复习提纲

数据库系统概述

- 基本概念（概念）
  - 数据：具有一定格式或结构的符号串，属于软件范畴；
  - 数据库：是数据集合，具有统一结构形式并存放于统一存储介质内
    - 也就是：结构化、集中存储、数据共享
  - 数据库管理系统：管理数据库的系统软件
    - 文件管理系统发展而来
  - 数据库系统
    - 数据库、数据库管理系统、数据库管理员、软件平台、硬件平台
  - 数据库系统的四个特点（概念）
    - 数据的集成性
    - 数据的高共享性和低冗余性（数据的一致性和不一致性）
    - 数据独立性
    - 数据的统一管理与控制（数据的完整性检查、安全性保护、并发控制、数据库故障恢复）
  - 数据库内部结构体系（概念）
    - 三级模式：外模式、概念模式、内模式
    - 两种映射：外模式-概念模式映射、概念模式-内模式映射
  - 数据模式

数据模型

- 数据模型的基本概念（概念）
  - 数据模型：描述数据的结构，定义在该数据结构上可以执行的操作以及数据之间必须满足的约束条件
    - 组成成分：数据结构、数据操作、数据约束
  - 三种数据模型：概念数据模型，逻辑数据模型，物理数据模型
  - 三种“数据模型”和“三级模式”之间的对应关系？
    - 逻辑模型——外模式和概念模式
    - 物理模型——内模式
    - 注：概念模式不关系到数据库的实现，只到 E-R 图
- 数据模型的四个世界（概念）
  - 概念世界与概念模型
    - 概念世界、概念模型与具体的 DBMS 和计算机无关
    - 概念数据模型主要描述客观对象的数据特征及其相互关系
  - E-R 模型与 E-R 图：（应用）
    - 实体：客观存在，能相互区别的事物
    - 实体集：由具有相同实体型的实体所构成的集合
    - 属性：实体具有的某种特性或特征
    - 联系：一个实体集中的实体与另一个实体集中的实体的对应关系
    - 画 E-R 图
  - EE-R 模型与 EE-R 图：（概念）

        - IS-A 联系：如果实体集 B 是实体集 A 的一个子集，且具有比实体集 A 更多的属性，则我们称在实体集 A 与实体集 B 之间存在着一种特殊的“IS-A 联系”，其中 A 是 B 的超实体集，B 是 A 的子实体集。
        - 弱实体：如果一个实体 A 的存在需要依赖于其他实体集中的某个实体的存在，那么实体 A 被成为弱实体，例如职工 vs 家属，学生 vs 家长
- 信息世界和逻辑模型
- 关系模型：（概念）
    - 关系：
        - 行、列组成的二维表格
        - 关系的约束
            - 同一表中的属性名各不相同
            - 表中属性与属性的排放次序无关
            - 表中元组均不相同
            - 表中的元组与元组的排列次序无关
            - 表中每一分量必须是一个不可分割的基本数据项
    - 属性：每一列
    - 值域：每一个属性有一个取值范围
    - 元组：每一行
    - 关系模型：符合上述条件
    - 关系模式：一个关系的关系和属性名的集合构成该关系的关系模式
    - 关系模型：以符合 7 个条件的二维表为基本数据结构建立的模型
    - 关键字：关系中的一个属性集的值能唯一标识关系中的一个元组，且又不含多余的属性值，则称该属性集为该关系的关键字
        - 关键字可有多个，“关键字”=“候选关键字”
        - 主关键字：候选关键字选一个
        - 外关键字：表 A 的某属性集（注意：外关键字可以多个）取值来源于表 B 的主键，则该属性集即为 A 的外关键字

关系数据库系统

- 关系数据库系统的衡量准则
    - 空值（NULL）：无意义或当前未知的值
- 关系模型数学理论—关系代数
    - 关系模型（概念）
        - 关系数据结构
            - 表结构：
                - 表框架：n 个命名的属性组成
                - 表的元数：n 是表的元数
                - 表的基数：一个表框架可存放 m 个元组，m 是表的基数
            - 关系：
                - 二维表的性质
                    - 元组个数有限性
                    - 元组的唯一性
                    - 元组次序无关性

        - 元组分量的原子性
        - 属性名唯一性
        - 属性的次序无关性
        - 分量值域同一性
      - 满足这 7 个性质的二维表->关系
      - 满足这 7 个性质的二维表为基本数据结构所建立的模型->关系模型
    - 关键字：
      - 候选关键字：注意不能只根据一张表中已有元组来判断其由哪些候选键，必须根据属性语义+联系才能判断。
      - 主关键字
      - 外关键字
    - 关系数据库：关系子模式-视图（view）
- 关系操纵：
  - 五种基本操作：元组选择、属性指定、两个关系的合并、元组插入、元组删除
  - 数据查询：
    - 两个关系的合并：通过多个关系之间的两两合并，可以将其最终合并为一个关系
    - 单个关系内的元组选择：选择满足指定条件的元组
    - 单个关系内的属性指定：选择结果所需要的属性（值）
  - 数据删除
    - 基本单位：元组；
    - 操作过程：确定被删除的元组；执行删除操作
  - 数据插入：在指定关系中插入一个或多个新的元组
    - 一条操作只能插一个关系
  - 数据修改：在一个关系内修改指定元组某些列上的值
    - 操作过程：删除需要修改的元组；然后插入修改后的新元组
  - 空值的处理
    - 空值的比较运算：不能参与比较运算
    - 含有空值对象的运算表达式计算：进行+、-、*、/的运算的结果均应该为空值
    - 含有空值的逻辑表达式，则其运算结果为逻辑假
    - 含有空值成员的集合上的统计运算：元组时空值可以不参与计算，列时空值应该考虑进来
    - SUM,AVG,MAX,MIN,COUNT 操作时集合的“空值”元素不统计在内
    - 统计计算中空集 SUM,AVG,MAX,MIN 的统计结果为空值；COUNT 统计结果为 0；
- 关系中的数据约束
  - 实体完整性约束：主键中的属性不能有空值
  - 参照完整性约束：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
  - 用户定义的完整性：用户自己定义的属性取值约束

- 关系的表示（概念）
  - 关系的表示
    - 关系是元组的集合，元组是元组分量的集合，一个 n 元关系是一个 n 元有序组的集合
  - 笛卡尔乘积
- 关系操纵的表示（应用）
  - 关系上的五种基本操作<==>关系代数中的五种基本运算
    - 元组选择<==>选择运算（选择元组）
    - 属性指定<==>投影运算（选择属性）
    - 关系的合并<==>笛卡儿积
    - 元组的插入<==>并运算
    - 元组的删除<==>差运算
  - 基本运算的应用实例
- 关系模型与关系代数（概念）
- 关系代数中的扩充运算（应用）
  - 交：【补充】
  - 除法：【补充】
  - 联接与自然联接：【补充】
  - 外联接：
  - 扩充运算与基本运算之间的关系
  - 扩充运算的应用实例
- 关系代数实例（应用）
  - 综合的关系代数应用实例

- 关系数据库语言 SQL'92（概念）
  - SQL 数据定义功能（应用）
    - SQL 中的数据类型
    - 基本的表结构定义命令
      - CREATE TABLE 表 (列名 数据类型 [NOT NULL], …);
    - 基本的表结构修改命令
      - ALTER TABLE 表 ADD 列名 数据类型 [NOT NULL];
      - ALTER TABLE 表 DROP 列名;
      - DROP TABLE 表
  - SQL 数据操纵功能（应用）
    - SQL 语言与关系代数的关系
    - 映像语句结构
    - 基本查询功能：LIKE，IS NULL 谓词；表的联接查询与自联接查询；ORDER BY 子句的功能
    - 嵌套查询：IN，SOME/ANY/ALL，EXISTS 等谓词；
      - 相关子查询：嵌套查询中的子查询只需要被执行一次，然后利用所获得的中间查询结果来计算外层的查询语句，由内到外
      - 独立子查询：子查询中调用外层查询中的表及其元组变量，外层表变，都需要重新执行子查询来获得相关的中间查询结果，由外到内
    - 子查询的合并：UNION/INTERSECT/EXCEPT [ALL]运算：并、交、

差

- 统计查询：
  - 统计与分组统计查询；
  - 空值与空集在统计函数中的处理方法
- 复杂数据查询：
  - **两层的 NOT EXISTS 嵌套结构，以实现关系代数中的除法运算的查询功能。**

- SQL 的更新功能（应用）
  - 删除功能：DELETE FROM 表 WHERE 条件
  - 插入功能：INSERT INTO 表(列 1, 列 2, …) VALUE (1, 2, …)| subq
    - 常量元组的插入
    - 带子查询的元组插入
  - 修改功能：UPDATE 表 SET 列名=expr| NULL| subq WHERE 条件
- 视图
  - 视图概念：由若干张表经映像语句构筑而成的表
  - 视图与基表的区别：
    - 被称为视图的二维表本身并不实际存在于数据库内，而是保留了其构造信息。视图被称为“虚表”
    - 当用户执行视图上的访问操作时，数据库管理系统讲根据视图的定义命令将用户对于视图的访问操作转换成相应基表上的访问操作
  - 视图的定义命令：CREATE VIEW 视图 AS 映像语句 [WITH CHECK OPTION]
    - 嵌套定义功能
  - 视图的删除命令：DROP VIEW 视图
    - 视图删除中的连锁反应
  - 视图的优点：提高了数据独立性、简化用户观点、提供自动的安全保护功能

数据库的安全性与完整性保护（概念）

- 数据库的安全性保护
  - 数据库安全：
    - 要求数据库用户：通过规定的访问路径、按照规定的访问规则来访问数据库内的数据，并且接受来自 DBMS 的各种检查
    - 主体集访问客体集，数据库安全研究的是有关实体的主/客体划分以及主/客体之间的访问关系的控制
  - SQL 对数据库安全的支持：
    - C1 级别的支持：主体、客体及主/客体分离；身份标识与鉴别、数据完整性、自主访问控制与授权功能
  - SQL 语言所提供的与数据库安全保护有关的命令（应用）TO 和 FROM 不要弄混了
    - GRANT 操作权限列表 ON 操作对象 **TO** 用户名列表 [WITH GRANT OPTION]

- REVOKE 操作权限列表 ON 操作对象 **FROM** 用户名列表 [RESTRICT|CASCADE]（连锁回收或不存在连锁回收问题时回收权限，否则拒绝回收）

- 数据库的完整性保护
  - 完整性规则的三个内容：
    - 实体完整性
    - 参照完整性
    - 用户定义完整性
  - DBMS 提供数据完整性约束条件的定义和检查的优点？
  - 完整性约束的设置、检查与处理：在 SQL 语言的 CREATE TABLE 命令中提供的完整性约束定义子句（应用）(S 勿漏！)
    - FOREIGN KEY (列名) REFERENCES 表 ON 操作 CASCADE| RESTRICT| SET NULL
    - CREATE TABLE Student(
      - sno NUMBER(5) CONSTRAINT C1 CHECK(sno BETWEEN 90000 AND 99999),
      - sname VARCHAR(20) CONSTRAINT C2 NOT NULL,
      - sage NUMBER(3) CONSTRAINT C3 CHECK(sage<29)
      - CONSTRAINT main_key PRIMARY KEY(sno),
      - CONSTRAINT fk_dept FOREIGN KEY(外键) REFERENCES 表(主键) ON UPDATE CASCADE, ON DELETE RESTRICT,
      - …);

事务处理、并发控制与故障恢复技术

- 事务处理（概念）
  - 事务的定义与 ACID 性质
    - 事务：由某个用户所执行的一个不能被打断的对数据库的操作序列
    - ACID：
      - 原子性：不可分割的操作序列
      - 一致性：数据库从一个一致的状态转换到另一个一致的状态
      - 隔离性：事务之间执行相互独立，互不干扰
      - 持久性：事务执行完成后对数据库的所有更新应永久地反映在数据库中
  - 事务活动及其状态转换图
    - “提交”和“异常终止”意味着一个事务地执行结束
  - 事务控制及相关的参数设置语句?
    - 事务的提交与回滚 commit；回滚 rollback；中止：abort
    - 事务的读/写类型与隔离级别：SET TRANSACTION ISOLATION LEVEL [READUNCOMMITED| READCOMMITED| READREPEATABLE| SERIALIZABLE]
  - 三种数据不一致现象：丢失修改，脏读，不可重复读
- 并发控制技术（概念）
  - 事务
    - 事务的并发性，并发控制

- 调度：一个或多个事务中的数据库访问操作，按照这些操作被执行的时间排序所形成的一个操作序列
- 串行调度：一个事务所有操作结束，另一个事务所有操作以此类推
- 可串行化调度：某个调度对数据库状态的影响和某个串行调度相同
- 冲突：调度中一对连续操作，交换它们的执行顺序，涉及的事务中至少有一个的行为会改变
- 冲突可串行化：非冲突交换将调度 S 转换成串行调度（冲突等价）
  - 冲突可串行化调度一定是可串行化调度，可串行化调度不一定是冲突可串行化调度
- 冲突可串行化的判定方法：

■ 封锁
- 共享锁（S 锁）
- 排它锁（X 锁）
- 锁相容矩阵：理解 S 锁和 X 锁

		其它事务已持有的锁		
		X锁	S锁	-
当前事务申请的锁	X锁	No	No	Yes
	S锁	No	Yes	Yes

- 合适事务：如果一个事务在访问数据库中的对象 A 之前按照要求申请对 A 的封锁，在操作结束后释放 A 上的封锁，这个事务就是合适事务
- 基于封锁技术的并发控制实现方法
- 封锁协议：
  - 三级封锁协议
    - 一级封锁协议：写前获取 X 锁直到事务结束才释放；防止丢失修改
    - 二级封锁协议：一级+读前获取 S 锁直到读结束才释放；防止丢失修改、脏读
    - 三级封锁协议：**一级**+读前获取 S 锁直到事务结束才释放；防止丢失修改、脏读、不可重复读
  - 两阶段封锁协议，2PL 协议；采用两阶段封锁协议的事务：2PL 事务
    - 第一个阶段（扩展阶段）：申请并获得锁
    - 第二个阶段（收缩阶段）：释放所有申请获得的锁
- 合法调度：满足下面三个要求

❑ 在每一个事务T_i中
- ➢ 第1点：采用如下的封锁协议：
  - 读动作 $r_i(A)$ 之前必须有 $sl_i(A)$ 或 $xl_i(A)$，而且在两者之间没有 $u_i(A)$
  - 写动作 $w_i(A)$ 之前必须有 $xl_i(A)$，而且在两者之间没有 $u_i(A)$
  - 每一个 $sl_i(A)$ 或 $xl_i(A)$ 之后必须有一个 $u_i(A)$

- 第2点：必须遵循‘两阶段封锁协议’
  - 在任何 $sl_i(A)$ 或 $xl_i(A)$ 之前不能有 $u_i(B)$
    - A和B可以是同一个数据对象

- 第3点：保证事务调度的合法性
  - 对任意两个不同的事务 T_i 和 T_j，其调度必须满足：
    - 如果 $xl_i(A)$ 出现在调度中，那么后面不能再有 $sl_j(A)$ 或 $xl_j(A)$，除非中间插入了 $u_i(A)$
    - 如果 $sl_i(A)$ 出现在调度中，那么后面不能再有 $xl_j(A)$，除非中间插入了 $u_i(A)$

- 两阶段封锁协议与冲突可串行化的关系：由 2PL 事务所构成的任意合法调度 S 都是冲突可串行化的
- 多粒度封锁：封锁粒度与多粒度封锁
  - 意向共享锁 IS 锁：加 N，表示准备在 N 的某些后裔结点加 S 锁
  - 意向排它锁 IX 锁：加 N，表示准备在 N 的某些后裔结点加 X 锁
  - 共享意向排它锁 SIX 锁：加 N，表示 N 本身加 S 锁，并准备在 N 的某些后裔结点加 X 锁
  - 申请封锁：自上而下；释放封锁：自底向上
  - 优点：减少加锁、解锁的开销；提高了系统的并发度

- 数据库恢复技术（概念）
  - 数据库恢复的含义：在数据库遭受破坏后及时进行恢复的功能
  - 数据库恢复的方法：利用数据冗余原理，将数据库中的数据在不同的存储介质上进行冗余存储，当数据库本身收到破坏时，可以利用这些冗余信息进行恢复
  - 数据库恢复常用措施：数据转储、日志、数据库镜像
  - 数据库故障的分类
    - 小型故障（事务范围内故障）：事务内部故障
    - 中型故障（系统停止工作但磁盘数据不受影响，可以恢复）：系统故障、外部影响
    - 大型故障（内存及磁盘数据严重破坏，需要彻底恢复）：磁盘故障、计算机病毒、黑客入侵
  - 数据库故障恢复三大技术
    - 数据转储：静态转储/动态转储，海量转储/增量转储，
    - 日志：是由数据库系统创建和维护的，用于自动记载数据库中修改型操作的数据更新情况的文件
      - 日志的内容：每个更新操作的事务标识、更新对象、更新前的值和/或更新后的值；每个事务的开始、结束等执行情况；其它信息
      - 日志的组成：事务的更新操作的执行情况
      - 日志的作用：确保事务执行的原子性、实现增量转出、实现故障恢复
      - 日志的记载原则：先写日志后修改数据库
      - 三种类型的日志：

        - UNDO 日志
        - REDO 日志
        - UNDO/REDO 日志
      - 如果在恢复过程中再次出现系统崩溃，如何进行系统恢复？再次恢复即可
    - 事务的撤销（UNDO）与重做（REDO）
- 恢复策略：小型/中型/大型故障的恢复策略
  - 小型故障的恢复：利用未结束事务的 UNDO 操作进行恢复
  - 中型故障的恢复：非正常中止的事务：执行 UNDO 操作；已完成提交的事务：还留在内存缓冲区，未存入磁盘，故障时内存缓冲区数据丢失：执行 REDO 操作
  - 大型故障：先利用后备副本进行数据库恢复，再利用日志进行数据库的恢复。步骤如下：

1. 将后备副本中的数据拷贝到数据库中
2. 检查日志文件：确定哪些事务已经执行结束，哪些尚未结束
3. 按照日志的记载顺序
   a) 逆向：对尚未结束的事务作撤消处理(undo)
   b) 正向：对已经结束的事务作重做处理(redo)

数据库中的数据交换（概念）

- 数据交换的管理
  - 连接管理：负责在数据主、客体间建立实质性的关联，包括服务器指定、内存区域分配等；也可断开两者之间的关联；
    - CONNECT TO <连接目标>；DISCONNECT <断开对象>
  - 游标管理：
    - 游标的定义：EXEC SQL DECLARE 游标名 CURSOR FOR 子查询 [ORDER BY …] [FOR 权限]
      - 注意比如 SELECT … FROM … WHERE cid = :cust_id…有冒号的这个变量是为了和主程序变量交互而定义的新变量
    - 游标的打开：EXEC SQL OPEN 游标名
    - 游标的使用：WHILE (TRUE){EXEC SQL FETCH 游标名 INTO :agent_id, :dollar_sum; printf(…)}
    - 游标的关闭：
      - EXEC SQL WHENEVER NOT FOUNT GOTO FINISH;
      - 。。。（游标使用）
      - FINISH: EXEC SQL CLOSE 游标名
    - 可滚动游标的定义：EXEC SQL DECLARE 游标名 [INSENSITIVE] [SCROLL] CURSOR [WITH HOLD] FOR 子查询 [ORDER BY …] [FOR 权限]
    - 可滚动游标在数据更新命令中的使用：EXEC SQL FETCH [{NEXT | PRIOR | FIRST | LAST value_spec} FROM] 游标名 INTO :agent_id

- 动态 SQL：根据实际情况生成并调用的 SQL 语句
  - SQL 正文动态可变、变量个数动态可变、类型动态可变、SQL 语句引用对象动态可变
  - 和静态 SQL 对比

❑ 静态SQL
- 在通过预编译时，SQL命令就被分析并为它们的执行作好了相应的准备工作。在程序运行时，只需要调用预先优化好的访问路径
  - 优点：性能好
  - 缺点：只能根据缺省参数值进行优化，其访问路径并非是最优访问路径

❑ 动态SQL
- 程序在运行时动态生成的SQL命令
  - 优点：可以根据运行时的数据库最新情况选择最优访问路径
  - 缺点：动态地进行SQL语句的语法分析和访问路径选择

  - 什么情况下使用动态 SQL?

2. 在什么情况下需要使用动态SQL?
- 应用程序需要在执行过程中生成SQL语句；
- SQL语句用到的对象在预编译时不存在；
- 希望SQL语句的执行能够根据执行时的数据库系统内部的统计信息来采用最优的访问策略

数据库的物理组织（概念）

- 索引技术与散列技术
  - 顺序文件的组织方式
  - 索引文件的组织方式
    - 在顺序文件上的索引技术：稠密索引、稀疏索引、多级索引（应用）

- 稠密索引 与 稀疏索引 的区别
  - 索引文件的定义不同
    - ❖稀疏索引只能用于顺序文件上的索引组织
    - ❖稠密索引中的每个索引项对应数据文件中的一条记录，而稀疏索引中的每个索引项则对应数据文件中的一个磁盘块
  - 需要的磁盘空间大小不同
  - 在记录的查找定位功能上存在差别：
    - ❖稠密索引：可以直接回答是否存在键值为K的记录
    - ❖稀疏索引：需要额外的磁盘I/O操作，即需要将相应的数据文件中的磁盘块读入内存后才能判别该记录是否存在

- ◆ 非顺序文件中的索引技术
- ■ B/B+树文件
  - ◆ 树的结构与各个节点的组成内容
  - ◆ B+树上的搜索算法
  - ◆ B+的特点及其与 B 树的区别

▪ 组织方式不一样
- ❖ B+树：所有有效的索引关键字值都必须存储在叶结点中，其内部结点中的键值只用于索引项的查找定位。
- ❖ B树：有效的索引关键字值可以出现在B树的任意一个结点中。

▪ 内部结点不同
- ❖B+树：关键字值 + 子树指针
- ❖B树：关键字值 + 记录指针 + 子树指针
  - 因此B树结点的扇出（*即一个结点可以拥有的最大子结点数目*）较小，从而导致整个B树的高度大于B+树的高度。

▪ 随机查找效率的区别
- ❖B+树：所有关键字的查找速度基本一致
- ❖B树：依赖于关键字所在结点的层次

▪ 在B树中没有提供对索引关键字的顺序扫描功能。

▪ B树的插入、删除操作较B+树复杂。

- ■ HASH 文件：散列索引文件的组织方式

第八章 关系数据库规范化理论

- ● 概述（概念）
  - ■ 模式设计质量的评价指标：数据冗余度，插入/删除等更新异常
- ● 规范化理论
  - ■ 函数依赖（FD）
    - ◆ 各种函数依赖的定义：完全/部分 FD，平凡/非平凡 FD，直接/传递 FD（概念）
    - ◆ Armstrong 公理系统（概念）
    - ◆ 基于函数依赖的关键字定义（概念）
    - ◆ 属性集闭包的计算算法（应用）
    - ◆ 关键字的计算算法（应用）
  - ■ 与函数依赖有关的范式（概念）
    - ◆ 范式：
      - ● 1NF：不可分割
      - ● 2NF：1NF+每个非主属性都完全函数依赖于关键字
      - ● 3NF：2NF+每个非主属性都不传递函数依赖于关键字
      - ● BCNF：1NF+若 X->Y 则 X 必含有该关系模式的关键字

- ◆ 各级范式的分解方法
- ■ 规范化所引起的一些问题
  - ◆ 函数依赖的逻辑蕴涵，函数依赖集的等价（概念）
  - ◆ 最小函数依赖集及其判定条件（概念）
  - ◆ 最小函数依赖集的计算算法（应用）
  - ◆ 模式分解的无损联结性、依赖保持性及其判定方法（概念）
  - ◆ 直接到 3NF 且满足无损联结性和依赖保持性的模式分解算法（应用）

第九章　数据库设计

- ● 数据库设计概述（概念）
  - ■ 数据库设计的基本任务：在一定平台制约下，根据信息需求与处理需求设计出性能良好的数据模式
- ● 数据库的概念设计
  - ■ 数据库概念设计的过程（概念）
    - ◆ 用户分解、视图设计（实体与属性设计、联系与继承设计）、视图集成
  - ■ 视图集成的原理、策略（概念）
    - ◆ 等同：两个或多个数据对象具有相同的语义
    - ◆ 聚合：数据对象之间的一种组成关系
    - ◆ 抽取：将不同实体中的相同属性提取成一个新的实体，并构造成具有继承关系的结构
  - ■ 视图集成的步骤（概念）
    - ◆ 确定所采用的原理与策略
    - ◆ 规划视图集成步骤
    - ◆ 发现并解决可能存在的冲突现象
  - ■ 视图集成冲突的解决办法：视图修改（概念）
  - ■ E-R 模型与 EE-R 模型的设计（应用）
- ● 数据库的逻辑设计
  - ■ E-R 模型和 EE-R 模型向关系模型的转换（应用）

第十章　数据库管理（概念）

- ● 数据库管理概述：
  - ■ 数据库管理：数据库设计完成后，经过数据库建立、数据库加载之后投入实际运行，并在运行的过程中进行监控和维护的这些管理维护工作
  - ■ 数据库管理员：实施管理工作的人
- ● 数据库管理的内容：
  - ■ 数据库的建立（数据库运行环境的设置、数据模式的建立、会话环境的建立、数据加载）
  - ■ 数据库调整（例如逆规范化、关系分割、快照、调整索引、分区等）
  - ■ 数据库重组（对数据库进行重新整理，重新调整存储空间）
  - ■ 数据库重构（重构数据库模式）
  - ■ 数据库安全性控制
  - ■ 数据库完整性控制

- 数据库的并发控制（性能调整、死锁、活锁解除等）
- 数据库的故障恢复
- 数据库的监控

- 数据库管理员 DBA
  - 核心人物，若干人员组成、有最高级别特权、还需要完成相关行政管理和参与数据库设计的部分工作

记的都是加红的
数据库期末复习——很经常考

- 第一章
  - 数据：具有一定格式或结构的符号串，属于软件范畴
  - 数据库（DB）
    - 是数据集合，具有统一结构形式并存放于统一存储介质内
    - 也就是：结构化、集中存储、数据共享
  - 数据库管理系统（DBMS）：
    - 管理数据库的系统软件
    - 文件管理系统发展而来
  - 数据库系统（DBS）五个组成部分：数据库、数据库管理系统、数据库管理员、软件平台、硬件平台
  - 数据库四个特点：数据的集成性、数据的高共享性和低冗余性（数据的一致性和不一致性）、数据独立性、数据的统一管理与控制（数据的完整性检查、安全性保护、并发控制、数据库故障恢复）
  - 数据库内部结构体系：
    - 三级模式：外模式、概念模式、内模式
    - 两种映射：外模式-概念模式映射、概念模式-内模式映射
- 第二章
  - 数据模型：
    - 数据模型的组成：数据结构、数据操作、数据约束
    - 数据模型的核心：数据结构
  - 数据模型表示每一步转化结果
    - 概念数据模型（概念模型）
    - 逻辑数据模型（数据模型）
    - 物理数据模型（物理模型）
  - 三种数据模型和三级模式之间的对应关系？
    - 逻辑模型——外模式和概念模式
    - 物理模型——内模式
    - 注：概念模式不关系到数据库的实现，只到 E-R 图
  - 四个世界：
    - 现实世界：
    - 概念世界：概念模型表示
      - E-R 模型（掌握）、EE-R 模型（掌握，继承、弱实体）、面向对象模型、谓词模型
      - 会画 E-R 图、自联结、弱实体、继承怎么画

- ◆
- 关系
  - ◆ 行、列、基数、元数
  - ◆ 关系的五个约束
    - 同一表中的属性名不同
    - 表中属性与属性顺序无关
    - 表中元组均不相同
    - 表中的元组与元组的排列次序无关
    - 表中每一分量必须是一个不可分割的基本数据项
- 第三章
  - 关系上五种操作和关系代数的五种基本运算：注意交运算不是基本运算
    - ◆ 元组选择——选择运算
    - ◆ 属性指定——投影运算
    - ◆ 关系的合并——笛卡尔乘积
    - ◆ 元组的插入——并运算
    - ◆ 元组的删除——差运算
  - 投影运算和选择运算不能互换
  - 关系的笛卡尔乘积
  - 三种扩充运算：交、除、自然连接
  - 关系代数实例
    - ◆ “至少”型：例如检索至少修读为 S5 所修读的一门课的学生姓名
      - 结果元组满足的条件？
      - S5 修读哪些课？
      - 如何从 S5 修读过的课程中找出符合条件的学生姓名？
    - ◆ “所有”型（除法）：例如检索修读 S4 所修读的所有课程的学生姓名
      - 结果元组满足的条件?
      - S4 修读了哪些课程？
      - 如何根据一组课程查出修读了其中的所有课程的学生元组？
      - 注意除法的被除数一定不能有多余属性
    - ◆ 最大型？大于？最小？……
  - SQL 数据定义功能
    - ◆ 基表创建命令 CREATE TABLE
    - ◆ 基表修改命令 ALTER TABLE
    - ◆ 基表删除命令 DROP TABLE
  - SQL 数据操纵功能
    - ◆ 映像语句
      - 目标子句 SELECT
      - 范围子句
      - 条件子句
      - 分组子句
        - BY 不要漏了
      - 分组查询子句
      - 排序输出子句 ASC|DESC（老师上课强调了原来考试考这个）
        - BY 不要漏了

        - 这些子句的执行顺序？你会了吗，老师上课也强调了
    - 常用谓词
        - [NOT] LIKE
        - IN
        - ANY|ALL
        - [NOT] EXISTS
        - UNION, INTERSECT, EXCEPT
            - 会自动去除冗余元素，如果要不去除，就要在后面加上 ALL 关键字
    - “至少”型：例如查询至少修读学号为 S5 的学生所修读的一门课程的学生的学号
        - 连接
    - 空值？完全掌握了吗？
        - 对集合元素统计：忽略空值
        - 在空集上统计：COUNT 返回 0 其余返回空
    - 分类/分组统计查询？ GROUP BY
        - 常见错误：SELECT 后面的非聚集函数属性一定要和 GROUP BY 相同！老师之前和现在上课都强调了
    - SQL 的更新功能
        - 删除功能 DELETE FROM
        - 元组插入 INSERT INTO 的整条语句，有好多内容？顺序？确定会了吗
        - 修改功能 UPDATE
        - 视图 VIEW：
            - 视图和基表的区别？“大家一定要注意！！”
                - 并不实际存在，仅仅保留了构造信息
                - 用户执行视图上的访问操作->数据库系统转换成基表上的查询语句
            - 视图定义 CREATE VIEW
            - WITH CHECK OPTION 是干嘛的？
- 第四章
    - 数据库的安全
        - 要求数据库用户：规定的访问路径、规定的访问规则、接受 DBMS 检查
        - 数据库安全有关的实体：主体+客体，主体单向访问客体
        - SQL 提供 C1 级别的数据库安全支持
            - 主体、客体及主客体分离
            - 身份标识与鉴别
            - 数据完整性（引用完整性、参照完整性……）
            - 【还有一点】
        - 授权语句（以前考过）：GRANT … ON … TO … [WITH GRANT OPTION]
        - 回收语句：REVOKE … ON … FROM … [RESTRICT 不存在连锁回收问题时回收|CASCADE 连锁回收]
    - 数据库的完整性
        - 指数据库中数据的正确性和一致性
            - 正确性：数据的有效性、有意义
            - 一致性：多用户并发访问数据库的情况下，保证数据库更新不会不一致

        - 三类数据完整性约束
            - 实体完整性：基表的主关键字中属性取值不能为空
            - 参照完整性：例如上级领导
                - 关系 S 上引用外关键字 F 上的值或者是空值，或者是 S 上有的值（看 PPT 确定一下）
            - 用户定义的完整性（例如触发器，考试不做要求）
- 第五章
    - 事务：不能被打断的对数据库的操作序列
        - 逻辑工作单位，不要搞错
        - 按照预定顺序执行
        - 串行
    - 事务的四个特性（ACID）
        - 原子性 A
        - 一致性 C
        - 隔离性 I
        - 持久性 D
        - 例如问：满足了事务的什么特性？要把具体什么特性写出来
    - 三种并发执行错误：会判断了吗
        - 丢失修改
        - 脏读
        - 不可重复读
    - 数据访问操作
        - INPUT, OUTPUT, READ, WRITE
    - 事务的并发执行
        - 调度：操作序列
        - 串行调度：内部不存在并发
    - 冲突：一对连续操作，if 交换顺序 then 涉及的事务中至少有一个行为会改变
        - 判断冲突会了吗？
    - 冲突等价：一个调度能够相邻交换变成另一个调度
    - 冲突可串行化
        - 怎么判断冲突可串行化？
        - 可串行化的充分条件，不是必要条件
        - 怎么转化成冲突可串行化？
    - 封锁
        - 排它锁：读写操作
        - 共享锁：读操作
        - 锁相容矩阵
    - 三级封锁协议
        - 一级封锁协议？
        - 二级封锁协议？基于一级
        - 三级封锁协议？注意不基于二级，而是基于一级
        - 三级封锁协议和可能出现的数据不一致现象之间的关系？
    - 两阶段封锁协议
        - 申请阶段

        - 释放阶段：事务一旦开始释放锁，就不能申请任何锁
    - 两阶段封锁事务（2PL 事务）？定义
    - 多粒度封锁？——稍微了解
        - 意向锁
            - 意向共享锁
            - 意向排它锁
            - 共享意向排它锁
    - 故障恢复——这里我好像不是很熟悉
        - UNDO 日志：开始、提交、放弃、更新，注意 V 记录的是旧值
            - UNDO 日志的记载规则?
        - REDO 日志：开始、提交、放弃、更新，注意 V 记录的是更新后的值
        - UNDO/REDO 日志：格式与上一样，区别在于更新记录的格式<T,X,v,w>
            - 记载规则?
            - 既可重做也可【待补充】
- 第六章
    - 游标管理：数据库 SQL 变量是集合型，程设语言是标量型，游标一个一个访问集合元素
    - 动态 SQL：根据程序的实际使用情况生成并调用 SQL 语句
    - 静态 SQL 和动态 SQL 的比较
        - 静态
            - 优点：性能好
            - 缺点：只能根据缺省参数值优化，访问路径并非最优
        - 动态
            - 优点：可以根据运行时数据库的最新情况选择最优访问路径
            - 缺点：
    - 什么情况下需要使用动态 SQL？（有三个情况）
        - 【待补充】
- 第七章
    - 记录在磁盘中的分配方式
        - 单块单记录、单块多记录
            - 特点：记录不能跨块，存储空间浪费
        - 多块单记录、多块多记录
            - 允许记录跨块，但多块单记录仍然存在空间浪费现象，多块多记录则无
    - 索引技术与散列技术
        - 稠密索引
            - 键-指针对，索引项
            - 稠密索引：数据文件的每条记录在索引文件中都有对应的索引项
        - 稀疏索引
            - 只为文件的每一个磁盘块设立一个索引项
        - 稠密索引和稀疏索引的比较?
            - 索引文件定义不同
                - 稀疏索引只能用于
                - 【待补充】
            - 需要的磁盘空间大小不同

- 在记录的查找定位功能上存在差别
  - 【待补充】
- 性能比较如何计算？需要的磁盘空间（数据文件+索引文件）？记录查找需要的磁盘 I/O 次数？

- B/B+树
  - 叶结点
  - 内部结点
- B+树的四个特点
  - 平衡性、过半性、自适应性、【待补充】
- B 树和 B+树的六个比较
  - 【待补充】
- 散列技术

- 第八章
  - 关系的规范化
    - 函数依赖&多值依赖
    - 规范化、范式
  - 函数依赖
    - 函数依赖：完全、部分、平凡、非平凡、直接、传递？
    - Armstrong 公理系统——了解即可
      - 基本规则
        - 自反规则、增广规则、传递规则
      - 扩充规则
        - 分解规则、合并规则、伪传递规则
      - 逻辑蕴涵
      - 属性依赖集的闭包？
    - 键/码（完全函数依赖）
    - 主属性（**所有关键字**）、非主属性？主属性集、非主属性集？
    - 两个算法
      - 属性集闭包算法？
      - 关键字算法？
  - 与函数依赖有关的范式 X->Y
    - 1NF：不可分
    - 2NF：非主属性完全函数依赖于关键字
    - 3NF：每个非主属性都不传递函数依赖于关键字
      - X 不是关键字或是关键字的一个真子集
    - BCNF：1NF+X 必然包含该关系的关键字
    - 属于 BCNF 一定是 3NF，反之不成立
  - 函数依赖集的等价：双向是否可以逻辑蕴涵
  - 最小函数依赖集/最小覆盖：判断冗余、判断部分函数依赖，就是最小的
  - 无损连接性、依赖保持性？定义？
    - 不满足无损连接性的情况？
    - 无损连接的充分必要条件？（老师说这个大家要记得！）
    - 依赖保持性？
- 第九章

- 数据库设计的四个阶段
  - 需求分析、概念设计、逻辑设计、物理设计
- 联系的转换
  - 联系转换成关系或者关系模式？关键字是什么？每个关系有什么属性？
  - 全参与、非全参与？
  - 1：1
    - 均非全？
    - 一个非全一个全？
    - 均全？
  - 1：n
    - 均全？
    - 多端非全？
  - n：m
    - 总是转换成三个，注意 R 关系中是联合主键
- 关系数据库管理系统 DBMS，为了提高数据库的性能
  - 逆规范化
  - 关系分割
  - 尽量使用快照

- 第十章
  - 数据库管理员 DBA
  - 缓冲池对性能的影响
    - 缓冲池是【待补充】
    - 从内存中访问数据的数据比磁盘访问的速度快