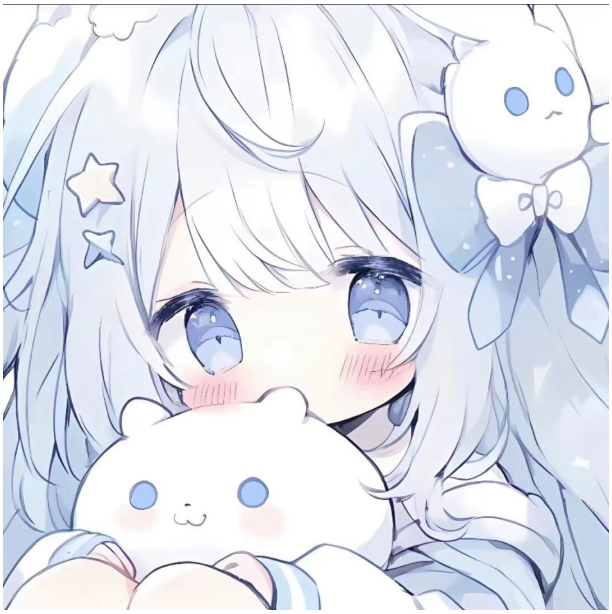
数据库系统原理课程

往年真题

烂石

2025 年 3 月 15 日



1 选择题 (40 points)

1.1 日志文件的作用

日志文件是数据库系统中至关重要的组件，主要用于保障事务的 ACID 特性（原子性、一致性、隔离性、持久性），确保数据的一致性和系统故障后的恢复能力。

1.2 为什么要应用数据库

- 共享与高效利用
- 减少冗余与提高一致性
- 支持并发与安全性
- 数据独立性与事务处理

1.3 DB/DBS/DBMS 之间关系

1.3.1 层级关系

DBS（系统）> DBMS（软件）> DB（数据）
DB 是数据存储，DBMS 是管理 DB 的软件，DBS 是整体系统。

1.3.2 核心功能区分

DBMS：提供数据管理功能（如事务、安全、查询优化）。DBS：包含 DBMS、DB、应用程序及用户，是完整的数据管理解决方案。

1.3.3 常见误区

混淆 DB 和 DBMS（如认为“数据库就是 MySQL”）。认为 DBS 仅包含 DB 和 DBMS，忽略应用程序和用户。

2 填空题 30 points

2.1 逻辑独立性是什么?

逻辑独立性是指当数据的逻辑结构（概念模式）改变时，应用程序不需要修改。

2.2 把一个 sql 语句转化成关系代数

2.2.1 填空标准答案示例：

将 SQL 语句 'SELECT A, B FROM R WHERE C > 5' 转换为关系代数表达式：
π_{A,B}(σ_{C>5}(R))
（其中：- π 表示投影（对应 SELECT 字段），
- σ 表示选择（对应 WHERE 条件），
- R 是关系名。）

2.2.2 常见填空题考点：

1. SQL 的 'SELECT'对应关系代数的投影（π）。
2. SQL 的 'WHERE'对应关系代数的选择（σ）。
3. SQL 的 'JOIN'对应关系代数的连接（⋈）。
4. SQL 的 'GROUP BY'对应关系代数的分组（Γ）。

2.3 关系范式

2.3.1 关系范式填空题模板

1. 关系模型应满足一定的规范要求，称为关系范式（或规范化），其中最基本的三个范式是第一范式（1NF）、第二范式（2NF）、第三范式（3NF）。

2. 第一范式（1NF）要求关系模式中的每个属性的值域中不可再分，即每个属性都是原子值。

3. 第二范式（2NF）要求关系模式必须满足1NF，且非主属性完全函数依赖于候选键（或主键）。

4. 第三范式（3NF）要求关系模式必须满足2NF，且非主属性对候选键不存在传递函数依赖。

5. 巴斯-科德范式（BCNF）要求关系模式中的每一个函数依赖 X→Y，决定因素 X 必须包含候选键（或超键）。

6. 对于关系模式 R，若 R 3NF 且消除了非平凡的多值依赖，则 R 属于第四范式（4NF）。

7. 第五范式（5NF）又称为投影-连接范式（Projection-Join Normal Form），要求关系模式中不存在连接依赖。

表 1: 数据库范式核心条件

范式	核心条件
1NF	每个属性的值域不可再分（原子性）。
2NF	满足 1NF，且非主属性完全依赖于候选键。
3NF	满足 2NF，且非主属性不传递依赖于候选键。
BCNF	满足 3NF，且所有函数依赖的决定因素都是候选键。
4NF	满足 BCNF，且不存在非平凡的多值依赖。
5NF（PJNF/BC 范式）	满足 4NF，且不存在非平凡的连接依赖。

2.3.2 关键概念总结

2.3.3 常见考点

范式之间的关系:

每个更高范式都是前一个范式的超集（如：BCNF 3NF 2NF 1NF）。

典型反例：

违反 2NF：如学生（学号，姓名，课程 1，成绩 1，课程 2，成绩 2），非主属性部分依赖主键。

违反 3NF：如部门（部门号，部门名，经理，经理电话），经理电话传递依赖于部门号。

违反 BCNF：如员工（项目，员工，技能），若项目 → 员工且员工 → 技能，则决定因素不包含候选键。

规范化的作用：

消除数据冗余、插入异常、删除异常和更新异常。

2.4 插入异常、修改异常（更新异常）和删除异常

2.4.1 填空题模板

1. 插入异常是指_____，通常由于关系模式中存在_____或_____ 导致。

   答案：无法插入某些有意义的数据行；部分函数依赖；传递函数依赖

2. 修改异常（更新异常）是指_____，例如修改某个属性值时需要_____，容易导致数据不一致。

   答案：修改数据时需要修改多处冗余数据；修改多个元组

3. 删除异常是指_____,例如删除某个元组时可能同时丢失_____

答案：删除元组时导致有用信息丢失；其他相关数据

4. 关系模式未规范化时，容易出现_____、_____ 和_____，这些是关系规范化要解决的核心问题。

答案：插入异常；修改（更新）异常；删除异常

5. 若关系模式中存在非主属性对候选键的传递依赖，则可能导致_____ 和_____。

答案：插入异常；删除异常

6. 规范化到_____ 可以消除非主属性对候选键的传递依赖，从而解决部分异常。

答案：第三范式（3NF）

7. 在关系模式中，如果存在多值依赖，则可能引发_____，需通过规范化到_____ 解决。

答案：插入异常；第四范式（4NF）

2.4.2 关键概念总结

表 2: 数据库规范化异常类型表

异常类型	定义	示例
插入异常	无法插入某些有意义的数据行，因主键或外键约束导致	无法插入”课程”表中尚未存在的课程信息，因需要关联学生表的主键
修改异常	修改数据时需要修改多处冗余数据，导致不一致或操作繁琐	修改”部门名称”需遍历所有员工记录，容易遗漏
删除异常	删除元组时导致有用信息丢失（如删除学生记录时同时丢失其选课信息）	删除一个学生记录时，该学生的选课记录也被删除，但选课信息可能需要保留

2.4.3 常见考点解析

- 异常产生的原因：
  - 插入异常：非主属性对主键存在部分依赖或传递依赖
  - 修改异常：冗余数据（如重复存储的部门名称）需要同时修改多处

  - 删除异常：主键关联的数据被级联删除，导致非主属性信息丢失

- 规范化的作用：
  - 1NF 到 3NF：解决插入、修改、删除异常（通过消除非主属性的传递依赖）
  - BCNF：进一步消除决定因素非键的函数依赖
  - 4NF/5NF：解决多值依赖和连接依赖导致的异常

- 典型反例：
  - 未规范化表：学生（学号，姓名，课程 1，成绩 1，课程 2，成绩 2）
    - 插入异常：若学生未选满 2 门课程，无法插入
    - 删除异常：删除学生记录时，所有课程信息被删除
  - 规范化后：拆分为学生（学号，姓名）和选课（学号，课程，成绩），消除异常

2.5 专门的关系运算，选择、投影、连接、（除）

2.5.1 填空题模板

1. 选择运算（Selection）的符号是_____，其作用是_____。

   答案：σ（西格玛）；从关系中筛选满足条件的元组（行）。

2. 投影运算（Projection）的符号是_____，其作用是_____。

   答案：π（派）；从关系中选择特定的属性（列），并消除重复元组。

3. 连接运算（Join）分为_____和_____，其中自然连接要求_____

   答案：θ 连接（如等值连接、不等连接）；自然连接；连接条件为公共属性的等值，并且自动去除重复列。

4. θ 连接的符号是_____，其一般形式为 R_____S。

   答案：∞（无穷符号）；$\sigma_{<\text{条件}>}(R \times S)$，但通常直接写为 $R \bowtie_{<\text{条件}>} S$。

5. 除运算（Division）的符号是_____，其作用是_____。

   答案：$\div$；用于查询满足“对于某个属性集的所有值，都存在对应的元组”的条件。

6. 自然连接要求两个关系有_____,而 θ 连接可以基于_____进行连接。

   答案：公共属性；任意属性间的条件（如等值、不等值）。

7. 投影运算的结果_____（是/否）自动去重，因为关系模型中元组是_____ 的。

   答案：是；无序且唯一。

8. 关系 $R \div S$ 的结果是_____，即 R 中满足_____的元组。

   答案：R 中在除数 S 的属性集上的所有组合均在 R 中存在对应元组；对于 S 的所有元组，R 中存在元组与之匹配。

2.5.2 关键概念总结

表 3: 关系代数运算对照表

运算	符号	作用	示例
选择	σ	筛选满足条件的元组（行）	$\sigma_{\text{年龄}>20}(R)$：选择年龄大于 20 的元组
投影	π	提取指定属性（列），并自动去重	$\pi_{\text{学号, 姓名}}$(学生)：提取学号和姓名列
连接	$\bowtie$	合并两个关系的元组，基于条件（如等值或不等值）	学生 $\bowtie$ 选课（学号相等）合并学生和选课表
除法	$\div$	查询满足“对于 S 的所有元组，R 中存在匹配”的元组	$R \div S$：如查询选修了所有课程的学生

2.5.3 常见考点解析

- 运算符号与作用：
  - 选择（σ）：行级过滤（`WHERE` 子句的数学基础）
  - 投影（π）：列级筛选（类似 SQL 的 `SELECT` 列）
  - 连接（$\bowtie$）：表合并操作（对应 SQL 的 `JOIN`）
  - 除法（$\div$）：复杂查询（如“所有 X 都满足 Y”）
- 运算特点：

  - 投影自动去重：关系模型要求元组唯一
  - 自然连接：自动去除重复列（如两个表的公共属性只保留一列）
  - 除法条件：$R \div S$ 的结果是 R 中在 S 的属性集上的投影

- 典型反例：
  - 除法应用示例：
    * R（学号，课程）和 S（课程）
    * $R \div S$ 的结果是“选修了 S 中所有课程的学生学号”
    * 若 $S = \{C1, C2\}$，则结果需同时包含 C1 和 C2 的学号

3 简答题 30points

3.1 数据挖掘步骤

数据挖掘的典型步骤包括：
业务理解（明确目标）；
数据理解（探索数据）；
数据准备（清洗、转换数据）；
建立模型（选择算法建模）；
模型评估（验证性能）；
部署应用（实际落地）。

3.2 文件系统阶段的缺陷

文件系统阶段的主要缺陷包括：
数据冗余与不一致性（更新困难）；
数据共享性差（文件格式不统一）；
数据独立性低（程序与数据强耦合）；
安全性不足（权限控制粗放）；
缺乏完整性控制；
并发控制问题；
查询效率低下；
缺乏统一管理。

3.3 视图与基本表的联系与区别

1. 联系：

(a) 视图基于基本表定义，依赖其数据；

(b) 修改基本表会影响视图结果，视图更新（若允许）会反向修改基本表。

2. 区别：

(a) 存储方式：

- 基本表：存储真实数据；
- 视图：仅存储查询定义。

(b) 修改权限：

- 基本表：支持直接增删改查；
- 视图：默认仅允许查询，更新受限制。

(c) 安全控制：

- 视图可隐藏敏感字段，实现数据访问权限控制。

(d) 独立性：

- 视图隔离底层表结构变化，提升应用逻辑独立性。

3.4 系统故障的恢复策略

系统故障的恢复策略包括：
日志记录：记录事务操作，支持回滚和重做；
检查点机制：定期将内存数据写入磁盘，减少恢复范围；
UNDO（回滚）：撤销未提交事务的修改；
REDO（重做）：重做已提交但未持久化的事务；
预写日志（WAL）：确保数据与日志的顺序一致性。

3.5 调度封锁

3.6 数据库管理系统存取数据的流程

数据库管理系统存取数据的流程包括：

1. 用户请求与解析（语法、语义检查）；
2. 查询优化（选择访问路径）；
3. 执行引擎（执行计划）；
4. 事务管理（并发控制、日志记录）；
5. 存储引擎（访问物理数据）；

6. 缓冲区与持久化（脏页刷新、检查点）；

7. 结果返回。

3.7 事务隔离级别的区别

3.7.1 核心区别总结

表 4: 隔离级别之间的核心区别

隔离级别	脏读	不可重复读	幻读	并发性能	典型实现
读未提交	✓	✓	✓	高	无锁机制
读已提交	×	✓	✓	较高	锁或 MVCC（部分场景）
可重复读	×	×	✓	中等	MVCC（如 MySQL）
串行化	×	×	×	低	排他锁

关键问题的定义

1. 脏读（**Dirty Read**）：读取到其他事务未提交的数据，若该事务回滚，则数据无效。

2. 不可重复读（**Non-Repeatable Read**）：同一事务内多次读取同一数据，结果不同（因其他事务已提交更新）。

3. 幻读（**Phantom Read**）：同一事务内多次查询同一条件的数据，结果集行数或内容变化（因其他事务插入了符合条件的新数据）。

3.8 事务的 AICD

3.8.1 ACID 特性总结

3.8.2 关键问题

1. 为什么需要 **ACID**？

   - 保证事务的可靠性，确保数据在并发、故障等复杂环境下仍能正确处理。

2. **ACID** 如何实现？

   - 通过日志（保证原子性和持久性）、锁/MVCC（保证隔离性）、约束（保证一致性）等机制。

表 5: ACID 特性总结

特性	核心作用	典型场景
原子性	避免“半完成”操作	银行转账、订单创建
一致性	保证数据合法性和业务规则	数据库约束（如主键、外键）
隔离性	避免并发问题（如脏读、幻读）	多用户同时操作同一数据
持久性	数据提交后永不丢失	系统崩溃后恢复数据

3. **ACID** 与 **BASE**（**NoSQL** 的软状态模型）的区别？

    - ACID 强调强一致性，BASE 强调最终一致性（如高并发场景下牺牲强一致性换取可用性）。

3.8.3 备考建议

- 记忆技巧：用“原子、一致、隔离、持久”四字口诀记忆 ACID。
- 结合实例：用银行转账、订单系统等经典案例理解每个特性的作用。
- 关联隔离级别：隔离性（Isolation）与事务的四个隔离级别（如 Read Committed）直接相关，需结合理解。

3.9 完整性约束

3.9.1 约束类型与其核心作用

表 6: 数据库约束类型总结

约束类型	核心作用	典型场景
实体完整性	确保主键唯一且非空	学生表的学号、订单表的订单号
域完整性	限制字段的取值范围和类型	年龄限制、成绩范围
参照完整性	保证外键引用有效	订单与客户表关联、部门与员工关联
用户定义完整性	实现业务规则（如折扣率限制）	价格区间、性别选项
唯一性约束	确保字段值唯一（允许空值）	邮箱、身份证号
默认值约束	自动填充字段的默认值	创建时间、状态标志

3.9.2 备考建议

1. 记忆关键点：用“实体、域、参照、用户定义”四类约束框架记忆。
2. 结合 **SQL** 语法：掌握 `PRIMARY KEY`、`FOREIGN KEY`、`CHECK`、`UNIQUE` 等约束的 SQL 实现。
3. 案例分析：通过实际表设计（如学生选课系统）理解约束的应用场景。
4. 常见考点：
   - 不同约束的区别（如主键与唯一性约束）。
   - 违反约束的后果及处理方式。
   - 约束对数据一致性的保障作用。

3.10 数据库的优化

3.10.1 优化原则

表 7: 数据库优化原则

优化原则	具体说明
先分析后优化	通过慢查询日志、性能监控工具（如 ‘SHOW PROCESSLIST‘、‘pt-query-digest‘）定位瓶颈。
避免过度优化	优先解决核心问题，避免为小问题引入复杂方案。
平衡读写	根据业务场景权衡一致性与性能（如 OLTP vs OLAP）。

3.10.2 简答题答题要点

1. 核心方法：
   - 索引优化
   - 查询优化
   - 存储优化
   - 并发控制
2. 关键工具：

- ‘EXPLAIN‘
- 慢查询日志
- 执行计划分析

3. 典型场景：

- 分页优化
- 减少全表扫描
- 合理分区
- 避免死锁

3.11 关系模型和网状模型的数据结构

3.11.1 核心区别对比

表 8: 关系模型与网状模型的核心区别

特征	关系模型	网状模型
数据结构	二维表格（关系）	有向图（记录 + 指针）
关联方式	主键/外键（显式声明）	指针（隐式地址引用）
查询语言	SQL（声明式）	网状语言（过程式）
标准化	高（支持范式）	低（依赖指针，冗余可能高）
灵活性	查询灵活（通过 SQL）	查询复杂（需指针导航）
维护成本	低（结构化约束）	高（依赖指针路径）
适用场景	结构化、事务性数据	复杂多对多关系、资源受限环境

3.11.2 关键问题总结

1. 关系模型的优势：

- 简单直观，支持标准 SQL，易于维护。
- 符合 ACID，适合高并发事务场景。

2. 网状模型的局限性：

- 结构复杂，查询需依赖指针路径。
- 扩展性差，修改结构需调整大量指针。

3. 历史地位：

- 网状模型是关系模型的前身，已被关系模型取代（如 IBM 的 IMS 系统逐渐过渡到 DB2）。

3.11.3 考研备考建议

1. 重点记忆：
   - 二维表 vs 网状图
   - 主键/外键 vs 指针
   - SQL vs 过程式查询
2. 对比分析：
   - 通过表格对比结构、关联方式、查询语言、适用场景等。
3. 案例理解：
   - 关系模型：学生-课程表通过学号关联。
   - 网状模型：部门记录通过指针直接指向多个员工记录。

4 大题 50points

4.1 SQL 语句 30 points/8

考点包括：创建视图、修改表结构、插入数据、删除数据、group by 语句

4.2 E-R 图设计 12 points

4.3 函数依赖-范式 8 points