数据库概念题

- **试述文件系统与数据库系统的区别和联系**

区别：

1. 文件系统的数据一般来说仅面向某一个应用，文件的共享性差、冗余度大、独立性差，文件的记录虽然是有结构的，但整体无结构
2. 数据库系统中的数据不仅仅面向某一个应用，而可以面向整个组织或企业，数据共享性高、冗余度小，具有高度的物理独立性和逻辑独立性

联系：

1. 文件系统与数据库系统都是计算机系统中管理数据的软件
2. 文件系统是操作系统中重要组成部分，而数据库管理系统是独立于操作系统的软件
3. 数据库中数据的组织和存储是通过操作系统中文件系统来实现的

- 举出适合文件系统而不是数据库系统的应用例子，以及适合用数据库系统的应用例子

■数据库管理系统

- DataBase Management System
- 管理数据库的一种大型复杂软件系统
- 主要功能包括：
  - 数据定义功能
  - 数据组织、存储、管理功能
  - 数据操纵功能
  - 数据库事务管理和运行管理
  - 数据库的建立与维护...

数据库系统通常由**程序、数据库、数据库管理系统、用户**组成

■ 数据管理的任务

- 数据存储（Data Storage）
  合理组织数据，持久化存储
- 数据维护（Data Maintenance）
  维护数据内容，对数据集合进行增、删、改等操作
- 数据查询（Data Query）
  从数据存储中提取需要的数据
- 其它如数据安全等...

- **数据库用户**
  - **数据库管理员（DBA）**

负责安装数据库管理系统，通过数据库管理系统维护数据库，制定和执行安全策略、备份恢复策略和调优策略等，保障系统平稳高效运行

  - **应用系统开发人员**

设计和开发数据库应用系统，主要（在应用程序代码中）通过DML语言使用数据库中的数据

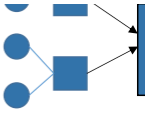
  - **终端用户**

使用数据库系统完成特定业务。主要通过应用程序人机界面访问操作数据库，不需要具备数据库知识。

■ 数据库系统特点

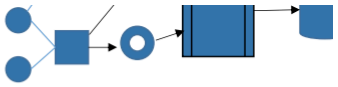
- 数据结构化
- 数据共享
- 数据独立性
- 方便的用户接口
- 统一的数据管理与控制功能

- 模式--逻辑层



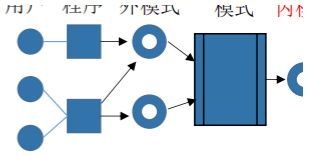
  - 模式（也称概念模式）
    - ✓数据库中全体数据的逻辑结构和特征的描述
    - ✓所有用户的公共数据视图，综合了所有用户的需求
  - 一个数据库只有一个模式
  - 模式的地位：是数据库系统模式结构的中间层
    - ✓与数据的物理存储细节和硬件环境无关
    - ✓与具体的应用程序、开发工具及高级程序设计语言无关
  - 模式的定义
    - ✓数据的逻辑结构（数据项的名字、类型、取值范围等）
    - ✓数据之间的联系
    - ✓数据有关的安全性、完整性要求等

- 外模式—用户层



  - 外模式（也称子模式或用户模式）
    - ✓ 数据库用户（包括应用程序员和最终用户）使用的局部数据的逻辑结构和特征的描述
    - ✓ 外模式是基于数据视图（View）实现。视图（View）是与某一应用有关的数据的逻辑表示
  - 外模式的地位：介于模式与应用之间
    - ✓ 模式与外模式的关系：一对多
    - ✓ 外模式通常是模式的子集
    - ✓ 一个数据库可以有多个外模式。反映了不同的用户的应用需求、看待数据的方式、对数据保密的要求
    - ✓ 对模式中同一数据，在外模式中的结构、类型、长度、保密级别等都可以不同
  - 外模式与应用的关系：一对多
    - ✓ 同一外模式也可以为某一用户的多个应用系统所使用，
    - ✓ 但一个应用程序只能使用一个外模式。
  - 外模式的用途
    - ✓ 保证数据库安全性和数据独立性的一个有力措施
    - ✓ 用户通过外模式中的数据

- 内模式—物理层



  - 内模式（也称存储模式）
    - ✓是数据物理结构和存储方式的描述
    - ✓是数据在数据库内部的表示方式
      - 记录的存储方式（顺序存储，按照B树结构存储，按hash方法存储等等）
      - 索引的组织方式
      - 数据是否压缩存储
      - 数据是否加密
      - 数据存储记录结构的规定
  - 一个数据库只有一个内模式

外模式/模式映像的作用：

✓作用：保证数据的逻辑独立性

当模式改变时，数据库管理员修改有关的外模式 / 模式映象，使外模式保持不变

应用程序是依据数据的外模式编写的，从而应用程序不必修改，保证了数据与程序的逻辑独立性。

模式/内模式映像的作用：

✓作用：保证数据的物理独立性

当数据库的存储结构改变了（例如选用了另一种存储介质），数据库管理员修改模式 / 内模式映象，使模式保持不变，应用程序不受影响。保证了数据与程序的物理独立性。

概念模型的特点

- 概念模型用于信息世界的建模，对现实世界进行第一层的抽象
- 独立于计算机系统
- 是数据库设计的有力工具
- 数据库设计人员和用户之间进行交流的语言

逻辑模型的特点

- 逻辑数据模型是对现实世界的第二层抽象。负责将概念数据模式映射为数据库的逻辑结构。
- 直接与DBMS有关,有严格的形式化定义,以便在计算机系统中实现。
- 它通常有一组严格定义的无二义性语法和语义的DB语言，人们可以用这种语言来定义、操纵DB中的数据。
- 逻辑数据模型的三要素：
  - **数据结构、数据操作和数据约束**

- 一些建模准则
  - ✓ 忠实于用户需求
  - ✓KISS准则 (keep it simple and stupid)
  - ✓避免冗余
  - ✓能抽象为属性的，就不要抽象为实体

- 约束是数据库模式的重要组成部分
  - ✓ 丰富了数据的语义
  - ✓ 保证数据的有效性和完整性
  - ✓ 实现更高效的存储和数据查询

✓ 能抽象为属性的，不要抽象为实体

✓实体 VS 属性

✓ 理论上所有数据对象都可以被看成实体

✓ 但是只有在符合下面两个情况之一的时候才需要把事物抽象为实体

1. 事物有至少一个非键属性（不属于键的属性）

   学生（姓名） VS 书（书名，页数，出版年代）

2. 事物处于“一对多”或“多对多”联系中“多”那一端

✓ 不符合以上情况的事物，都可以抽象为属性

■ER模型 vs. 关系模型

- 都用于数据建模
- ER模型有很多概念
  - ✓ 实体、关系、属性等
  - ✓ 适用于描述应用需求
  - ✓ 不适合在计算机上实现

    (只有数据结构，没有定义数据操作)
- 关系模型
  - ✓ 只有一个概念：关系（relation）
  - ✓ 用一组表的集合来描述世界
  - ✓ 适合在计算机上实现，能进行高效的数据操作

■ SQL的特点

- 综合统一
- 高度非过程化
- 面向集合的操作方式
- 以同一种语法结构提供两种使用方法
- 语言简洁，易学易用

■ 视图的作用

- 视图能够简化用户的操作
- 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
- 视图能够对机密数据提供安全保护

存储过程的优点

- ✓是SQL和模块化编程的结合，能够完成复杂业务功能
- ✓在创建的时候进行预编译，可以提高SQL执行效率
- ✓位于数据库服务器上，调用的时候无需通过网络传输大量数据
- ✓可以做为一种安全机制来加以充分利用。例如参数化的存储过程可以防止SQL注入式的攻击

- 使用自定义函数的优点类似存储过程：

1. 允许模块化程序设计：只需创建一次函数并将其存储在数据库中，以后便可以在程序中调用任意次。 用户定义函数可以独立于程序源代码进行修改。
2. 执行速度更快：MySQL 会缓存函数执行计划，重复调用时直接使用缓存计划，减少解析和优化开销，特别适合频繁调用的计算密集型操作。
3. 减少网络流量：基于某种无法用单一标量的表达式表示的复杂约束来过滤数据的操作，可以表示为函数。 然后，此函数便可以在 WHERE 子句中调用，以减少发送至客户端的数字或行数。

B+树索引

- 数据库系统中使用最广泛的多级索引
- 特点
  - ✓ 将索引键组织成一棵平衡树， 即从树根到树叶的所有路径一样长
  - ✓ 数据（指向基本表记录存储位置的指针）存储在叶结点
  - ✓ 最底层的叶节点包含每个索引键和指向被索引行的指针（行id）
  - ✓ 叶节点之间有通道可供平行查询
  - ✓ 每一个叶节点都和磁盘页面大小一致
  - ✓ 查询的时间复杂度：$O(\log_m n)$ (m为分叉数，即B+树的阶)

为什么不用B树索引

- B树将数据（行id）放到非叶节点中，导致每个节点能存放的索引项减少，树的层级更多，检索时需要更多磁盘IO
- B+树将所有数据在叶节点中有序存放并构成一个链表
  - 范围查询的效率更高
  - 缓存命中率更高 (空间局部性原理)
- B+树的更新维护代价更小

■ 散列索引特点

- 散列索引是CPU密集型的，B+树索引是I/O密集型的（I/O次数多于散列索引）
- 散列索引在进行等值查找时速度很快
- 散列索引无法用于范围查找
- 不适合在重复值很多的列上建立哈希索引
- 哈希索引重构代价很大，不适合在更新频繁的表中建立哈希索引

■ 聚簇索引特点

- 物理顺序只有一个，因此一张表只能有一个聚簇索引
- 在聚簇索引列上的查询速度比B+树索引快
- 数据在物理上按顺序排在数据页上，重复值也排在一起，因而在使用包含范围检查(between、<、<=、>、>=)或使用group by或order by的查询时，可以大大提高查询速度
- DML频繁的表中不要建立聚簇索引，因为会带来大量索引数据维护的开销
- MySQL在表的主键上建立聚簇索引

■ 什么是数据库的完整性

- 数据的正确性和相容性
- 防止不合语义的数据进入数据库。

触发器 vs 存储过程

- 但相较于存储过程，不能脱离宿主存在
- 由数据库自动调用执行，用户不能调用
- 没有参数和返回值

规范化的目的：

- 使结构合理
- 消除存储异常
- 使数据冗余尽量小
- 便于插入、删除和更新

- 非规范化设计的主要优点
  - 减少了查询操作所需的连接
  - 减少了外部键和索引的数量
  - 可以预先进行统计计算，提高了查询时的响应速度
- 非规范化存在的主要问题
  - 增加了数据冗余
  - 影响数据库的完整性
  - 降低了数据更新的速度
  - 增加了存储表所占用的物理空间