

一、选择题

1. 数据管理系统相关知识

3. 产生 E-R 图是在数据库设计的哪个阶段

4.

关于关系模型,下列叙述不正确的是()。

A. 一个关系至少要有有一个候选码

B. 列的次序可以任意交换

C. 行的次序可以任意交换

D. 一个列的值可以来自不同的域

5. 有两个关系 R(A,B,C) 和 S(B,C,D), R 有 100 个元组, S 有 300 个元组, 将 R 和 S 自然连接, 得到的结果包含几个列

A. 6

B. 4

C. 5

D. 2

6. 在上述关系中, 得到的结果包含的元组的数量

A. 30000

B. 100

C. 300

D. 无法确定

7. 关系代数 $\{t | t \in R \vee t \in S\}$

A. $R \cup S$

B. $R \cap S$

C. $R \times S$

D. $R \div S$

8. 下列表示关系 R 在 f 上的投影的是哪一个

所有视图都可以插入数据 (判断)

14. 封锁是为了什么

C. 并发控制

15. $N = 5, W = 2, R = 3$

A. 能保证读一致性, 不能保证写一致性

B. 能保证写一致性, 不能保证读一致性

C. 既能保证读一致性, 也能保证写一致性

D. 既不能保证写一致性, 也不能保证读一致性

二、简答题

1. 从数据共享性和持久性角度简述文件系统和数据库系统

2. 什么是概念模型, 有什么作用

3. 概述自主访问控制的原理和缺点

4. 完整性约束有哪几种, 以选课为背景分别举例说明

5. 聚簇的原理, 以教务系统为例分别从正/反两个方面说明其适合/不适合的场景

6. 概述检查点技术的原理, 说明使用检查点进行恢复的流程

7. 为什么要引入 NOSQL

8. 简述图数据库的基本原理、特点和适用场景。

三、综合题

1. ER 图

2. 关系数据库中有三个关系:

学生: S (Sno, Sname, Ssex, Sage, Sdept), 其中: Sno: 学号, Sname: 姓名, Ssex:

性别, Sage: 年龄, Sdept: 系名;

课程：C (Cno, Cname, Teacher)，其中：Cno：课程号，Cname：课程名，Teacher：教师；

选课：SC (Sno, Cno, Grade)，其中：Sno：学号，Cno：课程号，Grade：成绩。

用关系代数表达式实现下列 1、2 小题：

1. 检索年龄为 19 岁的女同学的学号和姓名；
2. 检索学生的学号、姓名、系名、课程名和成绩。

用 SQL 语言完成 3-5 小题：

3. 查询和“张三”同一个系的所有学生情况；
4. 检索学习了课程号“C001”的学生学号，姓名，与成绩，并按成绩递减排序；
5. 用 CREATE TABLE 语句定义基本表选课（需定义主键和外键，设 Sno 为字符型，长度为 6，Cno 为字符型，长度为 8，Grade 为整数）。

3.

考虑关系模式 $R(A, B, C, D)$ ，写出满足下列函数依赖时 R 的码，并给出 R 属于哪种范式 (1NF、2NF、3NF 或 BCNF)。

- ① $B \rightarrow D, AB \rightarrow C$;
- ② $A \rightarrow B, A \rightarrow C, D \rightarrow A$;
- ③ $BCD \rightarrow A, A \rightarrow C$;
- ④ $B \rightarrow C, B \rightarrow D, CD \rightarrow A$;
- ⑤ $ABD \rightarrow C$ 。

1. 从数据共享性和持久性角度简述文件系统和数据库系统

(1) 数据共享性

文件系统：

- 数据通常由各个应用程序各自管理
- 不同程序使用不同文件，**共享困难**
- 易产生**数据冗余、不一致**

数据库系统：

- 数据集中存储在数据库中
- 多个用户、多个应用可共享同一份数据
- 通过并发控制和权限管理保证共享的安全性

👉 **结论：**数据库系统的数据共享性显著优于文件系统。

(2) 数据持久性

文件系统：

- 数据持久存储在外存（磁盘）
- 缺乏系统级的恢复机制
- 一旦程序或系统异常，可能导致数据破坏

数据库系统：

- 数据同样持久存储在外存
- 提供**日志、恢复、检查点**等机制
- 能保证事务提交后的数据长期可靠保存

👉 **结论：**数据库系统在持久性和可靠性方面远强于文件系统。

2. 什么是概念模型？有什么作用

(1) 概念模型的定义

概念模型是对现实世界数据及其联系的抽象表示，
用于描述数据的语义结构，而不涉及具体的数据库实现。

常见形式：

- ER 模型（实体 - 联系模型）
-

(2) 作用

1. 抽象现实世界
 - 将现实中的对象抽象为实体、属性和联系
 2. 沟通桥梁
 - 便于数据库设计人员与用户之间交流需求
 3. 数据库设计基础
 - 是从需求分析到逻辑结构设计的重要过渡
-

3. 概述自主访问控制（DAC）的原理和缺点

(1) 原理

自主访问控制是由数据对象的所有者自主决定
其他用户是否以及如何访问该对象的安全机制。

- 用户通过 GRANT / REVOKE 授权或收回权限
 - 权限可被传递（WITH GRANT OPTION）
-

(2) 缺点

1. 权限传播不可控
 - 被授权用户可能再次授权他人
2. 安全性较弱
 - 易造成权限扩散
3. 难以抵御恶意攻击
 - 不能有效防止信息泄露

👉 因此，DAC 通常需要与 MAC（强制访问控制）结合使用。

4. 完整性约束有哪几种？以选课为背景举例

(1) 实体完整性

主键不能为空且唯一

例子：

- 学生表 S(S#, SNAME, ...)
 - 要求 S# 不允许为空、不可重复
-

(2) 参照完整性

外键必须引用已存在的主键值

例子：

- SC(S#, C#)
 - SC.S# 必须存在于 S 表
 - SC.C# 必须存在于 C 表
-

(3) 用户自定义完整性

用户根据实际业务需求定义的约束

例子:

- $GRADE \in [0, 100]$
 - 学生 $AGE \geq 15$
 - 一名学生同一门课只能选一次
-

5. 聚簇 (Clustering) 的原理及适用/不适用场景

(1) 聚簇原理

将具有相同或相近聚簇码值的元组

在物理存储上尽量放在一起。

目的:

- 减少磁盘 I/O
 - 提高查询效率
-

(2) 适合场景 (正面)

教务系统示例:

- 按“课程号 C#”对选课表 SC 聚簇
 - 查询“某门课程的所有学生”时
 - 数据物理上集中
 - 访问效率高
-

(3) 不适合场景 (反面)

- 频繁插入、删除
- 聚簇码变化频繁
- 多种查询方式并重 (无法同时满足)

例子:

- 同时高频按 S#、C# 查询 SC 表
 - 只能选一个聚簇码, 另一种查询效率下降
-

6. 概述检查点技术的原理及恢复流程

(1) 原理

检查点是系统在某一时刻将数据库的状态和日志信息同步保存到稳定存储中的机制。

作用:

- 缩短恢复时间
 - 减少日志扫描量
-

(2) 恢复流程

1. 系统故障后读取最近一次检查点
 2. 从检查点开始扫描日志
 3. **重做 (REDO)** 已提交事务
 4. **撤销 (UNDO)** 未提交事务
 5. 数据库恢复到一致状态
-

7. 为什么要引入 NoSQL

原因总结：

1. 关系数据库扩展性不足
 - 难以应对海量数据和高并发
2. 数据模型灵活性不足
 - 半结构化 / 非结构化数据处理困难
3. 分布式支持复杂
 - 跨节点事务代价高

👉 NoSQL 提供：

- 高扩展性
 - 灵活数据模型
 - 高可用性
-

8. 图数据库的基本原理、特点和适用场景

(1) 基本原理

以“节点（Node）和边（Edge）”的形式直接存储实体及其关系。

- 关系是一等公民
 - 避免多表 JOIN
-

(2) 特点

1. 关系表达直观
 2. 查询复杂关系效率高
 3. 适合多跳关系遍历
-

(3) 适用场景

- 社交网络
- 推荐系统
- 知识图谱
- 反欺诈分析