**数据**是数据库中存储的基本对象，描述事物的符号记录 **数据库**是长期储存在计算机内、有组织的、可共享的大量数据集合 **特征**数据按一定的数据模型组织、描述和储存可为各种用户共享、冗余度较小、数据独立性较高、易扩展 **数据库系统DBS**由数据库、数据库管理系统、应用程序、数据库管理员和用户组成**目的**是为用户提供数据的抽象视图 **数据库管理系统DBMS**是介于用户与操作系统之间数据管理软件系统。为数据库的建立、运行和维护提供统一管理和控制，提供定义数据和操纵数据接口，保证数据的安全性、完整性、并发使用及发生故障后的系统恢复 **数据库管理员(DBA)**对系统进行集中控制的人，包括数据库设计、规划、协调,最高特权的用户**作用**模式定义/存储结构及存储方法定义/模式及物理组织的修改/数据访问授权/日常维护) **管理信息系统（MIS）**办公系统、决策系统、生产系统和信息系统面向数量的执行系统、面向价值的核算系统、报告监控系统，分析信息系统、规划决策系统 **数据管理发展的三个阶段特点：人工管理阶段**没有相应的软件系统，数据不共享，数据不具独立性。计算机主要用于科学计算外存为顺序存取设备没有操作系统，没有数据管理软件 **文件系统阶段**系统提供一定的数据管理功能如索引文件、链接文件、增删文件等弊端（数据冗余不一致、数据检索困难、数据不共享、完整性问题、原子性问题、并发访问一场、数据安全问题） **数据库系统阶段**DBMS统一管理，数据共享，数据的冗余度小，易扩充；数据面向整个系统，反映了客观事物间的本质联系，而不是着眼于面向某个应用，是有结构的数据；数据安全性，一致性，具有较高的数据和程序的独立性把数据库的定义和描述从应用程序中分离出去 数据描述是分级的（全局逻辑、局部逻辑、存储） 数据的存取由系统管理，用户不必考虑存取路径等细节，从而简化了应用程序  **DBMS演变**传统(网状层次关系)新型 **DBMS特点**用户接口，非过程数据库语言/查询处理与优化策略/并发控制，解决冲突/恢复功能，数据一致性状态/完整性约束检查，实体完整性、引用完整性、域完整性、用户定义完整性/访问控制，访问权限，安全性问题 **DBMS组成1数据定义语言(DDL)**是说明数据库模式和数据的其他特性的语言，提供了定义关系模式、删除关系和修改关系模式的命令。**2数据操纵语言(DML)**是使得用户可以访问和操纵数据的语言，包括查询语言以及往数据库中插入元组、删除元组和修改数据库中元组的命令**3数据库运行控制程序(DCL)**初始化安全性控制完整性检查并发控制事务管理运行日志管理等/实用程序， 数据转储恢复程序性能监测数据库重组数据转换通信等 **DBMS功能部件**查询处理器（DML编译器DDL解释器查询求值引擎）存储管理器（权限及完整性管理器事物管理器文件管理器缓冲区管理器数据字典数据文件索引) **数据模型**是一个描述数据、数据联系、数据语义以及一致性约束的概念工具的集合**（**关系ER基于对象的半结构化数据模型**）概念数据模型**现实世界的数据抽象，描述一个单位的概念化结构,与DBMS无关，如ER面向对象**逻辑数据模型**DBMS层面上的数据形式。如层次网状关系**物理数据模型**数据存储结构，如物理块指针索引方式 **数据模式**数据模型下对数据的逻辑结构的描述 **模式**是指它的逻辑设计，**实例**是指它在特定时刻的内容 **关系模式的非形式化标准**：1关系模式的设计应尽可能只包含有直接联系的属性，不要包含有间接联系的属性。2 关系模式的设计应尽可能使得相应关系中不出现插入、删除和修改等操作的异常。如果出现任何异常，则要清楚的加以说明。并确保更新数据库的程序正确操作。3 关系模式的设计应尽可能使得相应关系中避免放置经常为空的属性。4 关系模式的设计应尽可能的使得关系的等值连接在主键和外键的属性上进行，并且保证连接以后不会生出额外的元组 **数据视图**数据库给用户提供数据的抽象视图**物理层（内模式）**磁盘存储方式中物理结构的描述，是DBMS设计人员看到的数据形式**逻辑层（模式或数据模式）**数据库全体数据的逻辑结构和特征的描述。DBA看到 **视图层（外模式子模式）**数据库部分数据的逻辑结构和特征的描述,用户看到,是模式的一个局部 **简述数据库系统三级模式结构及其同数据独立性之间的联系**：为了提高数据的物理独立性和逻辑独立性，使数据库的用户观点，即用户看到的数据库，与数据库的物理方面，即实际存储数据库区分开来，数据库系统模式是分级的，美国数据系统语言协会提出模式、外模式、存储模式三级模式的概念。三级模式之间有两级映像：存储结构改变时，修改模式/内模式映像，使模式保持不变，从而应用程序可以保持不变，成为数据的物理独立性；当模式改变时，修改外模式/模式映像，使外模式保持不变，从而应用程序可以保持不变，称为数据的逻辑独立性 **映射程序**外模式/模式映象：外模式和模式之间的对应关系，映象定义通常包含在各外模式中。模式/内模式映象：数据逻辑结构与存储结构之间的对应关系 **数据字典**保存模式，外模式和内模式的定义 **数据独立性**应用程序与DB数据结构之间相互独立**包括物理独立性**数据的物理结构的改变，修改模式/内模式映象，不影响数据库的逻辑结构，从而不致引起应用程序的变化**逻辑独立性**数据库逻辑结构的改变，修改外模式/模式映象，不需要修改相应的应用程序 **完整性约束（作用：**保证授权用户对数据库的修改不会破坏数据的一致性）**1．域完整性约束**属性值应是域中的值，属性的值能否为null，由语义决定。在确立关系模式时规定的，由DBMS负责检查 **2．实体完整性约束**每个关系应有一个主码，主码的值不能为null**3．引用完整性约束**不同关系之间或同一关系的不同元组间的约束。若关系R中有一个外码（相对于关系S），则R中每个元组的外码的值必须满足：(1) 或者取空值 (2) 或者等于S中某个元组的主码值。R与S可以是同一个关系**4．用户定义的完整性约束**用户定义的完整性约束是针对某一具体数据库的约束条件，由具体应用要求决定。 不是数据模型的一部分，但作为虚关系对用户可见的关系称为**视图 特点：**虚表，是从一个或几个基本表或视图中导出的关系，只存放视图的定义，不会出现数据冗余。基表中数据发生变化，从视图中查询的数据也发生变化 **视图定义**create view DEPTSAL( DNO, LOW, HIGH, AVERAGE, TOTAL ) as ( select DNO, min(SAL), max(SAL), avg(SAL), sum(SAL) from PROF group by DNO ) **视图可更新**from子句中只有一个数据库关系 select子句中只包含关系的属性名，不包含任何表达式，聚集或distinct声明 任何没有出现在select子句重的属性可以取空值，没有notnull约束也不是主码 查询中无groupby 或having子句 **在视图可更新情况下仍会出问题**，则在视图定义的末尾加上with check option子句，数据库系统拒绝不满足视图where子句的元组插入或更新 **断言**CREATE ASSERTION <断言名> CHECK <条件>断言是谓词，表达数据库总应该满足的一个条件 一旦定义了断言，系统验证其有效性，并且对每个可能违反该断言的更新操作都进行检查 撤消断言：drop assertion 断言名 **游标**在查询结果的记录集合中移动的指针 用于把集合操作转换成单记录处理方式。若一个SQL语句返回单个元组，则不用游标 若一个SQL语句返回多个元组，则使用游标 **触发器**是一条语句，当对数据库做修改时，它自动被系统执行 可以用来实现为被SQL约束机制指定的某些完整性约束 **JDBC**标准定义了java程序连接数据库服务器的应用程序接口 **ODBC（开放数据库互连）**标准定义了了一个API，应用程序用它来打开一个数据库连接，发送查询和更新 **联机分析处理系统（OLAP）**是一个交互式系统，它允许分析人员查看多维数据的不同种类的汇总数据**实体-联系模型ER**现实世界是由实体和实体之间的联系构成的，用于数据库设计，提供了一个方便的图形化表示方法以查看数据、联系和约束。实体客观存在并可相互区分的事物叫**实体**（唯一标识）**实体标识符**能惟一标识实体的属性或属性集，有时也称为**关键码**，或简称为**键**。**实体集的属性**是将实体集映射到域的函数。**联系**是多个实体间的关联。参与联系的实体组成的集合为**联系集**  **弱实体集**不具有足够属性构成主码的实体集。有主码的实体集称为强实体集。弱实体集必然存在依赖于强实体集，弱实体集与强实体集之间是一对多的联系弱实体集与其拥有者之间的联系称作**标识性联系。分辨码是**弱实体集中用于区别依赖于某个特定强实体集的属性集合，也称作**部分码。 特化概化**定义了一个高层实体集合一个或多个底层实体集之间的包含关系。**特化**是取出高层实体集的一个子集来形成一个低层实体集。**概化**是用两个或多个不相交的（低层）实体集的并集形成的一个高层实体集。两者互逆，都用标记为ISA的三角形来表示。高层实体集的属性被低层实体集自动继承（层次结构/格结构）**聚集**是一种抽象，其中联系集（和跟他们相关的实体集一起）被看作高层实体集并且可以参与联系。 **使用弱实体集的原因：**避免数据冗余（强实体集码重复）以及因此带来的数据不一致性。反映了一个实体对其他实体依赖的逻辑结构 。可以随他们强实体的集的删除而自动删除 **无关属性：**如果去除一个函数依赖中德属性，不会改变该函数依赖集的闭包，则称该属性是无关的 **范式**是关系模式的集合。某个域的元素被认为是不可再分的单元那么这个域就是**原子的**，一个关系模式R的所有的属性域都是原子的称关系模式R属于**第一范式1NF**（所有属性都是不可分的基本数据项） X不包含Y，**非平凡的函数依赖**。 若关系模式Ｒ∈1NF并且每个非主属性都完全函数依赖于R的码**则R∈2NF** 设 X 属于U ，则属性集X关于函数依赖集F的**属性闭包**为X+F =｛A｜A∈U且X→A能由F根据推理规则推出｝ **FC满足下列条件称为正则覆盖**（极小函数依赖集）1.FC中每个函数依赖的右部为单一属性2.FC中不存在函数依赖X→A，使得FC -｛X→A｝与FC等价3.在FC中不存在X→A，使得FC -{X→A}∪{Z→A}与FC等价，其中XつZ**求正则覆盖1.**根据分解规则，把函数依赖右端变成单属性2.对F1中各依赖X→A，令G=F1-{X→A}，若A∈X+G ，将X→A从F1中去掉3.对F2中左端是多属性的函数依赖X→A, 设X=B1,B2,…,Bm，若A∈(X-Bi)+F2 ，则将Bi从X中去掉 关系模式R=(U)的每个非主属性都不部分依赖也不传递依赖于码，则称R满足**第三范式**，记为R∈3NF。如果关系模式R=(U)的所有非平凡函数依赖α→β， α部分都包含R的一个候选码，则**称R属于BC范式，记作R∈BCNF说明：**1.α→β是平凡依赖时是允许的2.α是超码，决定因素是超码的依赖允许存在。决定因素不是超码不允许3.**3NF与BCNF的差别**：3NF 要消除的是非主属性对码的部分依赖和传递依赖，而BCNF要消除的是主属性对码的部分依赖或传递依赖。如果R∈3NF，检查所有决定因素是否都包含候选码，若是，R∈BCNFG 4.BCNF实现了模式的彻底分解，达到了最高的规范化程度。**无损连接分解的判断**设ρ={R1=（U1）,R2=（U2）,…,Rk=（Uk）}是R的一分解，对于R的任一个关系r满足 r=ΠU1(r) ∞ ΠU2(r) ∞ … ∞ Π Uk(r) 设ρ是R的一个分解，F是R的函数依赖集，如果G= ΠR1(F)∪ΠR2(F)∪…Π Rk(F)逻辑蕴涵F，则称ρ是保**持函数依赖分解**，**3NF的模式分解算法1.3NF且保持函数依赖性的分解算法**1.求Fc 2.如果Fc中有一函数依赖X→A,且XA=U，则ρ={R},输出ρ，结束。3.对Fc中的每一个X→Ai都构成一个关系模式Ri=(X , Ai)。如果Fc中有X→A1,X→A2,…,X→An(左部决定因素相同)则以R=(X,A1,A2,…,An)构成一个模式**2.求分解关系模式为3NF且保持函数依赖性和无损连接性的分解**1按保持依赖的3NF分解算法求出ρ={R1,R2,…,Rk}2设X是R的一个码，如果某个Ui包含了X，则τ=ρ，否则，令 Rk+1=(X)，τ=ρ ∪{Rk+1(X)}**BCNF无损连接分解算法1.**初始化ρ={R}**2**.如果ρ中所有关系模式均已是BCNF转(4)，否则，从ρ中找出非BCNF的关系模式S，在S中必有X→A, X不包含S的码，A也不在X中。将S分解为S1=(X，A)，S2=(Us-A)，Us为S的属性组**(3)**在ρ中以{S1,S2}代替S，转(2)**（4)**结束，输出ρ**平凡多值依赖 多值依赖与函数依赖区别** 函数依赖规定某些元组不能出现在关系中，也称为相等产生依赖 多值依赖要求某种形式的其它元组必须在关系中，称为元组产生依赖 对于X→→Y，如果Y是X的子集或X∪Y=U，称X→→Y成立，称为平凡多值依赖 设R=（U），D是R上的函数依赖和多值依赖集，如果D的每个X→Y, X→→Y ，X必是R的超码，那么R属于**4NF 4NF的无损连接分解算法**1.初始化ρ={R} 2.如果ρ中所有关系模式均已是4NF转4 3.从ρ中找出非4NF的关系模式S，在S中必有X→→Y, X不包含S的码，Y也不在X中。将S分解为S1=(X，Y)，S2=(Us-Y)，Us为S的属性组。在ρ中以{S1,S2}代替S，转2 4结束，输出ρ **事务**访问并可能更新数据的逻辑工作单位，一组操作序列，一个或一段应用程序。并发控制以事务为单位进行的 。 **事物是**构成单一逻辑工作单元的操作集合，要么完整的执行，要么全不执行**2.空值NULL：**表示“无意义”，当实体集在某个属性上没有值时设为null；或者表示“值未知”即值存在，但目前没有获得该信息；当空值参与运算，结果为空值。**特性**事务的属性)**ACID特性**：原子性**（保证**事务的所有影响在数据库中要么全反映要么根本不反映**）**一致性**（保证**事务必须是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**（保证**并发执行的事务相互隔离.一个事务内部的操作及使用的数据对并发的其它事务是隔离的**）**持久性**（保证**是指一个事务一旦提交，它对数据库中数据的改变就应该是永久性的**）**。  **“活锁”问题**：系统可能使某个事务永远处于等待状态，得不到封锁的机会。解决方法：采用“先来先服务”的策略，也就是简单的排队方式。如果运行时，事务有优先级，那么很可能使优先级低的事务，即使排队也很难轮的上封锁的机会。此时可以采取“升级”方法来解决，也就是当一个事务等待若干时间还轮不上封锁时，可以提高其优先级别，这样总轮的上封锁。**“饿死”问题**:有可能存在一个事务序列，其中每个事务都申请对某个数据加s锁，且每个事务在授权加锁后一小段时间释放封锁，此时若另有一个事务T2欲在该数据项上加x锁，则将永远轮不上封锁的机会，这种现象称为“饿死”。可以用下列方式授权加锁来避免事务饿死。当事务T2中请对数据项Q加S锁时，授权加锁的条件：1 不存在数据项Q上持有X锁的其他服务；2 不存在等待对数据项Q加锁且先于T2申请加锁的事务。**死锁问题**：系统中有两个或两个以上的事务都处于等待状态，并且每个事务都在等待其中另一个事务解除封锁，他才能继续执行下去，结果造成任何一个事务都无法继续执行，这种现象称系统进入死锁状态。当死锁发生时，系统必须回滚两个事务中的一个。一旦某个事物回滚，该事务锁住的数据项就被解除了，其他事物就可以访问这些数据项，继续自己执行。 **封锁协议**事务运用X锁和S锁对数据对象加锁时的约定**X锁排它型锁**又称为写锁若事务T对数据对象Q加上X锁，则只允许T读取和修改Q，直到T释放Q上的锁，其他任何事务都不能对Q加上任何类型的锁，提出加锁，只能等待 **S锁共享型锁**又称为读锁若事务T对数据对象Q加上S锁，其它事务还能再对Q加上S锁，但加不上X锁。**一级封锁协议**事务T在修改数据Q之前必须先对其加X锁，直到事务结束才释放。事务只读数据时，协议不要求加锁.一级封锁协议可防止丢失修改。但不能保证可重复读和不读“脏”数据。 **二级封锁协议**事务T在修改数据Q之前必须先对其加X锁，直到事务结束才释放。事务T在读取数据Q之前必须先对其加S锁，读完后即释放S锁。二级封锁协议可防止丢失修改，防止读“脏”数据，但不能保证可重复读。 **三级封锁协议**事务在修改数据Q之前必须先对其加X锁，直到事务结束才释放。事务在读取数据Q之前必须先对其加S锁，直到事务结束才释放S锁。三级封锁协议可防止丢失修改，防止读“脏”数据，保证可重复读。 **两阶段封锁协议**对任何数据读写之前，先要获得对该数据的封锁，释放一个封锁之后，事务不再加任何锁。两段：增长阶段，申请加锁阶段；缩减阶段，释放锁阶段。事务开始为增长阶段，一旦释放锁，进入缩减阶段。所有事务都遵守两阶段协议，并发调度是可串性化的两阶段封锁协议可能发生死锁**时间戳**事务启动时间记为 TS（Ti）**W-TS（Q）**所有执行了写Q操作的事务中最年轻事务的时间戳（值最大）。**R-TS（Q）**所有执行了读Q操作的事务中最年轻事务的时间戳（值最大）。**时间戳协议**Ti 读数据Q时（1）若TS（Ti）< W-TS（Q），撤消Ti并重新启动（2）若TS（Ti）> W-TS（Q），执行读操作，R-TS（Q）= MAX（R-TS（Q）,TS（Ti ））Ti 写数据Q时（1）若TS（Ti）＜ R-TS（Q），撤消Ti并重新启动（2thomas 写规则）若TS（Ti）＜ W-TS（Q），撤消Ti并重新启动。若TS（Ti） ＜ W-TS（Q），撤消Ti并重新启动（3） 其它情况，执行写操作，W-TS（Q）=TS（Ti）**E—R模型定义**现实世界的事物抽象为实体，数据粒度相当于记录，描述实体用属性，数据粒度相当于字段，实体之间用联系表示。用E—R图表示，局部E—R图，全局E—R图。**面向对象模型定义**现实世界的事物抽象为对象，用类来定义，对象的特征用属性，对象之间用类与子类表示，对象的处理用封装其中的方法，调用方法用消息。**层次模型定义**实体用结点表示，实体之间的联系用树表示**网状模型定义**实体由记录表示，记录间联系由系（set）表示，系是一个指针关系模型**关系模型定义**实体由元组（记录）表示 实体集由关系（表）表示 实体之间的联系也通过关系表示。 **DBMS演变**传统(网状层次关系)新型（并行数据库，知识库系统，主动数据库，多媒体数据库、CAD数据库、图形数据库、图像数据库、智能数据库，面向对象数据库，数据仓库，工程数据库，空间数据库。） **关系**二维表，由关系名标识 **元组**表中的一行，称之为n元组 **记录**属性表中的一列，属性名标识 **码**能够唯一确定一个元组的最小属性集 **全码**表中找不出码，所有属性组成**主属性**任一候选码中的属性**非主属性**不在任何一个候选码中 **关系的性质**每列同类型,同域,不同的列可以同域/列的顺序无关，交换列的次序,仍是同一个表/行的顺序无关，交换行的次序,仍是同一个表/任意两行不能全同 , 码的本质/任意两列的列名不能相同, 属性区分/每一分量是不可分的数据项, 关系数据库本质 **模式图**是数据库中模式的图形化表示，显示了数据库中的关系，关系的属性主码和外码 **超码**是一个或多个属性的集合，这些属性的集合可以使我们在一个关系中唯一地标识一个元组 **候选码**是最小的超码，是一组构成超码的属性集，但任意子集都不是超码。关系一个候选码选作**主码。外码**对于参照关系中每个元组来说，它在外码属性上的取值肯定等于被参照关系中的某个元组在主码上的取值 **关系查询语言**定义了一组运算集，这些运算可作用于表上并输出表作为结果；可以组合成表达式表达所需的查询。 **关系代数**一种抽象的查询语言，基于集合论，是DML的传统表达方式定义了一套在表上运算且输出结果也是表的代数运算。关系代数定义了关系查询语言中使用的**基本运算（**选择投影并集合差笛卡尔积**）元组关系演算**和**域关系演算（**使用从属性域中取得的域变量而不是整个元组的值**）**是**非过程化语言（**用户只需描述信息而不用给出获取该信息的具体过程**）**，代表了关系查询语言所需的基本能力。 **关系查询语言（**集合运算：并、交、差、广义笛卡尔积。专用运算：选择、投影、连接、除、更名、赋值元组关系演算和域关系演算）（基于谓词逻辑表示表的操作**） 实体集**是具有相同类型及共享相同性质（属性）的实体集合。实体名和属性名组成**实体型**对关系模型就是关系模式 **属性**实体特征，通过其值区分不同的实体 **域**属性的取值范围 **域**P的域是P所引用的所有值的集合，既包括P自身用到的值又包括P中涉及关系的元组中出现的所有值 **属性分类**：简单属性，复合属性；单值，多值；**派生属性**从其他相关的属性或实体派生出来的属性值被派生的叫基属性或存储属性 **多值属性**用双椭圆表示**派生属性**用虚椭圆表示 主码下划线表示 **实体集的属性**是将实体集映射到域的函数 **映射的基数**表示通过联系集可以和另一实体相关联的实体的个数 **联系的参与度**每个实体有一个参与联系的次数，取最小、最大的参与次数min（部分参与，全参与）和max（最大参与次数）定义实体集的参与度 **角色**实体在联系中的作用称为实体的角色（同一实体集不止一次参与一个联系集） 如果实体x的存在依赖于实体y的存在，则称x**存在依赖**于y y称作支配实体，x称作从属实体 如果y被删除，则x也要被删除 设A R B，若A存在依赖于B，则A**全部参与**联系R 分辨码和强实体集主码构成主码 ER**图中的表示：** 弱实体集以双边框的矩形表示 标识性联系以双边框的菱形表示 从联系集用双线（全部参与）连接弱实体集 用箭头（一对多联系）指向强实体集 弱实体集的分辨符用下划虚线标明  **函数依赖定义**设R=(U)表示一个关系模式，属性集合U={A1,A2,…,An}，X和Y是U的子集，对R=(U)下的任一个关系r的任意两个元组t1，t2，如果t1［X］=t2［X］，则t1［Y］=t2［Y］，称属性集X与属性集Y存在函数依赖。记作X→Y X的真子集仍决定Y则Y对X**部分函数依赖**否则完全函数依赖 R下的所有函数依赖组成**R的函数依赖集，记为F** 设F是关系模式R的一个函数依赖集，X、Y是R的属性子集，如果从F中的函数依赖能够推出X→Y，则称F**逻辑蕴涵**X→Y 关系模式R=(U)中，F及F所蕴涵的全体函数依赖称为F的闭包，记为F+。如果F=F+，则称F是**函数依赖的完备集** 设R=(U)的两个函数依赖集F和G，如果F+=G+，则称F和G是等价的称F覆盖G且G覆盖F  **F和G是否等价的方法**检查F中的每个函数依赖X→Y，检查Y是否属于X+G。检查G中的每个函数依赖X→Y，检查Y是否属于X+F。**3 NF的判断1.** 确定R的候选码，主属性，非主属性，决定因素(2) 如果R的每个属性都是基本数据项，则R∈1NF 3) 如果R∈1NF,检查所有非主属性是否对所有候选码都不存在部分函数依赖。若是，R∈2NF。当候选码都是单一属性时，R必然属于2NF(4) 如果R∈2NF，检查所有非主属性是否对所有候选码都不存在传递函数依赖，若是，R∈3NF。当R没有非主属性时，**R必属于3NF**。 **多值依赖的定义**设R=（U），X，Y，Z是U的子集，Z=U-X-Y，对于R的任一r, 如果有（x,y1,z1）和(x,y2,z2)属于r, 则有（x, y1,z2）和（x, y2,z1）属于r, 那么称Y多值依赖X，Z多值依赖X。记为X→→Y，X→→Z。 **多值增补率**：若X(双箭头）Y且W属于Z，则XZ双箭头YW 多值传递率：若X双箭头Y且Y双箭头Z，则X双箭头Z – Y 复制率：若X单箭头Y，则X双箭头Y 联合率：若X双箭头Y，Z属于Y，W∩Y=空且W单箭头Z，则X单箭头Z 多值合并率：若X双箭头Y且X双箭头Z ，则X双箭头YZ取交率：若X双箭头Y且X双箭头Z ，则X双箭头Y∩Z取差率：若X双箭头Y且X双箭头Z ，则X双箭头Y – Z， X双箭头Z –Y **完整性约束与SQL1.在属性值上的约束1.**非空约束：要求某属性取值不能为空值SNAME CHAR(8) NOT NULL2.基于属性的检查子句CHECK(AGE>15)3.域约束子句 用CREATE DOMAIN定义域时CREATE DOMAIN AGE SMALLINTCHECK((VALUE >= 15) AND VALUE <= 25))**2.主码约束**主码定义形式 主码子句：PRIMARY KEY(Sno) 主码短语：Sno CHAR(4) PRIMARY KEY **3.外码约束（参照完整性）**在SC表中，定义FOREIGN KEY (Sno) REFERENCES S(Sno) Sno在SC中是外码，在S中是主码，Sno在SC中取值或为空或为S中某个Sno值 作为主码的关系称为基本（参照）关系，作为外码的关系称为依赖关系**删除基本关系元组1.**RESTRICT方式只有当依赖关系中没有一个外码值与要删除的基本关系的主码值相对应时，才可以删除该元组，否则系统拒绝此删除操作2.CASCADE方式将依赖关系中所有外码值与基本关系中要删除的主码值所对应的元组一起删除3.SET NULL方式删除基本关系中元组时，将依赖关系中与基本关系中被删主码值相对应的外码值置为空值 如FOREIGN KEY (Sno) REFERENCES S(Sno) [ON DELETE [CASCADE | SET NULL] ]修改类似 **对约束的命名、撤消和添加** AGE SAMLLINT **CONSTRAINT** AGE\_VAL **CHECK**(AGE >= 15 AND AGE <= 25) **alter** table S **drop constraint** S\_PK **alter** table SC **add constraint** SC\_CHECK **check**(Sno in select Sno from S) **域约束的创建、撤消与添加**create domain AGE\_DOMAIN smallint constraint DC\_AGE check(value <= 25 and value >= 15) alter domain AGE\_DOMAIN add constraint DC\_AGE check(value <= 35 and value >= 15) alter domain AGE\_DOMAIN drop constrain DC\_AGE **索引：**在关系的属性上所创建的索引是一种数据结构，它允许数据库系统高效的找到关系中那些在索引属性上取给定值的元组而不用扫描关系中的所有元组 例create [cluster] **index** s-index **on** S（SN） **安全性控制定义**安全性控制保护数据库以防止不合法的使用所造成的数据泄露和破坏。其基本措施是存取控制**安全性措施**物理级人际级操作系统级网络级数据库系统级**权限的转授和回收**允许用户把已获得的权限转授给其他用户，也可以把已授给其他用户的权限再回收上来 **授权命令 例：**grant select ,insert on S to Liming with grant option grant all on S to public**回收权限** revoke insert on S from Limin**g 不需要游标的数据操作**结果是一个元组的select语句；insert语句；delete语句；update语句**需要游标的数据操作**当select语句的结果中包含多个元组时，使用游标可以逐个存取这些元组 活动集：select语句返回的元组的集合 当前行：活动集中当前处理的那一行。游标即是指向当前行的指针 **除运算的定义**R=(X，Y)，S=(Y)，对应两个表r、s 则：r÷s=｛tr［X］｜ tr∈r∧ S  Yx ｝**域类型**char（n）：固定长度的字符串varchar（n）：可变长字符串int：整数smallint：小整数类型numeric（p，d）：定点数，小数点左边p-d位，右边d位float：浮点数double precision：双精度浮点date：日期（年、月、日）time：时间（小时、分、秒）interval：两个date或time类型数据之间的差 **3NF的模式分解算法证明**1法的关键是求Fc。2法产生了一个保持函数依赖且化为3NF的分解。证：因为Fc与F等价，Fc的每个依赖都在一个子模式中，他们的并集是逻辑蕴涵Fc ,所以保持依赖十分明显。用反证法，设Ri=(X Ai)不是3NF，则Fc在Ri上的投影必含有函数依赖Z→B , Z不是Ri的候选码，B是非主属性，分两种情况：(1) B = Ai, Z是X的一部分，既然Z→Ai 存在，按极小函数依赖集的定义，应由Z→Ai代替X→A，这与Fc是极小函数依赖集矛盾。(2) B≠Ai，Z和B都属于X，因B是非主属性，X不是Ri的候选码，设Ri候选码为K，故K是X的真子集，按极小函数依赖集的定义，K→Ai应取代X→Ai，这与Fc是极小函数依赖集的假设矛盾。因此Ri属于3NF, 证毕。**证明3nf 一定属于2nf 反证：若R∈3NF， 但R∉2NF，则按2NF定义，一定有非主属性部分依赖于码。设X为R的码，则存在X的真子集X’，以及非主属性Z，使得X’→Z。于是在R中存在码X，属性组X’，以及非主属性Z，使得X→X’，X’→Z，X’→X成立，这与R∈3NF矛盾。 所以R∈2NF。证明BCNF ⊂ 3NF反证：若R∈BCNF，但R∉3NF，则按3NF定义，一定有非主属性对码的传递依赖，于是存在：R的码X ，属性组Y，以及非主属性Z（Z Y），使得X→Y， Y→ Z，Y→X成立 。由Y→Z，按BCNF定义，Y含有码，于是Y→X成立，这与Y→X矛盾。所以R∈3NF R属于4NF，必属于BCNF。证明**：假设R属于4NF，但不属于BCNF，那么有X→A存在且X不是超码，如果XA=U，显然X是超码；如果XA<>U, 根据X→A成立，有X→→A成立，此时X不是超码，违反了4NF条件。因此R必是BCNF。 事务的执行顺序称为一个**调度**表示事务的指令在系统中执行的时间顺序。 如果多个事物依次执行，则称为事物的**串行调度** 如果利用分时的方法依次处理多个事物，则成为事物的**并发调度**  **冲突等价** 如果调度s经过一系列非冲突指令交换转换成调度s'，两者冲突等价当一个调度S与一个串行调度冲突等价时，则称该调度S是**冲突可串行化**。如果某个调度视图等价于一个串行调度，则称该调度是**视图可串行化(**两个调度S、S’满足 对每个数据项Q的初值，两个调度是同一个事务读。 事务Ti读取的Q值，在两个调度中，都是由同一个事务产生的。Q的终值，两个调度是同一个事务写入。 则S与S’等价。若S’是一个串性调度，则S是视图可串性化)的，冲突可串行化调度一定是视图可串行化的  **视图可串性化的扩展优先图（带标记）判断**T1，T2，Ti,Tj.表示为结点对调度S增加两个虚事务: 头事务Tb只有写操作Wb（Q）,尾事务Tf只有写操作Rf（Q）得s’找出所有写读对Wi(Q)Rj(Q), 加0标记有向边Ti-0→Tj 对调度中间的每个Wk(Q) 对应头写读对Wb(Q)Ri(Q), 加0标记有向边Ti-0→Tk 对应尾写读对Wj(Q)Rf(Q), 加0标记有向边Tk-0→Tj 对应中间的写读对Wi(Q)Rj(Q), 加p标记有向边Tk-P→Ti和Tj -P→Tk（两条边）如果无环，是视图可串性化的调度。如果去掉P标记边对的一条，无环，也是视图可串性化调度。等价的串性调度是，按有向边出口依次截取结点。**试述ER模型、层次模型、网状模型、关系模型和面向对象模型的主要特点。** ER模型：接近于人的思维，容易理解；与计算机无关，用户容易接受。只能说明实体间语义的联系，不能进一步说明详细的数据结构。 层次模型：记录之间的联系通过指针实现，查询效率较高。只能表示1：N联系，实现M:N结构较复杂；由于层次顺序的严格和复杂，引起数据的查询和更新操作也很复杂。 网状模型：记录之间联系通过指针实现，M：N联系也容易实现（每个M：N联系可拆成两个1：N联系），查询效率较高。编写应用程序比较复杂，程序员必须熟悉数据库的逻辑结构。关系模型：用关鍵码而不是用指针导航数据，表格简单，用户易懂，编程时并不涉及存储结构、访问技术等细节。 **为什么关系中的元组没有先后顺序** 因为关系是一个元组的集合，而元组在集合中的顺序无关紧要。因此不考虑元组间的顺序，即没有行序。 **交互式SQL**：在终端交互方式下使用的SQL语言称为交互式SQL**嵌入式SQL**：嵌入在高级语言的程序中使用的SQL语言称为嵌入式SQL。**关系连接的连接成分**包括两个输入关系、连接条件（决定两个关系中哪些元组相互匹配，以及连接结果中出现哪些属性如nature、on <谓词>、using (A1, A2 ,…, An）连接类型（决定如何处理与连接条件不匹配的元组如inner join、left outer join、right outer join、full outer join）**自然连接**出现在结果关系中的两个连接关系的元组在公共属性上取值相等，且公共属性只出现一次**on <谓词P>**出现在结果关系中的两个连接关系的元组在公共属性上取值满足谓词条件P，且公共属性出现两次**using (A1, A2 ,…, An)** (A1, A2 ,…, An)是两个连接关系的公共属性的子集，元组在(A1, A2 ,…, An)上取值相等，且(A1, A2 ,…, An)只出现一次**内连接**舍弃不匹配的元组**左外连接**内连接+左边失配的元组（缺少的右边关系属性用null）**右外连接**内连接+右边失配的元组（缺少的左边关系属性用null）**全外连接**内连接 + 左边失配的元组（缺少的右边关系属性用null）+ 右边失配的元组（缺少的左边关系属性用null）**cross join**两个关系的笛卡儿积**union join**左边关系中失配的元组+ 右边关系中失配的元组 对于外连接，连接条件是必须的 对于内连接，连接条件是可选的，没有连接条件等价于两个关系的笛卡儿积 **对象－关系数据模型**通过增加面向对象特征和处理新建数据类型来扩展关系模型；允许元组的属性域为复杂类型，包括如嵌套关系等非原子性的值；在扩展建模能力的同时，保留关系的基本特征，如说明性查询；提高与现有关系语言的兼容性 **对关系模型的扩展（**嵌套关系、复杂类型、特殊化、继承、对象标识）**复合类型（**结构类型：不同类型元素的有序集合 数组类型：同类元素的有序集合 多集类型(multiset)：同类元素的无序集合，并且允许一个成员多次出现 集合类型(setof)：同类元素的无序集合，一个成员最多只能出现一次 **DBMS的主要功能**有：（1）数据库的定义功能 （2）数据库的操纵功能 ）数据库的保护功能　（4）数据库的存储管理 （5）数据库的维护功能 （6）数据字典。**死锁发生的好坏原因方法**：死锁本身是一件坏事，但坏事可以变成好事，如果不让死锁发生，让事务任意并发下去，有可能破坏DB的数据，或用户读了错误的数据，从这个意义上说死锁的发生时一件好事，能防止错误的发生，当死锁发生时，系统的死锁处理机制和恢复程序就能起作用，系统就会选择一个处理死锁代价小的事务，将其撤销，做rollback操作，使系统有可能摆脱死锁状态，继续下去。**传统关系模型与对象-关系模型区别 传统**属性只能是基本数据类型属性可以是复合数据类型 **对象**可以直接表达E-R中的复合属性和多值属性 **数据类型可以嵌套例**课程成绩单{(MATHS，80)，(PHYSICS，90)，(OS，85)，(DB，95)}外层是集合类型，内层是结构类型  **对象数据库定义，例**：create type MyString char varying. create type MyDate(day integer,month char(10),year integer) create type Document (title MyString,author-list setof(MyString),,date MyDate,keyword-list setof(MyString)) create table doc of type Document **类型继承 c**reate type Person (name MyString,social-security integer) create type Student (degree MyString, department MyString) under Person create type Teacher (salary integer, department MyString)under Person **多重继承**引起的属性冲突 通过as给属性重命名来消除冲突 **表级继承**允许一个实体同时存在于多于一个的表中，使得一个对象可以拥有多个类型create table people (name MyString,social-security integer) create table students (degree MyString, department MyString) under people create table teachers(salary integer,department MyString) under people **表级继承的角色** 表级继承允许对象有多重类型，而不必象类型继承那样要求具有一个最明确类型 如一个对象可以同时在students和teachers子表里，表示一个身为学生的助教，不必再为它创建一个继承students与teachers的子表 对象可以获得或失去角色——通过向子表里插入或删除对象 **表级继承的一致性要求**父表与子表之间的完整性约束父表(people)中的一个元组至多与它的每一个子表(students和teachers)中的一个元组相对应（不能有两个student对应同一个people） 子表(students、teachers)中的每个元组恰好与它的父表(people)中的一个元组相对应（不能出现student不是people的情况，也不能出现一个student是多个people的情况） 子表可以不必存储的所继承的父表的属性（除主码外）**引用类型1.**类型引用类型的属性可以是对属于指定类型的对象的应用 如声明：author-list setof (ref(Person)) 表明author-list是一个对Person对象引用的集合2.表引用 author-list setof (ref(people)) 用表的主码或元组的标识符来实现对元组的引用}**典型题目：1>属性闭包的求法2>候选码的求法**：一个属性集函数决定U，若它的任一个子集都不能函数决定U，则属性集可作为候选码。候选码从箭头左边的属性中找。仅在箭头左边的属性一定在候选码中。**3>判断F和G是否等价：**检查F中的每个函数依赖X→Y，检查Y是否属于X+G。检查G中的每个函数依赖X→Y，检查Y是否属于X+F。4>**计算函数依赖集F的正则覆盖（正则覆盖未必唯一）**F的正则覆盖是一个与F相等的最小函数依赖集。特点：每个函数依赖的右端是一个属性；函数依赖集没有多余的依赖（A→B，C→D，两个都不可以删掉）；每个函数依赖左端决定因素没有多余属性。（AB→C，A，B都不可以删掉）**5>3NF的判断：步骤的关键是确定候选码。**(1) 确定R的候选码，主属性，非主属性，决定因素。 (2) 如果R的每个属性都是基本数据项，则R∈1NF。(3) 如果R∈1NF,检查所有非主属性是否对所有候选码都不存在部分函数依赖。若是，R∈2NF。当候选码都是单一属性时，R必然属于2NF。(4) 如果R∈2NF，检查所有非主属性是否对所有候选码都不存在传递函数依赖，若是，R∈3NF。当R没有非主属性时，R必属于3NF。**6>判断一个分解是否是无损分解。**方法一（分解为2个子模式的）：将R分解成R1，R2，若中至少存在以下一个依赖，则是无损分解：(U1∩U2)→(U1-U2)∈F+ 或(U1∩U2)→(U2-U1)∈F +方法二（分解成2个以上子模式的）：(1) 构造n列（属性），k行（子模式）的矩阵M(2) 逐个检查F中函数依赖X→Y,修改表中的元素。过程如下：按X列值相等确定各行，修改Y列对应行的值；如果其中之一为aj，则全改为aj，否则全改为bmj，m为这些行中的最小行号。**7>典型题目：保持函数依赖的分解的判断。**投影的定义：F在子模式Ri=（Ui）上的投影为：ΠRi (F)={X→Y｜X→Y∈F+∧XY属于Ui}注意：从F +中找。保持函数依赖的分解：设ρ是R的一个分解，F是R的函数依赖集，如果G= ΠR1(F)∪ΠR2(F)∪…Π Rk(F)逻辑蕴涵F，则称ρ是保持函数依赖分解，或简称保持依赖。**8>3NF的模式分解算法（且保持函数依赖）:**(1)求Fc(2)如果Fc中有一函数依赖X→A,且XA=U，则ρ={R},输出ρ，结束。(3)对Fc中的每一个X→Ai都构成一个关系模式Ri=(X , Ai)。如果Fc中有X→A1,X→A2,…,X→An(左部决定因素相同)则以R=(X,A1,A2,…,An)构成一个模式。**9>求分解关系模式为3NF且保持函数依赖性和无损连接性的分解。10>BCNF的无损连接分解**方法：(1) 初始化ρ={R}。(2) 如果ρ中所有关系模式均已是BCNF转(4)，否则，从ρ中找出非BCNF的关系模式S，在S中必有X→A, X不包含S的码，A也不在X中。将S分解为S1=(X，A)，S2=(Us-A)，Us为S的属性组。(3) 在ρ中以{S1,S2}代替S，转(2)。(4) 结束，输出ρ**ID3算法** 基本思想是贪心算法。首先检测训练数据集的所有属性，选择信息增益最大的属性A建立决策树根节点。由该属性特征的不同取值建立分支。对各分枝的实例子集递归，用该方法建立树的节点和分枝，直到某一子集中的数据都属于同一类别，或者没有特征可以再用于对数据进行分割。**属性信息增益（互信息）计算**：I（U，V）=H（U）-H（U/V）=类别熵–条件熵其中，U为类别集合，{N，P}V为属性的值集合**类别熵（先验熵）的计算：**H（U）=∑P(ui)log2(1/P(ui))=-∑P(ui)log2(P(ui))，P(ui)=属于ui类的样本数/全体样本数**条件熵（后验熵）的计算**：H（U/V）=∑P(vj)∑P(ui︱vj)log2(1/P(ui|vj)))，P(vj)=属性取值为vj的样本数/全体样本数；P(ui︱vj)=属性取值为vj的样本中属于ui类的样本/属性取值为vj的样本数**朴素贝叶斯分类算法:**每个样本表示为n维向量:X={x1,x2,…,xn},属性A1,A2,…An的值**。**设定m个类C1,C2,…Cm,对未知类的样本X,当且仅当P(Ci︱X)> P(Cj︱X) ,(1≤j ≤m, i≠j), 将X分配给类Ci, 即P(Ci︱X)是最大的,按贝叶斯公式:P(Ci︱X) = P(X︱Ci) P(Ci)/ P(X。其中: P(X)对每个类是常数,只需求P(X︱Ci) P(Ci) **。**P(Ci)=Ci中的训练样本数/训练样本总数**。**如果P(Ci)未知,假定P(C1)= P(C2)=…= P(Cm)按属性值相互条件独立的朴素假定P(X︱Ci)= ∏nk=1 P(Xk︱Ci)，其中: P(Xk︱Ci)=类Ci中具有Xk值的样本数/类Ci中样本数**k-平均算法**方法：初始随机选择k个对象，代表类的平均值（类中心），其余对象计算与每个类中心的距离，划分到最近距离的类中，重新计算类中心，重复。**k-中心点算法**：1）任选k个对象作为初始类中心点2）repeat3) 将每个对象指派给离中心点最近的类 4）随机选择非中心点h, 计算h替换中心点i的总代价RChi , 如果RChi < 0 , 用h替换中心点，然后对非中心点对象按与新的中心点的距离，指派类5）until 不再发生变化**说明：RChi=∑ki=1∑p∈Ci(︱p-mi’︱2 -︱p-mi︱2)**其中： RChi为h替换中心点i的总代价，p是给定对象，mi 是类C i 的中心点， mi’是类C i 用h替换中心点i的后的中心点。基本思想是设置初始中心点后，反复用非中心点对象替换中心点，以改进聚类质量。因为k-平均对孤立点是敏感的 **Apriori算法（找频繁项集算法）：**基本概念：频繁项集——所有支持度大于最小支持度的项集称为频繁项集，或简称项集。强关联规则——同时满足最小支持度阈值和最小可信度阈值的规则称为强规则。找出所有的频繁项集，由频繁项集产生强关联规则。**TF-IDF加权标引：**TF(term frequency):词条频率,在文档中出现的次数,定义为: tfj(Di)**。**IDF(inverse document frequency):文档频率倒数,出现词条tj的文档占整个文档集合的倒数的对数,定义为: Log(N/nj)**。**N是文档总数, nj是出现词条tj的文档数TF-IDF权= tfj(Di)\* Log(N/nj)显然,TF越大说明该词条在文档中出现越多,越重要**。**nj越大,Log(N/nj)越小,说明tj出现的文档越多,越不重要,例如“的”出现在所有文档中,不重要.IDF取对数的目的是使权对N变化不敏感**文件系统的缺点**：数据的冗余和不一致性，数据访问困难，数据孤立，完整性问题，更新的原子性问题，多用户的并发访问，安全性问题**列出选修了全部课程的学生姓名：**任意：not exists(not exists)不存在一门课该同学没有选过：select SNAME from S where not exists (select CNO from C where not exists (select \* from SC where SC.CNO=C.CNO and SC.SNO=S.SNO)) 超集:not exists(B except A)所求学生的选课集为所有课程的超集：select SNAME from S where not exists((select CNO from C)EXCEPT (select CNO from SC where SC.SNO=S.SNO)) ÷：not in(not in) 所求学生不在如下集合中：学生学号与任一课程组合不包含在SC中：select SNAME from S where SNO NOT IN (select SNO from C,S where (SNO,CNO) NOT IN(select SNO,CNO from SC))**ER图实体间联系的转换**总原则：N：M联系单独建表，其他，增加属性**两实体间1：N 的联系**。将1方实体的码纳入N方实体对应的关系中，作为N方实体关系的外码。如果联系本身有属性，则将属性也纳入N方实体对应的关系中。**同一实体间1：N 的联系**。在实体对应的关系中增加属性，属性包括：与N个体相联系的1个体的码，以及联系的属性。**两实体间M：N 的联系**。联系单独转换为一个关系，属性包括：两实体的码、联系的属性。**同一实体间M ：N 的联系**。联系单独转换为一个关系，属性包括：个体的码、与之有联系的另一个体的码、以及联系的属性。**多实体间M：N 的联系**。联系单独转换为一个关系，属性包括：各实体的码、联系的属性。**两实体间1：1 的联系**。将任一方实体的码纳入另一方实体对应的关系中，作为外码。如果联系本身有属性，则将属性也纳入对方实体对应的关系中。**强实体集**，将每个强实体集直接转换为一个关系，实体的码作为关系的码，实体的属性作为关系的属性。Student (ID,name,tot\_cred)**弱实体集**，将每个弱实体集直接转换为一个关系，关系中增加强实体集的码。**多值属性**单独建表**冲突可串性化的优先图判断法**判断调度S是否是冲突可串性化:T1，T2，...Ti,...Tj...表示为结点。如果 Ri(Q)之后有Wj(Q),或Wi(Q)之后有Rj(Q) ，或 Wi(Q)之后有Wj(Q), 则Ti、Tj之间加有向边Ti→Tj若图无环，S是冲突可串性化的。等价的串性调度是，按有向边出口依次截取结点。