数据库概论（90113201）

第 1 次课后作业参考答案

一、填空题（每题 1 分，共 70 分）

1. **答案**：载体
2. **答案**：数据共享
3. **答案**：数据共享
4. **答案**：改善系统性能，提高系统效率
5. **答案**：硬件平台
6. **答案**：数据约束
7. **答案**：谓词模型
8. **答案**：实体
9. **答案**：二维表
10. **答案**：功能扩展
11. **答案**：数据库系统管理
12. **答案**：移植
13. **答案**：物理性少
14. **答案**：数据的统一管理与控制
15. **答案**：逻辑独立性
16. **答案**：内模式

17. **答案**：物理
18. **答案**：计算机世界
19. **答案**：网络
20. **答案**：ISO/IEC 9075
21. **答案**：属性
22. **答案**：至少存在一个
23. **答案**：笛卡儿乘积
24. **答案**：m（m 为指定投影的属性个数）
25. **答案**：嵌入式
26. **答案**：CREATE TABLE
27. **答案**：*
28. **答案**：MIN
29. **答案**：DROP TABLE
30. **答案**：映像语句
31. **答案**：安全数据库（或可信数据库）
32. **答案**：自主访问控制与授权功能
33. **答案**：USAGE 权
34. **答案**：并发控制技术
35. **答案**：用户定义的完整性规则
36. **答案**：不能
37. **答案**：主键
38. **答案**：全局约束（断言 assertion）

39. **答案**：触发过程
40. **答　案**：CREATE ASSERTION <name> CHECK(<condition>)
41. **答案**：持久性
42. **答案**：事务管理子系统
43. **答案**：编写事务的应用程序员
44. **答案**：可串行化
45. **答案**：恢复管理子系统
46. **答案**：异常中止
47. **答案**：提交事务（commit transaction）
48. **答案**：不可重复读
49. **答案**：共享锁
50. **答案**：丢失修改
51. **答案**：主存储器（memory）
52. **答案**：顺序文件
53. **答案**：记录指针（即记录在数据文件中的存储地址）
54. **答案**：稠密索引
55. **答案**：稀疏索引
56. **答案**：稀疏
57. **答案**：B+
58. **答案**：映射
59. **答案**：顺序

60. **答案**：日志信息、用户信息、审计信息等
61. **答案**：用户
62. **答案**：信息
63. **答案**：同一个
64. **答案**：WITH CHECK OPTION
65. **答案**：可以
66. **答案**：并发
67. **答案**：释放所有申请获得的锁
68. **答案**：事务
69. **答案**：海量
70. **答案**：自动光盘机

二、简答题（每题 3 分，共 30 分）

1. **答案**：数据库系统包含数据库、数据库管理系统、数据库管理员、软件平台和硬件平台等。数据库管理系统是管理数据库的系统软件，负责数据的组织、操纵、维护等，是数据库系统的核心组成部分。数据库管理员负责对数据库进行规划、设计、建立、调整、维护和监视等工作，通过使用数据库管理系统来实现对数据库的管理，以保障数据库系统的正常运行。
2. **答案**：概念数据模型侧重于对客观世界中复杂事物的结构描述及它们之间的内在联系的刻划，不涉及具体的描

述细节和物理实现因素，是一种面向客观世界和用户的模型，与具体采用的 DBMS 及计算机实现无关。逻辑数据模型着重于数据模型在数据库系统一级的实现，利用具体的 DBMS 所提供的工具（DDL）来定义，是一种面向数据库系统的模型，是概念数据模型转换为物理数据模型的中介。物理数据模型给出了数据模型在计算机内部的真正物理结构，是一种面向计算机物理实现的模型，一个概念数据模型需先转化为逻辑数据模型，再通过所选择的 DBMS 进一步转化为物理数据模型才能在计算机中得以物理实现。

3. **答案**：关系数据库系统的优点包括：

    1. 数据结构简单：以二维表为基本结构，数据组织形式直观清晰。
    2. 使用方便：不涉及系统内部物理结构，采用非过程性数据子语言，用户只需提出操作要求，无需关注具体实现过程。
    3. 功能强：模型的表达能力强，可方便地修改数据间的联系，灵活选择数据存取路径，还具有高级的数据操纵语言。

4. 数据独立性高：包括物理独立性和逻辑独立性，数据的物理结构或逻辑结构改变时，应用程序无需修改。
5. 理论基础深：有着坚实的数学理论基础。
6. 可移植性好：便于在不同的计算机系统上使用。
7. 标准化程度高：SQL 语言的标准化使得不同数据库系统间的交互和操作更规范。
8. 分布式功能：支持分布式数据库，可实现数据在不同节点的分布存储和处理。
9. 开放性：提供各种网络环境下的数据访问接口。
10. 功能扩展：能满足不同应用场景的多样化需求。

以数据结构简单为例，在学生成绩管理系统中，学生的信息（学号、姓名、成绩等）可以很方便地用二维表来表示。每个学生的信息作为一行（元组），每个属性（学号、姓名、成绩）作为一列，这种结构简单直观，无论是查询、插入还是修改数据都易于理解和操作。

4. **答案**：```sql

```
CREATE TABLE Student (
sno CHAR(5) NOT NULL,
sname VARCHAR(20),
sage INT,
```

```
sdept VARCHAR(20)
);
```

plaintext

– `CREATE TABLE`：创建表的关键字。

– `Student`：要创建的表的名称。

– `sno CHAR(5) NOT NULL`：定义`学号`字段，数据类型为长度为 5 的定长字符型，`NOT NULL`表示该字段不能为空值，确保每个学生都有学号。

– `sname VARCHAR(20)`：定义`姓名`字段，数据类型为长度最大为 20 的变长字符型，用于存储学生姓名。

– `sage INT`：定义`年龄`字段，数据类型为整型，用来记录学生的年龄。

– `sdept VARCHAR(20)`：定义`系别`字段，数据类型为长度最大为 20 的变长字符型，用于表示学生所属的系别。

5. **答案**：实体完整性规则：在一个基表的主关键字中，其属性的取值不能为空值。例如在学生表中，假设学号是主关键字，每个学生记录的学号都必须有值，不能为 NULL，否则无法唯一确定每个学生的记录，保证了表中记录的唯一性和完整性。

参照完整性规则：外键要么取空值，要么是被引用表中当前存在的某元组上的主键值。例如在选课表中，学号作为外键，它的值必须是学生表中已经存在的学号，或者为空（表示尚未确定学生）。如果选课表中出现一个在学生表中不存在的学号，就会破坏数据的一致性，而参照完整性规则确保了这种情况不会发生，维护了不同表之间数据的关联性。

6. 答案：

原子性：事务中的所有操作要么全部执行成功，要么全部不执行，就像一个不可分割的整体。在银行转账例子中，从账户 A 向账户 B 转账，包括从账户 A 扣款和向账户 B 加款这两个操作，这两个操作必须同时完成或者同时不完成，不能出现只完成其中一个操作的情况，保证了数据的一致性和完整性。

一致性：事务执行前后，数据库的状态保持一致，所有数据必须满足数据库中显式定义的各种完整性约束和用户心目中的隐式数据约束。比如银行转账前后，两个账户的总金额应该保持不变，确保了数据的正确性和合理性。

隔离性：一个事务的执行与其他并发执行的事务相互隔离，互不干扰。在多个转账事务同时进行时，每个事务都感觉不到其他事务的存在，就好像自己是唯一在执行的事务一样，

避免了并发事务之间的相互影响，保证了每个事务执行的独立性和正确性。

持久性：事务一旦完成全部操作，它对数据库的所有更新应永久地反映在数据库中，即使系统发生故障也不会丢失。例如银行转账成功后，无论系统之后发生什么故障，转账的结果都应该被保留，保证了数据的永久性和可靠性。

7. 答案：

一级封锁协议：事务 T 在“写”数据对象 A 之前，必须先申请并获得 A 上的“X 锁”，并维持到事务 T 的执行结束（包括 Commit 与 Rollback）才释放被加在 A 上的“X 锁”。它能防止“丢失修改”现象。例如在飞机售票系统中，两个事务同时对剩余机票数量进行修改，若没有一级封锁协议，可能会出现一个事务的修改被另一个事务覆盖的情况；而在一级封锁协议下，当一个事务获取“X 锁”对机票数量进行修改时，其他事务无法同时修改，避免了丢失修改问题。

二级封锁协议：在一级封锁协议的基础上，事务 T 在“读”数据对象 A 之前，必须先申请并获得 A 上的“S 锁”，在“读”操作完成后即可释放 A 上的“S 锁”。它能防止“丢失修改”和“脏读”现象。比如在一个库存管理系统中，事务 A 读取某商品库存数量后，事务 B 对该库存数量进行修改并提交，若没

有二级封锁协议，事务 A 可能读到事务 B 未提交的修改数据（脏读）；而二级封锁协议通过在读取前加“S 锁”，保证在读取过程中数据不会被其他事务修改，避免了脏读的发生。

三级封锁协议：在一级封锁协议的基础上，事务 T 在“读”数据对象 A 之前，必须先申请并获得 A 上的“S 锁”，并维持到事务 T 的执行结束（包括 Commit 与 Rollback）才释放被加在 A 上的“S 锁”。它能防止“丢失修改”“脏读”和“不可重复读”现象。例如在银行账户余额查询场景中，事务 A 在读取账户余额时加“S 锁”并保持到事务结束，这样在事务 A 执行过程中，其他事务无法修改该账户余额，避免了在同一事务中多次读取同一数据出现不一致的情况（不可重复读）。

8. 答案：数据库故障分类及恢复策略如下：

– 小型故障：即事务内部故障，故障影响范围在一个事务之内，不影响整个系统的正常运行。恢复策略是利用未结束事务的 undo 操作进行恢复，撤销事务对数据库已做的修改，使数据库回到事务执行前的状态。

– 中型故障：即系统故障，可导致整个系统停止工作，但磁盘数据不受影响。在系统重启时，需要恢复两类事务：非正常中止的事务，执行 undo 操作，撤销其未完成的修改；已完成提交但更新操作的修改结果还留在内存缓冲区中，尚

未写入磁盘的事务，执行 redo 操作，将这些事务的修改结果写入磁盘，以保证数据的一致性。

– 大型故障：如磁盘故障、计算机病毒、黑客入侵等，可导致内存及磁盘数据的严重破坏。恢复策略是先利用后备副本进行数据库恢复，将后备副本中的数据拷贝到数据库中；然后检查日志文件，确定哪些事务已经执行结束，哪些尚未结束；接着按照日志的记载顺序，逆向对尚未结束的事务作撤消处理（undo），正向对已经结束的事务作重做处理（redo），使数据库恢复到故障前的正确状态。

9. 答案：

稠密索引：数据文件中的每条记录在索引文件中都有对应的索引项，索引项包含记录的关键字值和指向记录本身的指针，并按关键字值顺序排序。例如在一个员工信息表中，若以员工编号为索引关键字建立稠密索引，每个员工的编号在索引文件中都有对应的索引项。当需要查找某个员工的信息时，可直接通过索引文件快速定位到该员工的记录，适用于频繁精确查找某条记录的场景。

稀疏索引：如果数据文件是顺序文件，只为数据文件的每个磁盘块设一个索引项，记录该磁盘块中第一条数据记录的关键字值及该磁盘块的首地址。比如在一个存储大量订单信息的顺序文件中，每个磁盘块存储若干订单记录，稀疏索引为

每个磁盘块建立一个索引项。当查找某订单记录时，先通过稀疏索引找到对应的磁盘块，再在磁盘块内查找具体订单，适用于数据量较大且查找频率相对较低的场景，可减少索引文件的大小，降低存储开销。

多级索引：在索引文件上再建立索引构成多级索引结构。直接建立在数据文件上的索引为第一级索引，根据第一级索引文件建立的索引为第二级索引，依此类推，从第二级索引开始都是稀疏索引。例如在一个超大型的图书数据库中，数据量巨大。第一级索引可以是稠密索引，对每本图书的关键信息建立索引项；随着数据量增加，为更快查找索引项，建立第二级、第三级等稀疏索引。查找时，先通过高级索引快速定位到低级索引块，再逐步找到具体记录，大大提高了查找效率，适用于超大型数据库的快速查找场景。

10. 答案：

设计简单图书馆管理系统数据库的思路如下：

– 依据数据库系统的集成性特点，将图书馆的各类数据，如书籍信息、读者信息、借阅信息等集成管理。采用统一的数据结构和全局数据模式，例如创建书籍表、读者表和借阅表。

– 利用关系数据模型，以二维表形式组织数据。书籍表可包含书号、书名、作者、出版社、出版年份、库存数量等

属性；读者表包含读者 ID、姓名、联系方式、借阅状态等属性；借阅表包含借阅 ID、读者 ID、书号、借阅时间、应还时间、实际还书时间等属性。通过外键关联不同表，如借阅表中的读者 ID 关联读者表，书号关联书籍表，以体现数据间的联系。

- 使用 SQL 语言进行数据库设计。通过`CREATE TABLE`语句创建上述表格，如：

```sql
CREATE TABLE Books (
    book_id CHAR(10) NOT NULL PRIMARY KEY,
    title VARCHAR(100),
    author VARCHAR(50),
    publisher VARCHAR(50),
    publish_year INT,
    stock INT
);

CREATE TABLE Readers (
    reader_id CHAR(10) NOT NULL PRIMARY KEY,
    name VARCHAR(50),
    contact VARCHAR(50),
    borrowing_status VARCHAR(20)
```

```
);

CREATE TABLE BorrowRecords (
    borrow_id CHAR(10) NOT NULL PRIMARY KEY,
    reader_id CHAR(10),
    book_id CHAR(10),
    borrow_time DATE,
    due_time DATE,
    return_time DATE,
    FOREIGN KEY (reader_id) REFERENCES
Readers(reader_id),
    FOREIGN KEY (book_id) REFERENCES
Books(book_id)
);
```

还可使用 SQL 的查询、插入、更新和删除语句来实现图书馆管理系统的各种功能。例如，通过查询语句实现书籍查询、读者借阅信息查询；通过插入语句添加新的书籍记录、读者记录和借阅记录；通过更新语句修改书籍库存数量、读者信息和借阅状态；通过删除语句删除过期或无效的记录等。