闭卷考试题型：  
  选择、填空、判断（2种）  
  简答题（\*\*\*是什么，会考并发控制和恢复）  
  关系操作：关系代数、SQL语言  
  设计题：范式（ER图）和数据库设计  
发展历史不考  
**第1章 绪论**数据库概念与特点（考的概率低）  
数据库系统构成（硬件、软件、数据平台、管理系统、人员）  
数据库系统三级模式结构：

外模式（子模式，用户模式）：某一数据库用户看见和使用的局部数据的逻辑结构和特征的描述，一个外模式是模式的一个子集，用户视图。

模式（逻辑模式、全局模式,唯一）：是数据库中全体数据的逻辑结构和特征的描述，是所有用户的公共数据视图，全局视图。

内模式（物理模式，存储模式，唯一）：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

二极映象与数据独立性：

外模式/模式映像（不唯一）：描述某一外模式与模式之间的对应关系，该映像定义包含在外模式的定义中。

数据的逻辑独立性：模式改变时，修改映像，保持外模式不变，应用程序是根据数据的外模式编写的，进而保持应用程序不变。

模式/内模式映像（唯一）：描述数据库模式与内模式之间的对应关系，该映像定义包含在模式定义中。

数据的物理独立性：内模式改变时，修改该映像，保持模式不变，进而保持应用程序不变。  
**第2章 关系数据库**  
关系数据结构的形式化定义:  
R（U,D,DOM,F）关系名、组成关系的属性名集合、U中属性所来自的域，属性向域的映像集合，属性间数据的依赖关系集合  
关系的完整性约束

实体完整性：若属性A是基本关系R的主属性，则A不能取空值

参照完整性：若属性或属性组F是基本关系R的外码，它与基本关系S的主码相对应（关系R与S不一定是不同的关系），则对于R的每个元组在F上的值必须为空值或者等于S中的某个元组的主码值

用户定义完整性：反映具体应用所涉及的数据必须满足的语义要求，关系模型应采用统一的系统的方法处理此类完整性  
关系操作：选择、投影、连接（一道编程题，根据题目要求，写出关系代数的表达式）  
**第3、4、5章 SQL语言**（看PPT，掌握表的定义和增删改查即可）  
语言概述:

特点

1、一体化语言，综合统一:集DDL、DML、DCL于一体,运行时模式修改不受限制,用户数据与系统数据的统一

2、非过程化语言：指出做什么即可，存取路径和操作过程对用户透明

3、集合操作方式：一次一集合

4、一种语法，两种方式：联机交互或嵌入高级语言

5、语言简洁，易学易用

SQL语言与数据库三级模式结构的对应

1、模式：对应基本表

SQL支持基本表的定义、操纵、删除等

2、外模式： 对应视图或基本表

SQL支持视图的定义、操作、删除等.

3、内模式：对应存储文件

SQL支持索引的定义、使用、删除

SQL支持数据存储位置定义等  
数据定义语言DDL：定义基本表、数据约束（缺省值、主码、外码、check约束，要求会写）、视图  
数据操纵语言DML：增、删、改、查（重点）  
 单表：条件查询、分组统计，排序，PPT上的都考  
 连接：普通内连接  
 嵌套：内外层无关的嵌套查询  
 触发器、游标、存储过程不考！  
**第6章 关系理论（看PPT，理解第二范式和第三范式即可）**  
函数依赖：  
1.定义：  
已知：关系模式R(U)，X、Y是U的子集。对R(U)的任意一个关系r，若r中不可能存在两个元组，它们在X上的取值相同，但在Y上的取值不同，则：称X函数确定Y，或Y函数依赖于X，记做X-> Y。  
2、平凡的和非平凡的函数依赖  
非平凡函数依赖：X->Y，且Y不是X的子集。  
平凡的函数依赖：X->Y，但Y是X的子集。  
3、完全函数依赖和部分函数依赖  
若：X -> Y，且对于X的任意一个真子集Xˊ，X’ 非函数确定 Y  
则：称Y完全函数依赖于X，记作X f-> Y;  
若：X -> Y，且存在X的一个真子集Xˊ， Xˊ -> Y  
则：称Y部分函数依赖于X，记作X p-> Y;  
4、相互决定  
若：X -> Y，且Y -> X，则：X <–>Y  
5、传递依赖  
若：X -> Y（非平凡），且Y 非函数决定 X，Y -> Z，则：Z对X传递依赖  
X 只由一个属性组成，则 X ->Y 必定是完全函数依赖  
X 是多属性组成，才考虑 X ->Y 是否是部分函数依赖  
规范化: 一个属于低级范式的关系模式，通过模式分解可以转化为若干个属于高级范式的关系模式的集合，这种过程就叫规范化。  
1NF：  
每一个分量必须是不可再分的最小数据项，满足这一条件的关系模式属于第一范式。  
即：不能以集合、序列等作为属性值，不能大表套小表。  
2NF（不存在部分函数依赖）：  
若 R∈1NF，且每一个非码属性完全函数依赖于码，则 R∈2NF  
存在问题  
 数据冗余：对选择了 K 门课的学生，系和住处存在数据重复，占用空间  
 插入异常：若向一个S-L-C插入一个未选课的学生,码值一部分为空无法插入,学生的固有信息无法插入  
 删除异常：若S-L-C中一学生退选了唯一一门课,主属性被删除,整个元组也被删除,学生固有信息被删除  
 更新复杂：对转系，需要把元组中的住处一并更改，若选择了k门课，需要更新k次,造成修改的复杂化  
问题解决：非 2NF 模式的分解与规范化  
非主属性有两种：一种完全依赖于码; 一种是部分依赖于码， 将第二种属性分离，加上决定它们的码的子集组成新关系  
结论：  
 没有非主属性的关系模式必属于 2NF  
 单属性作码的关系模式属于 2NF  
 二目关系模式属于 2NF  
 若 R 属于 1NF，但 R 不一定属于 2NF  
3NF（不存在传递函数依赖）：设 R∈1NF，若不存在这样的码 X，属性组 Y，及非码属性 Z (Z 不是 Y 的子集)，使得 X -> Y，（Y 非函数决定 X），Y -> Z 成立，则 R< U，F > ∈ 3NF。  
问题：数据冗余，更新复杂  
结论：不存在非主属性的关系模式属于 3NF  
 二目关系模式属于 3NF，不会存在非码属性对码的部分依赖和传递依赖  
 若 R 属于 3NF，那么 R 一定属于 2NF。  
 若 R 属于 2NF，但 R 不一定属于 3NF。

**第7章 数据库设计**  
 数据库设计阶段及其任务（了解各阶段名称）  
 1.需求分析阶段  
 准确了解与分析用户需求  
 （包括数据与处理）  
 整个设计过程的基础，最困难、最耗费时间的一步  
 了解原系统的工作概况和不足  
 收集并分析用户对新系统的需求  
 将上述调研结果进行分析、概况和抽象，并用模型表达出来  
 (1、信息要求：获取哪些信息（具体的内容和特性）？-> 存储哪些数据？  
 (2、处理要求：系统要做哪些处理？响应时间？处理方式？  
 (3、数据约束条件：安全性、完整性、响应时间等等。

2.概念结构设计阶段  
 是整个数据库设计的关键  
 对用户需求进行综合、归纳与抽象，形成一个独立于具体DBMS的概念模型  
 分类(classification): 定义某一类概念作为现实世界中一组对象的类型**.**在E-R模型中,实体型就是这种抽象  
 聚集(aggregation): 定义某一类型的组成成分。在E-R模型中,若干属性的聚集组成实体型,属于聚集抽象  
 概括(generalization): 定义类型之间的一种子集联系,即将具有某些共同特征的对象合并成更高一层的对象  
 3.逻辑结构设计阶段  
 将概念结构转换为某个DBMS支持的数据模型  
 对逻辑数据模型进行优化  
 4.数据库物理设计阶段  
 为逻辑模型选取一个最适合应用环境的物理结构（包括存储结构和存取方法）  
 5.数据库实施阶段  
 运用DBMS提供的语言工具、高级语言，根据数据库设计方案，建立数据库  
 编制与调试应用程序  
 组织数据入库  
 6.数据库运行与维护阶段  
 系统试运行后，可投入正式运行  
 数据库应用系统运行过程中，需要对其进行评价、调整与修改

**ER图绘制及转换为表结构（重点，看PPT）**  
1.强实体：指不依赖于其它实体而独立存在的实体型  
弱实体：指必须依赖于其它实体才能存在的实体型  
属主：被弱实体型所依赖的实体型称为标识属主，简称为属主  
弱实体型只有部分标识码，其属主的标识码与部分标识码绑在一起才能构成弱实体型的完整标识码  
在 E-R 图中，单边矩形框表示普通的强实体，双边矩形框表示弱实体  
2.简单属性和复合属性  
简单属性：由单个元素构成的独立存在的属性，也称为原子属性  
复合属性：由多个元素构成的属性，其中每个元素都可独立存在  
是否将复合属性分解为更小的属性：用户是否要单独使用子属性  
E-R 图表示：用椭圆来描述复合属性的子属性，并置于所属复合属性的下方，用无向边将子属性与其所属的复合属性连接起来。  
3.单值属性和多值属性  
单值属性：每个实体在该属性上只有唯一的一个取值  
多值属性：实体在该属性上可以有多个属性值  
在 E-R 图中，我们用单边椭圆表示单值属性，用双边椭圆表示多值属性  
4.基属性和派生属性  
派生属性（导出属性）：指可由其它属性值计算得到的属性  
基属性（存储属性）：指不能由其它属性值导出的属性  
在 E-R 图中，用虚线椭圆表示派生属性  
最大基数：对实体集 A 中的每个实体，实体集 B 中最多有多少个实体与之对应。根据最大基数的不同，联系分为 1:1，1:N，M:N 联系  
最小基数：是对联系类型的更强、更细致的约束，它指对实体集 A 中的每个实体，实体集 B 中最少有多少个实体与之对应。若最小基数为 0，则称 B 是可选的；若最小基数为 1，则称 B 是强制的  
弱实体型与其属主间的依赖关系属于标识联系，在 E-R 图中，我们用双边菱形框来表示标识联系  
关联实体：增设关联实体，将多元联系转化为多个二元联系

在 E-R 图中，一般用带有矩形外框的菱形来表示关联实体

（1）超类：包含一个或多个实体子集的实体型  
（2）子类：实体集中某个实体子集的共同性质和特征的抽象  
（3）超类和子类之间具有超类/子类联系，且联系的类型为 1:1  
（4）在 E-R 图中，用线将超类与三角连接起来，再用线连接三角与各子类  
E-R 图中，超类与子类间的双线表示强制参与约束，单线是可选参与约束  
约束超类中的实体是否可以同时属于多个子类。如果不可以，称该联系遵循不相交规则；反之，称遵循交叠规则  
E-R 图中，交叠规则不需要特别描述，不相交规则需要在超类和子类间的小三角中加上符号“×”来表示

5.确定实体、实体属性的方法  
仔细查看需求说明，从中挑出名词和名词词组  
组织感兴趣的人、地方和概念，或者组织环境中客观存在的对象也要抽象成实体型  
注意：有些名词代表实体，有些名词只表征实体的性质和特征。如，员工代表实体，而姓名却只表征员工的一种特征

**第9章 数据库恢复技术（填空，判断，简答）**  
 事务的基本概念和ACID特性  
 事务:用户定义的一个数据库操作序列，这些操作要么全做要么全不做，是一个不可分割的工作单位.  
 在关系数据库中，一个事务可以是一条sql语句，一组sql语句或整个程序  
 事务的开始与结束可以由用户显式控制（若用户没有显式定义，由数据库管理系统按默认自动划分），语句有  
 BEGIN TRANSACTION 开始事务  
 COMMIT 将对数据库的更新写回至磁盘上的物理数据库中  
 ROLLBACK： 回滚，将所有对数据库的已完成操作撤销  
 事务的ACID特性  
 原子性：是数据库的逻辑工作单元，要么都做，要么都不做。  
 一致性：使数据库从一个一致性状态变到另一个一致性状态。  
 隔离性：并发执行的各个事务之间不能互相干扰。  
 持续性：事务一旦提交，对数据库的修改是永久的。  
 事务内部的故障：事务在运行至正常终止点前被终止  
 可预期故障（如转帐资金不足） ：应用程序处理  
 不可预期故障：运算溢出、死锁、违背完整性等不能由程序处理  
 问题：事务未到达终点，数据库处于不正确状态

系统故障：指造成系统停止运转的任何事件，使得系统要重新启动  
 所有活动事务都非正常中止  
 已完成的事务，尚未写盘或部分写盘——数据不一致  
 未完成的事务，已部分写盘——数据不一致

故障产生的原因与后果  
 硬件故障、软件错误、误操作、恶意破坏  
 数据库被破坏，或数据库处于不一致状态  
 日志（概念，原则）和备份（概念，类型）  
 日志文件是用于记录事务对数据库的更新操作的文件，包括以记录为单位的日志文件（开始、结束、更新操作）和以数据块为单位（数据块）的日志文件  
 登记日志的原则  
 登记次序与并发事务执行的时间次序严格一致  
 先写日志原则后写数据库

故障恢复策略（重点掌握事务内部故障）  
 事务故障恢复：利用日志文件撤销此事务对数据库进行的修改  
 反向扫描日志文件，查找该事务的更新操作  
 对更新操作执行逆操作，将日志文件中更新前的值写入数据库  
 继续逆向扫描，对所有的更新操作做同样处理  
 遇到该事务的开始标志，扫描结束

系统故障恢复：撤销故障发生时未完成的事务，重做已完成的事务  
 不带检查点的恢复方法  
 生成REDO和UNDO队列：正向扫描日志文件  
 故障发生前已提交的事务，将其事务标识放入REDO队列  
 故障发生前未提交的事务，将其事务标识放入UNDO队列  
 UNDO队列  
 逆向撤销处理 【数据若写盘，日志一定也写盘】  
 REDO队列  
 正向重做处理 【提交的事务，日志一定已写盘】  
 缺点：效率低，很多需要重做处理的事务实际上已经将它们的更新操作结果写到数据库中，然而恢复子系统又重新执行了这些操作，浪费大量的时间，产生不必要的REDO

带检查点的恢复方法  
 （1）检查点（CheckPoint）  
 数据库正常运行的一个时间标志  
 在检查点时刻，系统强行将缓冲区内容写盘  
 （2）检查点种类  
 自动：系统参数Recovery Internal（RI）设置  
 人工：CheckPoint命令  
 （3）检查点记录内容  
 建立检查点时刻，所有正在执行的事务清单  
 这些事务最近一个日志记录的地址  
 （4）执行检查点命令，所作的动作和顺序  
 缓冲区中的日志写盘  
 在日志文件中，增加一个检查点记录  
 缓冲区中的数据写盘  
 重新开始文件新增一行，指示检查点在日志文件中的位置  
 （5）恢复过程  
 在重新开始文件中，找最后一个检查点在日志中的位置  
 在日志中取该检查点，得到活动事务清单放入UNDOLIST  
 从该检查点开始，正向扫描日志文件，新开始事务加入UNDOLIST，结束事务将其从UNDOLIST移入REDOLIST。  
 UNDOLIST中的事务：做UNDO处理（从后向前）  
 REDOLIST中的事务：做REDO处理（从前向后）

**第10章 并发控制**  
 锁的概念、分类和相容性

1. 封锁：事务T对某个数据对象操作之前，先向系统申请对该数据对象加锁，加锁后就对该数据对象有了一定的控制权，在释放锁之前，其他事务对该数据对象的使用将受到限制
2. 排它锁（写锁、X锁）：事务T对数据对象A加X锁，则其他事务不能对A加任何锁
3. 共享锁（读锁、S锁）：事务T对数据对象A加S锁，则其他事务只能对A加S锁，不能加X锁  
    

三个封锁协议

1级封锁协议  
 事务T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  
 正常结束（COMMIT）  
 非正常结束（ROLLBACK）  
 1级封锁协议可防止丢失修改  
 在1级封锁协议中，如果是读数据，不需要加锁的，所以它不能保证可重复读和不读“脏”数据。

2级封锁协议  
 1级封锁协议 + 事务T在读取数据R前必须先加S锁，读完后即可释放S锁  
 2级封锁协议可以防止丢失修改和读“脏”数据。  
 在2级封锁协议中，由于读完数据后即可释放S锁，所以它不能保证可重复读。

3级封锁协议  
 1级封锁协议 + 事务T在读取数据R之前必须先对其加S锁，直到事务结束才释放  
 3级封锁协议可防止丢失修改、读脏数据和不可重复读。

  
 死锁的预防和解决  
 预防  
 （1）一次封锁法  
 规则：要求每个事务必须一次将所有要使用的数据全部加锁，否则就不能继续执行  
 缺点：  
 降低并发度  
 难于事先确定要使用的数据：若将事务在执行过程中可能要封锁的数据对象全部加锁，则进一步降低了并发度  
 （2）顺序封锁法  
 规则：预先对数据对象规定一个封锁顺序  
 所有事务按顺序封锁  
 缺点：  
 维护成本高：数据对象非常多，且随数据的更新而不断变化，维护其封锁顺序非常困难，成本很高；  
 难于实现：事务要封锁的数据对象常常随事务的执行结果而动态变化，很难事先确定待封锁对象，因此也就很难按规定的顺序去施加封锁。

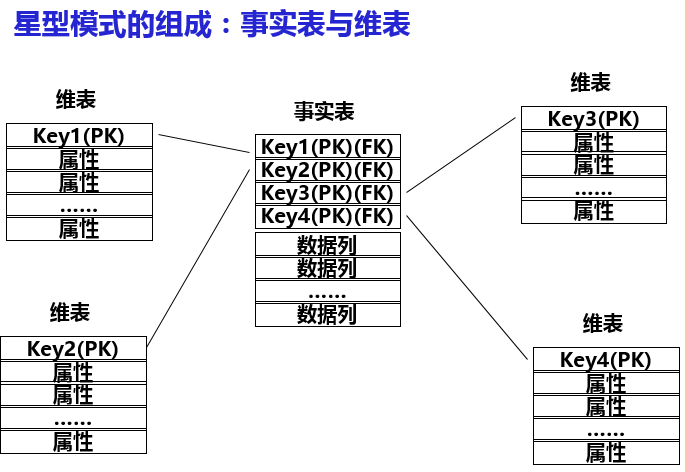
解决：回滚一个或多个事务，且使得代价最小  
 回滚策略：事务开始时，赋予每个事务唯一的时标，当事务T1对T2封锁的数据请求封锁时：  
 若 T1比T2老  
 则T1打伤T2，T1胜利，T2退回  
 否则T1等待  
 事务退回后重试时，时标保持不变

串行化调度（概念，判断，冲突和非冲突的调换）  
 可串行化调度：多个事务的并发执行是正确的，当且仅当其结果与按某一串行次序执行他们时的结果相同。  
 冲突操作：不同的事务对同一个数据的读写操作和写写操作，若改变冲突操作的次序，会影响数据库状态  
 冲突可串行化调度：对并发调度Sc，在保证冲突操作次序不变的情况下，若能够通过交换两个事务不冲突操作的次序，获得另一个调度sc’，且Sc‘是串行的，则称Sc为冲突可串行化调度。若一个调度是冲突可串行化调度，则一定是可串行化调度  
 **第13章 数据仓库（了解为主）**  
 数据仓库的定义和特点  
 1、定义  
 一种面向主题的、集成的、不可更新的、随时间不断变化的数据集合，用于支持企业或组织的决策分析处理.  
 本质上和数据库一样，是计算机内有组织、可共享的数据集合.  
 2、特征  
 面向主题 ：数据按照主题来建模、优化、组织.  
 集成性 ：数据来自企业内、外的多个数据源，经过清洗、转化、集成后存入数据仓库.  
 不可更新：数据仓库的数据一般不执行更新操作，对事务数据库的更新，通过数据集成过程反映到数据仓库.  
 时变性：不断有新数据追加到数据仓库；不断有超过存储期限的数据被删除，引起数据仓库中与时间有关的聚集数据的重新计算.。

ETL概念  
 ETL：Extract/Transformation/Load  
 针对外部数据源进行的数据抽取、转换、装载  
 针对加载的数据，进行不同粒度的汇总和综合

多维数据与星型建模  
 多维数据模型是数据分析时用户的数据视图，是面向分析的数据模型，用于给分析人员提供多种观察的视角和面向分析的操作

星型模式  
 事实表：描述实际发生的定量业务数据（如销售量、订单），包含组合关键字和事实数据。  
 维表  
 维度：人们总是从不同角度来观察事实，每个观察角度被称为一个维度，维表就是对维度的描述  
 包含：简单的主关键字和若干非主属性  
 事实表和维表的关系  
 一对多的关系：一个事实表周围环绕多个维表，形成星型结构  
 代表人们从多个维度观察同一事实数据



OLAP操作（联机分析处理）有哪些：  
 多维分析操作  
 切片（slice）：在某一个维上，选定一个维成员的动作  
 slice for time=“Q2”  
 切块（dice）：在某一个维上，选定某一区间的动作  
 dice for (location=“Montreal” or “Vancouver”) and (time=“Q1” or “Q2”) and (item=“home entertainment” or “computer”)  
 下钻（drill - down）：展开细节  
 上卷（roll-up）：在一定粒度上，取消细节，向上汇总  
 旋转（pivot）：旋转维度或维度的某个层次

**第14章 NoSQL数据库**  
 概念演变和基本理念（了解为主）  
   
 NoSQL 基本理念  
 不一定遵循关系数据库的一些基本要求，例如：关系表结构、标准SQL语言、事务的ACID特性等。  
 相比传统数据库， NoSQL 数据存储被简化，更灵活。  
 主要数据模型（文档、图、列簇、键值）

内存数据库概念和特点

概念  
 MMDB：Main Memory Database  
 内存数据库是将内存作为主存储设备，将数据放在内存中，直接操作的数据库

特点  
 “主拷贝”或“工作版本”常驻内存，直接的内存访问  
 与磁盘数据库同步，具有数据恢复机制  
 并发处理能力  
 高性能：微妙级的查询响应  
 高吞吐率和低访问延迟  
 硬件相关性

Redis数据库的概念、特点、数据结构（五种）  
 Redis 数据库概述  
 Remote Dictionary Server的缩写  
 开源高性能数据库：C语言开发  
 内存数据库：是基于内存的，亦可持久化  
 键值数据库：采用key-value数据模型  
 分布式数据库：支持分布式集群，具有扩展性

Redis数据库特性（记住每个特性名）  
 持久化：可以将内存中的数据保存到磁盘上，重启时可以再次加载并使用  
 两种持久化方式  
 RDB：在指定时间间隔内，生成数据集的时间快照  
 AOF：记录服务器执行的所有写操作，并在服务器启动时，重新执行这些命令来还原数据集。

高性能：读写速度大约 10万次 / 秒  
 SET操作每秒11万次  
 GET操作每秒8万次

主从复制：支持Master和多个Slave的主从关系  
 Slave会自动同步Master服务器的数据。  
 主服务器宕机，从服务器可替代主服务器为客户端服务  
 可以把持久化任务配置在从服务器上，减轻主服务器的压力  
 用途：实现读写分离、数据备份、灾难恢复等

扩展性强：支持集群架构  
 通过数据分片（sharding）实现数据的集群分布，线性可扩展，  
 部分节点失效或无法通讯时，仍然可继续处理请求  
 分片方式：  
 键空间拆分成16384个槽位，各节点负责其中部分槽位。  
 例如，一个集群由三个节点构成，则：节点A负责0-5500号哈希槽；节点B负责5501号到11000号；节点C负责11001号至16384号

客户端资源：提供多种语言的开源客户端类库  
 支持 Java，Python，C/C++，C#，PHP，JavaScript，Perl，Object-C，Ruby，Erlang 等语言  
 提供上述语言的 Redis 接口方法。如：面向 string、list、hash、set、sorted set等数据结构的各类操作。  
 提供了灵活的客户端连接 RedisClient PooledRedisClientManager 客户端缓存池

数据结构（String，Hash，List（存储有序的字符串列表, 重要数据结构之一），Set（集合数据，自动排重），Sorted set（有序且不重复的集合数据））