
 - D.不具有无损连接性, 也不保持函数依赖

事务：由一系列DML操作组成，这些操作，要么全做，要么全不做，它从第一个DML操作开始， rollback、commit或者DDL结束，拥有以下四种特性，详解如下：

- (操作)原子性：要么全做，要么全不做，例如银行转账，在没到最后一步完成转账之前，前面的所有操作都是无效的。
- (数据)一致性：事务发生后数据是一致的，例如银行转账，不会存在A账户转出，但是B账户没收到的情况。
- (执行)隔离性：任一事务的更新操作直到其成功提交的整个过程对其他事务都是不可见的，不同事务之间是隔离的，互不干涉。
- (改变)持续性：事务操作的结果是持续性的。

事务是并发控制的前提条件，并发控制就是控制不同的事务并发执行，提高系统效率，但是并发控制中存在下面三个问题：

丢失更新：事务1对数据A进行了修改并写回，事务2也对A进行了修改并写回，此时事务2写回的数据会覆盖事务1写回的数据，就丢失了事务1对A的更新。即对数据A的更新会被覆盖。

不可重复读：事务2读A，而后事务1对数据A进行了修改并写回，此时若事务2再读A，发现数据不对。即一个事务重复读A两次，会发现数据A有误。

读脏数据：事务1对数据A进行了修改后，事务2读数据A，而后事务1回滚，数据A恢复了原来的值，那么事务2对数据A做的事是无效的，读到了脏数据。

◆ 丢失更新

| T1 | T2 |
|------------------------------|-----------------------|
| ①读A=10 ② ③A=A-5写回 ④ | 读A=10 A=A-8 写回 |

◆ 不可重复读

| T1 | T2 |
|--|--------------------------|
| ①读A=20 读B=30 求和=50 ② ③读A=70 读B=30 求和=100 (验算不对) | 读A=20 A←A+50 写A=70 |

◆ 读“脏”数据

| T1 | T2 |
|--|-------|
| ①读A=20 A←A+50 写回70 ② ③ROLLBACK A恢复为20 | 读A=70 |

X锁是排它锁(写锁)。若事务T对数据对象A加上X锁，则只允许T读取和修改A，其他事务都不能再对A加任何类型的锁，直到T释放A上的锁。

S锁是共享锁(读锁)。若事务T对数据对象A加上S锁，则只允许T读取A，但不能修改A，其他事务只能再对A加S锁(也即能读不能修改)，直到T释放A上的S锁。

共分为三级封锁协议，如下：

- 一级封锁协议：事务在修改数据R之前必须先对其加X锁，直到事务结束才释放。可解决丢失更新问题。

| T1 | T2 |
|---|---|
| <p>① 对A加写锁</p> <p>②</p> <p>③ 读A=10</p> <p>④ A=A-5写回</p> <p>⑤ 释放对A的写锁</p> <p>⑥</p> <p>⑦</p> <p>⑧</p> | <p>对A加写锁 等待</p> <p>等待</p> <p>等待</p> <p>读A= 5</p> <p>A=A-8 写回</p> <p>释放对A的写锁</p> |

- 二级封锁协议：一级封锁协议的基础上加上事务T在读数据R之前必须先对其加S锁，读完后即可释放S锁。可解决丢失更新、读脏数据问题。

| T1 | T2 |
|--|--|
| <p>① 对A加写锁 ② 读A=20 ③ $A \leftarrow A + 50$ ④ 写回70 ⑤ ⑥ ROLLBACK ⑦ A恢复为20 ⑧</p> | <p>对A加读锁 等待 等待 读A=20 释放对A的读锁</p> |

- 三级封锁协议：一级封锁协议加上事务T在读取数据R之前先对其加S锁，直到事务结束才释放。可解决丢失更新、读脏数据、数据重复读问题。

| T1 | T2 |
|--|--|
| <p>① 对A与B加S锁（读锁） 读A=20 读B=30 求和=50 ② ③ 读A=20 读B=30 求和=50 释放对A和B的读锁</p> | <p>对A加X锁（写锁） 注：由于A已加了读锁，所以等待 等待 等待 等待 读A=20 $A \leftarrow A + 50$ 写A=70 释放对A的写锁</p> |

当多个事务并发执行时，任一事务的更新操作直到其成功提交的整个过程对其他事务都是不可见的，这一性质通常被称为事务的()。

- A.原子性
- B.一致性
- C.隔离性
- D.持久性

若事务T1对数据D1加了共享锁，事务T2、T3分别对数据D2、D3加了排它锁，则事务T1对数据();事务T2对数据()

- | | |
|----------------------|----------------------|
| A.D2、D3加排它锁都成功 | B.D2、D3加共享锁都成功 |
| C.D2加共享锁成功， D3加排它锁失败 | D.D2、D3加排它锁和共享锁都失败 |
| A.D1、D3加共享锁都失败 | B.D1、D3加共享锁都成功 |
| C.D1加共享锁成功， D3加排它锁失败 | D.D1加排它锁成功， D3加共享锁失败 |

SQL(Structured Query Language): 是关系数据库管理系统的标准查询语言，用于存取数据以及查询、更新和管理关系数据库

SQL的主要功能包括：

1. 数据定义语言(DDL): 用于定义数据库对象，如表、视图、索引等。包括CREATE、ALTER、DROP等语句。
2. 数据操作语言(DML): 用于对数据库中表的数据进行增删改操作。包括INSERT、UPDATE、DELETE等语句。
3. 数据查询语言(DQL): 用于查询数据库中表的记录。主要是SELECT语句及其子句
4. 事务控制语言(TCL): 用于管理数据库事务。包括COMMIT、ROLLBACK、SAVEPOINT等语句。
5. 数据控制语言(DCL): 用于管理数据库的权限和安全性。主要是GRANT和REVOKE语句。

数据库查询 (**DQL**) 语法: select ... from ... where ...order by...limit...

1. 排序order by, 默认为升序, 降序要加关键字DESC: select * from t1 order by sno
2. 分页limit, limit startIndex,pageSize, startIndex代表从第几项开始, pageSize代表展示多少项数据, startIndex从0开始算: select * from t1 limit 0,10
3. 分组查询group by, 分组时要注意select后的列名要适应分组, having为分组查询附加条件: select sno, avg(score) from student group by sno having(avg(score)>60)

常见的**DML**操作:

1. 更名运算as: select sno as "学号"from table
2. 字符串匹配: like, %匹配多个字符串, _匹配任意一个字符串: select * from t1 where sname like'a_'
3. 数据库插入: insert into values(): insert into t1 values('a', 66)
4. 数据库删除: delete from..where: delete from t1 where sno=4
5. 数据库修改: update ... set ... where ...: update t1 set sname='aa' where sno=3

某销售公司数据库的零件关系P(零件号, 零件名称, 供应商, 供应商所在地, 库存量), 函数依赖集 $F=\{\text{零件号} \rightarrow \text{零件名称}, (\text{零件号}, \text{供应商}) \rightarrow \text{库存量}, \text{供应商} \rightarrow \text{供应商所在地}\}$ 。零件关系P属于()。查询各种零件的平均库存量、最多库存量与最少库存量之间差值的SQL语句如下: `SELECT 零件号, () FROM P ()`

A.1NF

B.2NF

C.3NF

D.4NF

A.`AVG(库存量) AS 平均库存量, MAX(库存量)-MIN(库存量) AS 差值`

B.`平均库存量 AS AVG(库存量), 差值 AS MAX(库存量)-MIN(库存量)`

C.`AVG(库存量) AS 平均库存量, MAX(库存量)-MIN(库存量) AS 差值`

D.`平均库存量 AS AVG(库存量), 差值 AS MAX(库存量)-MIN(库存量)`

A.`ORDER BY 供应商`

B.`ORDER BY 零件号`

C.`GROUP BY 供应商`

D.`GROUP BY 零件号`

应用程序与数据交互: 应用程序通过程序接口来访问数据库并进行操作。

- 库函数级别访问接口: 最底层的访问方式, 比如使用OCI来访问数据库, 开发效率低, 依赖特定的数据库, 学习难度高
- 嵌入SQL访问接口: 直接将SQL语句写入到变成语句的源码中, 需要数据库厂商提供一个嵌入式SQL的预编译器, 方便对嵌入式SQL的代码进行预编译并操作数据库, 操作起来比较麻烦, 需要和DBMS进行交互, 而且性能也不高
- 通用数据接口标准: 为不同的数据库提供统一的接口(ODBC), 允许我们在程序里面书写SQL语句进行操作数据库, 不通过DBMS, 直接使用SQL操作数据库中的数据, 但是直接使用ODBC比较麻烦, 后来又发展出了DAO、RDO和ADO这些数据库访问接口以及专门给.NET使用的ADO和Java使用的JDBC
- ORM访问接口: 使用框架技术让对象和数据库中的表产生映射, 让我们直接操作对象就能修改表数据, 典型的有Hibernate、MyBatis、JPA等框架, 这种开发效率最高, 降低了程序员对数据库知识的要求

NoSQL: Non-Relational或者是Not Only SQL, 泛指非关系型数据库, 区分开关系型数据库, 并且不保证关系型数据库的ACID特性。

NoSQL的分类:

- 列式存储数据库: 跟传统的关系型数据库一样, 数据按行列进行存储, 这种类别通常用来应对分布式数据库的存储海量数据, 比如HBase
- 键值对存储数据库: 以key-value的形式来存储数据, 特点是简单、容易部署, 比如Redis
- 文档型数据库: 类似于键值对数据库, 可以看成是键值对数据库的升级版, 允许嵌套键值, 处理复杂数据的时候比传统的键值对存储效率高, 比如MongoDB
- 图数据库: 使用灵活的图形模型来存储数据, 能够拓展到多个服务器上, 适合存储通过图进行建模的数据, 比如社交网络、交通网络等, 常见的产品有Neo4J

NoSQL的特征: 易拓展、大数据量、高性能、灵活的数据模型、高可用

NoSQL的框架分层(从下至上): 数据持久层、数据分布层、数据逻辑模型层和接口层, 层次之间相辅而成, 协调工作

NoSQL适用于哪些场景: 数据模型比较简单、需要灵活性更强的系统、对数据性能要求高、不需要高度的数据一致性

键值对数据库

- Redis、MemCache
- 应用于内容缓存、处理大数据量的高访问负载、日志等
- 查找速度快但是数据无结构化

文档型数据库

- ConthDB、MongoDB（基于分布式文件存储的数据库，C++编写，主要用于处理大量文档；它是一种介于关系型数据库和非关系型数据库的中间产品，是nosql中功能最丰富、最像关系型数据库的非关系型数据库）
- 应用于web应用
- 数据结构要求不严格、表结构可变、不需要预定义表结构但查询性能不高且缺少统一查询语言

列存储数据库

- HBase（大数据）、Doris
- 应用于分布式文件系统
- 查找速度快、可扩展性强但功能相对局限

图关系数据库（不是存图形，而是存关系，比如：朋友圈、社交网络、广告推荐）

- Neo4j、InfoGrid
- 应用于社交网络、推荐系统
- 可以利用图结构相关的算法但是计算时需要全部图，导致不太好做分布式集群



真题



2024年05月第7题：以下关于数据库范式的说法中正确的是()

- A.数据库1NF要求每个属性都是非空属性
- B.数据库3NF要求不存在主属性之间的部分依赖和传递依赖
- C.数据库2NF每一个非主属性依赖于主键的一部分
- D.数据库2NF每一个非主属性完全依赖主键

2024年05月第8题：假设两个关系分别有m和n个元组，当两个关系进行笛卡尔积运算后，其结果的元组个数为()

- A. $m+n$
- B. $m-n$
- C. $m*n$
- D. m/n

2024年05月第9题：在数据库中，以下()不属于事务的特点

- A.原子性
- B.并发性
- C.一致性
- D.隔离性

2024年05月第10题：在数据库的关系运算中，R和S的交集可以用下列()表达式来代替。

- A. $R-(R-S)$
- B. $S-(R-S)$
- C. $R-(S-R)$
- D. $(R-S)-R$

2024年05月第11题：反规范化是指在()阶段有意地引入冗余，以提高数据库的读性能

- A.需求分析 B.概念结构设计 C.逻辑结构设计 D.物理结构设计

2023年11月第5题：关系模型（员工姓名，工资级别，工资金额），其中员工姓名是主键，工资级别决定工资金额，请问满足()范式

- A BCNF B 4NF C 3NF D 2NF

2023年11月第6题：若关系模式R(U)中，X、Y、Z是U的子集，并且 $Z=U-X-Y$ 。当且仅当对R(U)的任何一个关系T，给定一对(x,z)值，有一组Y的值，这组值仅仅决定于x值而与z值无关，则称“Y多值依赖于”或“X多值决定Y”成立。记为：
 $X \rightarrow\!\!\!-\rightarrow Y$ 。关系模式R ∈ 1NF, 若对于R的每个非平凡多值依赖 $X \rightarrow\!\!\!-\rightarrow Y$ ，且Y不含于X时，X必定含有码，则关系模式R(U,F) ∈ ()

- A.3NF B.BCNF C.4NF D.5NF

2023年11月第22题：采用三级模式结构的数据库系统中，展现给用户的是()。

- A.外模式 B.模式 C.内模式 D.用户模式



真题

2023年11月第8题：某电影院某日电影入座情况如下表所示。为调整场次，要统计2021年2月21日到场人数总数大于100的电影，可满足要求的SQL语句是()

- A.SELECT film,sum(attendance) FROM movie WHERE pdate ="20210221" HAVING sum(attendance)>100
- B.SELECT film,sum(attendance) FROM movie WHERE pdatee=20210221" AND attendance >100 GROUP BY film
- C.SELECT film,sum(attenedance) FROM movie WHERE pdate="20210221" GROUP BY film HAVING sum(attendance)>100
- D.SELECT film,sum (attendance)FROM movie WHERE pdate=20210221 AND sum(attendance)>100 GROUP BY film

| pdate | ptime | film | attendance |
|----------|-------|--------|------------|
| 20210221 | 10:40 | 唐人街探案3 | 43 |
| 20210221 | 12:50 | 您好，李焕英 | 52 |
| 20210221 | 14:30 | 唐人街探案3 | 55 |
| 20210221 | 19:10 | 您好，李焕英 | 54 |
| 20210221 | 22:00 | 刺杀小说家 | 47 |
| 20210222 | 10:40 | 唐人街探案3 | 40 |



THE END

功不唐捐，玉汝于成！

• 开启新征程 •

