数据库技术的重要性1）提高效率2）改变或创新商业模式3）改变生活方式4）增加人类自由的维度 内涵：是概念的性质和特征外延：是满足上述性质及特征的所有客观个体的集合二维表 内涵列 外延行 数据库管理阶段的基本特点：数据的管理者：DBMS(与应用独立的专门管理软件)数据面向的对象：现实世界(添加和删除方便易行)数据的共享程度：共享性高数据的独立性：高度的物理独立性和一定的逻辑独立性(随着不同发展时期，独立性程度在不断改进)数据的结构化：整体结构化 数据控制能力：由 DBMS 统一管理和控制数据模型是一种用来抽象、表示和处理客观世界数据对象结构的描述方式是对客观世界的模拟(一种主观建模）(1) 概念模型：（也称信息模型）是按用户的观点来对数据和信息建模；几乎不涉及计算机专业技术知识。E-R 模型（面向客观世界建模）2)数据模型：（逻辑）主要包括网状模型、层次模型、关系模型、对象模型等；是按计算机系统的观点对数据建模（面向计算机实现建模）两步抽象现实世界中的客观对象抽象为概念模型；把概念模型转换为某一 DBMS 支持的数据模型逻辑模型 关系数据（表）数据库系统是指由数据库管理系统和相关工具组成的软件系统，用于管理和操作大量数据 数据库系统开发环节需求分析.系统设计(概念设计,逻辑设计,数据存储设计物理模型).系统实现.系统运行与维护数据库开发数据需求.(概念设计.逻辑设计.物理设计).数据库实现.数据维护.数据分析 三层模式-两级映射数据逻辑重构.数据逻辑结构.数据物理存储 E-R 模型-是一种描述方法 E-R 图-采用 E-R 模型方法,对一具体应用的描述(结果)实体集&属性描述数据对象及特征(内部结构)；联系集描述数据对像间联系(外部结构)。联系集可以带属性表示联系的特征实体集：相关类型实体(对象)的集合联系集：相关类型联系(连线)的集合弱实体集特点 1)没有键；2)存在依赖于主实体集；3)键由主实体集键和它的分辨符合并构成。一对多 一是箭头多没剪头双线代表全参与 E-R 模型可描述任何复杂客观对象，实体集可以是任何一种复杂数据结构因为：E-R 模型重点是面向客观世界，建立易于用户理解的抽象数据模型（它不关心数据如何才能够被实际存储）在 ER 模型中，弱实体用双线矩阵表示；与弱实体相关的联系，用双线菱形表示。特化：自顶向下的设计过程 概化：自底向上的设计过程 部分概化：允许父实体不属于任何子实体集(缺省表示) 全部概化：每个父实体必属于某一子实体集(采用标识 total)聚集：是一种抽象：它将联系集（及其相关实体集）看着是一个更高层的抽象实体集 数据模型：是一个描述数据、数据联系、数据语义以及数据一致性约束的概念工具的集合应包括：数据结构：由一组创建数据库的规则（定义数据库的结构）组成 数据操作：定义对数据进行的操作类型（包括更新和查找数据库中的数据以及修改数据库的结构）约束条件：一组数据完整性定义规则，确保数据的正确性。非层次结构的描述 采用副本 缺点：数据冗余（增加空间，一致性维护难）虚拟记录(优化方法)不足：指针操作增加开销 定义物理存储结构邻接法：按照层次树前序穿越的顺序，把所有记录值(子段定长，定长记录)依次邻接存放。即通过物理空间的位置相邻来实现层次顺序。层次模型小结：优点：数据模型比较简单，操作简单。对于实体间联系是固定的，且预先定义好的应用系统，性能较高。提供良好的完整性支持。缺点：不适合于表示非层次性的联系。对插入和删除操作的限制比较多。查询子女结点必须通过双亲结点 网状模型的不同特点它去掉了层次模型的两个限制：允许多个结点没有双亲结点；允许结点有多个双亲结点。它还允许两个结点之间有多种联系（复合联系）网状模型(的属性)为何允许复杂数据类型？因为物理实现采用指针链网状模型小结优点：能够更为直接地描述现实世界。具有良好的性能，存取效率较高。缺点：其 DDL 语言极其复杂。数据独立性较差。由于实体间的联系本质上通过存取路径指示的，因此应用程序在访问数据时要指定存取路径。第 1 范式(1NF)要求关系的属性具有原子性主码完整性约束 但一个关系仅允许有唯一的主码，区分不同元组外码参照完整性约束(其所参照的主码的值;取空值)作用:1)说明元组间联系 2)保证数据有效性 关系模型的完整性约束实体完整性规则:在任何关系的任何一个元组中，主键的值不能为空值、也不能取重复的值。参照完整性规则（引用完整性规则）:规则要求：“不引用不存在的实体”。即：不允许在一个关系中引用另一个关系中不存在的元组。外码的取值：要么取其所参照的主码的值；要么取空值目的用于确保相关联的表间的数据保持一致。域完整性规则（用户自定义完整性规则）:由用户根据实际情况，定义表中属性的取值范围 聚集函数计算一个集合的数值，得到一个值做为结果 avg: 求平均值 min:求最小值 max: 求最大值 sum:求和 count:求元组数 简单实体集的转换新的关系模式 弱实体集没有足够属性形成主码的实体集的转换新的关系模式:1)属性集:弱实体集的属性集+主实体集的主码 2)主码:主实体集的主码+弱实体集的分辨符复杂实体集的转换新的关系模式(1NF):1)属性集:包含实体集的所有简单属性，此外其复杂属性需逐层展开变换为多个简单属性 2)主码:需根据唯一确定元组特征来确定注 1:弱实体联系冗余模式,可忽略掉！注 2:可推广到多元联系情形联系到关系模式的转换 合并主码相同的关系模式，因这些属性均由相同主码唯一确定！联系转换 1:M 1 方主键出现在 M 方成为外键 1:1 任意一方主键出现在另一方中，成外键 M：N 联系建为一新关系，其主键由两个父实体的主键复合组成 SQL DDL：数据定义语言，用来定义数据库对象：库、表、列等；DML：数据操作语言，用来定义数据库记录（数据）；DCL：数据控制语言，用来定义访问权限和安全级别；DQL：数据查询语言，用来查询记录（数据）create database1. 产生了一个数据库（空仓库，仅包括系统数据字典）2. 初始库小，数据增长需要时才增大库空间 3. 同时，还产生了一个日志存放的空仓库（备份回复用）4.还涉及到物理设计工作：库放在何位置、库大小、库增量而且日志仓库位置可用与数据仓库位置不同(保证安全)！并 Union 差 Except，交 Intersact 自然连接 Natural join 与迪卡儿积X两点最大不同 1）仅包含符合连接条件的元组 2）连接属性仅出现一次 聚集函数平均值 avg 最小值 min 值大值 max 总和 sum 计数 count 嵌套子查询嵌套子句可以多种方式用在 where 中!并显著增强了 SQL 的查询能力！distinct 去重复元组 删除 delete from instructor where dept_name in (select dept_name from department where building = ’Watson’);插入 insert into course (course_id, title, dept_name, credits) values (’CS-437’, ’Database Systems’, ’Comp. Sci.’, 4); insert into student values (’3003’, ’Green’, ’Finance’, null);insert into instructor select ID, name, dept_name, 18000 from student where dept_name = ‘Music’ and tot_cred > 144; 更新 update instructor set salary = salary * 1.03 where salary > 100000;update instructor set salary = case when salary <= 100000 then salary * 1.05 else salary * 1.03 end; 视图 create view physics_fall_2009 as select course.course_id, sec_id, building, room_number from course, section where course.course_id = section.course_id and course.dept_name = ’Physics’ and section.semester = ’Fall’ and section.year = ’2009’;1）可以在任何 SQL 语句中像表一样的被使用！2）增强查询能力且方便(用户/程序员)使用！3）还可以提供数据访问安全控制(隐藏数据)！4）作为外模式(1 级映射)有利于应用独立性！完整性约束键完整性约束（主码/主键）关系(模式)必需有一个主码，来区分不同元组！参照完整性约束（外码/外键）用另一关系的主码，来约束属性取值的有效性！非空、唯一、范围完整性约束 断言 当数据更新时，保持谓词为真。(否则拒绝更新)但作用有利有弊 授权 grant 回收 revoke 索引 create index dept_index on instructor(dept_name) drop index 函数 create function dept_count(dept_name varchar(20));return integer;begin;declare d_count integer;select count(*) into d_count;from instructor;where instructor.dept_name=dept_name;return d_count;end 存储过程 create procefure dept_count_proc(;int dept_name varchar(20),out d_count integer);begin;select count(*) into d_count;from instructor;where instructor.dept_name=dept_count_proc.dept_name;end 触发器 create trigger check-delete-trigger after delete on account;referencing old row as orow;for each row begin;delete from depositor;where depositor.customer_name not in;(select customer_name from depositor;where account_number <> orow.account_number)end 关系模式设计优化 当数据的某些部分能由其他部分推导出来，就意味着存在冗余 函数依赖如果对于关系实例 r 中的任意一对元组 t1 和 t2，有 t1.X = t2.X 逻辑蕴涵 t1.Y =t2.Y，那么，函数依赖 X→Y 在关系 r 中成立 关系模式优化的依据是什么如果一个关系满足一种范式（BCNF, 3NF 等.），就能判断该关系模式是否避免了某类问题。这样就能知道该关系是否需要分解**第一范式(1NF)** - 每个属性都是原子属性。 - 本质上所有关系都满足第一范式**第二范式（2NF）**- 任何满足第二范式的关系满足第一范式 - 所有非主属性必须依赖于整个候选码而不能依赖于候选码的部分属性。**函数依赖集的闭包** 由 F 逻辑蕴含的所有函数依赖的集合。属性集的闭包：属性集闭包是指在给定的函数依赖集 F 下，某个属性集 X 可以推导出的所有属性的集合**作用**测试超键；检测函数依赖；计算 F 的函数依赖集闭包 **模式分解的基本标准** 1、无损连接分解 2、保持函数依赖 **函数依赖集等价** 如果 $F1^+ = F2^+$，那么 F1 和 F2 等价 **Boyce-Codd 范式(BCNF)** 对于 R 有函数依赖集 FDs F ，如果 R 符合 BCNF 当且仅当每一个非平凡的 FD X → A in F , X 是 R 的超键 i.e., X → R in F+). - 对于 FD X → A 是平凡的，当且仅当 A ⊆ X. 分解得到的关系，符合 BCNF，但可能不满足保持函数依赖. 3NF 关系 R ，函数依赖集 FDs F 符合 3NF，当且仅当 F 中每一个函数依赖 X → A (X ⊆ R and A ⊆ R)，下符合列描述中的一个: - A ⊆ X (平凡的 FD), or - X 是超键, or - A 是 R 的键的一部分分解是无损的– 分解是保持函数依赖的，会产生冗余 物理设计(数据库存储技术)物理设计任务：考虑用文件表示逻辑数据模型(数据库模式)的不同方式。文件组织一个数据库被映射到多个不同的文件(file) ，文件由操作系统来管理，这些文件被永久存储在磁盘上！一个文件在逻辑上被组织成记录的一个序列，记录被映射到磁盘块(block)上。每个文件(file)被分成(变长)定长的存储单元-块(block)，块是数据存储和传输的基本单

位(默认一般是 4-8KB)。一个块可以包括很多记录(假设一个记录总比块小；对大数据如图片需单独处理和存储)，且一个记录的数据不能跨块存储 文件中记录的组织方式 堆文件：记录在文件空间中任意放置 顺序文件 ：按一定的顺序在文件中组织记录 散列文件：按照散列函数计算值存放相应记录 多表簇集文件：不同关系表里的记录存放在同一个文件中。实现多表聚集文件组织，需底层操作系统配合,实现对文件的管理(只有大型数据库系统才支持) 物理设计任务在定义关系模式时，需要确定采用定长还是变长记录；对每个关系模式，需要确定影响记录存放次序的搜索码；对每个关系模式，需要确定是否还需要建立辅助索引文件；对具有连接条件的一组关系模式，需要确定是否采用多表聚集文件存储；对应用的所有关系模式，需要确定应当划分为多少个数据库来存储；对每个数据库，需要确定数据库文件存放的物理路径（不同服务器，不同介质） 元数据 metadata 是记录数据的数据，用于描述数据属性，其中包括的数据结构含数据集的名称、关系、字段、约束等 关于关系的关系模式和其他元数据存储在称为数据字典或系统目录的结构中 缓冲区 1)CPU 处理信息快捷，但从磁盘读取记录缓慢.2)缓冲区一次 I/O 读硬盘上多个记录(按块),可明显减少磁盘 I/O 开销(连续读-节省时间)；3)缓冲区中的记录，可能为多个应用所需要，可明显减少磁盘 I/O 开销(重复读-浪费时间)。数据库缓冲区管理器 1)若 X 不在缓冲区中，则为其分配空间，并向硬盘(硬盘上物理数据库)请求 X；2)若缓冲区已满，则按某种测量策略清除部分块，清除前先将块中最新修改数据回写到硬盘（数据库）!3）缓存区从硬盘（数据库）读取 X; 缓冲区替换策略 最近最少使用 立即丢弃 最近最常使用 (数据库)索引是一种与(数据库)文件相关联的附加结构，额外增加的一个辅助文件！搜索码：用于在文件中查找记录的属性/属性组；索引项：由一个搜索码值和指向具有该搜索码值的一条/多条记录的指针构成。聚集索引：按照某种顺序存储 稠密索引：索引文件中，每个搜索码都有一个索引项稀疏索引：索引文件中，只为某些搜索码建立索引项 稀疏索引 1)降低索引文件空间开销，2)提高搜索效率(跳跃查找) 索引加快查找速度 1)索引小，可在内存中处理 2)索引排了序，查找效率高 3)避免逐个读全部记录文件 辅助索引 必须是稠密索引！对每个搜索码都有一索引项，对每个记录有一个地址指针。散列在查找一个特定的值时更可取，顺序索引在指出了一个值的范围的查询中比散列更可取 查询处理 查询过程：语法分析和翻译 优化 执行 查询处理的代价 磁盘存储（主要）传送磁盘块数和搜索磁盘次数、执行一个查询所用的 CPU 时间、在并行/分布式数据库系统中的通信代价 选择操作的实现简单的全表扫描方法 索引(或散列)扫描方法连接操作的实现嵌套循环方法 排序-合并方法 索引连接(index join)方法 Hash Join 方法 物化-将中间计算结果作为临时关系存放，支持上一层计算 流水线计算 减少查询执行中产生的临时文件数，通过将多个关系操作组合成一个操作的流水线来实现 查询代价 总代价=I/O 代价+CPU 代价+内存代价＋通信代价 优化策略 选择下移 投影下移 选择连接顺序”优化策略小关系的连接应优先查询优化步骤 1 将查询转换成某种内部表示，通常是语法树 2. 根据一定的等价变换规则把语法树转换成标准（优化）形式 3. 选择低层的操作算法对于语法树中的每一个操作计算各种执行算法的执行代价选择代价小的执行算法 4. 生成查询计划(查询执行方案)查询计划是由一系列内部操作组成的 查询优化 -openGauss 常量表达式化简、子查询优化、 选择下推和等价推理、外连接消除、DISTINCT 消除、IN 谓词展开、视图展开 启发式优化 尽早执行选择运算 尽早执行投影运算 事务：构成单一逻辑工作单元的操作集合。作用 DBMS 通过保证要么执行整个事务，要么一个操作也不执行，达到使数据始终处于一个一致状态。特性：A 原子性(事务要么完成要么不完成) C 一致性(执行前后数据库一致) I 隔离性(彼此不干扰) D 持久性(完成之后长久生效) 事务状态图作用：便于系统跟踪各事务执行情况 保证事务原子性和持久性特点 活动状态-部分提交状态-提交状态（失败状态-终止状态）调度：一组指令包括指令在系统中执行的特定时间顺序。包括 commit,abort 指令串行调度：调度中凡属于同一个事务的指令都紧挨着一起。即一个事务的指令都执行完成后在执行下一事务并行调度：调度中多个事务的指令在时间上相互交叉地在执行 可串行化调度：虽然可能是一个并行调度但在执行的效果上等同于某一个串行调度执行结果 冲突等价 可以通过一系列非冲突指令转换 冲突可串行化 一个调度和一个串行调度冲突等价 调度优先图无环，则可以冲突可串行化 可恢复调度 如果一个事务 Tj 读取了以前由事务 Ti 写入过的数据 ，则 Ti 的提交操作出现在 Tj 的提交操作之前 无级联调度则 Ti 的提交操作出现在 Tj 的读提交操作之前 事务隔离性级别 可串行化 可重复读 已提交读 未提交(脏读 幻读 不可重复读) 并发控制 1.提高吞吐量和资源利用率 2.减少等待时间 共享锁 排他锁 为了保证数据的一致性(事务的隔离性)和提高系统的并发处理能力! 封锁协议 死锁 形成两个事务“相互等待”对方释放资源各自才能往下继续做的僵局（不加控制的封锁机制）活锁 出现一个事务永远(长时间)等待某一数据项被其它事务释放后才能进行封锁的现象两阶段封锁协议 事务分两个阶段提出加锁和解锁申请 增长阶段 缩减阶段 保证冲突可串行化 两阶段封锁协议保证冲突可串行化，冲突可串行化未必符合两阶段封锁，级联回滚可能发生 严格两阶段封锁协议 事务所持有的排他锁必须在事务提交后方可释放 可保证调度不会出现级联回滚 强两阶段封锁协议 事务提交前不得释放任何锁 死锁 等待图检测 解除方法 选择牺牲者 回滚 饿死 预防 每个事务开始之前封锁他的所有事务项 对所有的数据项强加一个次序 使用抢占和事务回滚（时间戳 wait-die 老事务礼让: 新事务不等待,回滚；wound-wait 年轻事务礼让:老事务不等待,强制使新事务回滚）多粒度锁：允许使用多种粒度/大小不同的锁 显式加锁：直接加锁隐式加锁：上级结点被加锁（检查本事务的显式封锁是否与该事务的上级隐式封锁冲突，所有上级的显式封锁是否与将要加到下级的隐式封锁冲突）意向锁：如果对一个结点加意向锁，说明它的下层节点正在被加锁。也是说，对任意结点加锁，必须对上层结点加意向锁 IS（除了 X）IX(IS IX) S(IS S)SIX(IS) 时间戳对于系统中的每一个事务 Ti，把一个固定的时间戳和它联系起来，记为 TS(Ti):系统时钟，该事务进入系统的时间;逻辑计数器，该事务进入系统的计数器值 决定了串行化的顺序 W-timestamp(Q)成功执行 write(Q)的所有事物的最大时间戳 R-timestamp(Q)成功执行 read(Q)的所有事物的最大时间戳 read≥W-timestamp write≥W-timestamp&R-timestamp 时间戳保证冲突可串行化和不会出现死锁现象 可恢复的调度—如果事务 Tj 读取了先前由事务 Ti 所写的数据，事务 Ti 需要在 Tj 之前提交 不能完全防止级联回滚 快照隔离 快照隔离是在事务开始执行时给它数据库的一份快照。事务在该快照上操作，和其他并发事务完全隔离。快照中的数据值仅包括已经提交的事务所写的值。对只读事务来说是理想的，不需要等待。更新事务，需要在更新写入数据库之前，处理与其他并发更新的事务之间存在的潜在冲突。先提交者获胜 先更新者获胜 基于有效性检查的协议 (1)读阶段：各数据项的值被读入并保存在事务的局部变量中，所有的写操作都是对局部临时变量进行的，并不对数据库进行真正的更新；⑵有效性检查阶段：判定是否可以将写操作所更新的临时变量的值拷入数据库，同时又不违反可串行性；⑶写阶段：若事务通过了有效性检查，则实际的更新可以写入数据库，否则事务回滚。功能：自动预防级联回滚 可能出现饿死现象备份和恢复故障的种类 事务故障(逻辑错误，系统错误) 系统鼓掌(崩溃) 介质故障(磁盘故障) 其他 存储器是计算和保存的基础计算过程中数据仅临时使用而最终结果数据将永远保存 易失性存储器 非易失性存储器 稳定存储器(磁盘镜像) 数据访问 事务由磁盘向主存输入信息，然后再将信息输出回磁盘。以块为单位 磁盘上物理块 贮存 缓冲块 内存中临时存放块 磁盘缓冲区 日志 记录在稳定存储器中 在内存中进行运算 对数据库的修改(记录在磁盘上 检查点 减少恢复过程中需要检查的日志记录数量，提高恢复效率。内容：包括建立检查点时刻所有正在执行的事务清单和这些事务最近的日志记录地址。远程备份系统 主站点 辅助站点 故障检测 控制权的移交 恢复时间 提交时间(一方保险 两方强保险 两方保险) DBMS 的体系结构 用户：集中式结构、客户/服务器结构、分布式结构、并行结构 数据管理系统 三级模式结构 集中式体系结构 主存储器（CPU 处理器(多核) 硬盘控制器 USB 控制器 图形适配器）单用户系统（个人计算机 工作站） 多用户系统 粗粒度并行是指在多个处理机上分别运行多个进程，由多台处理机合作完成一个程序，一般用算法实现，提高吞吐量。细粒度并行是指在处理机的指令级和操作级的并行性，将单个任务并行的执行。 客户-服务器体系结构 （用户 应用）网络（（应用服务器三层表现层（UI）、业务逻辑层（BLL）、数据访问层（DAL））数据库系统）前端：应用事务逻辑 后端：数据库事务逻辑 在数据库中，数据模型可以分为三个层次，分别称为外模式数据库用户能够看到和使用的局部数据的逻辑结构和特征的描述。是数据库用户的数据视图，是与某一应用有关的数据的逻辑表示。(概念)模式数据中全体数据的逻辑结构和特征描述，是所有用户的公共数据视图。它是数据库系统模式结构的中间层，既不涉及数据的物理存储细节和硬件环境，也与具体的应用程序及其所使用的开发工具和内(存储)模式一个数据库只有一个内模式。它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。三层模式两层映射 数据逻辑重构-数据逻辑结构-数据物理存储 服务器系统体系结构 事务服务器 数据服务器 前者事务应是指更一般的事务逻辑，后者是指数据库中的事务概念。 并行系统 吞吐量 响应时间 并行 DBMS 的体系结构一般有共享内存的、共享磁盘的和无共享的三种形式 分布式系统 地理上分离，分别管理，互联速度更低 分为局部事务和全局事务(数据共享 自治性 可用性) 保持原子性 两阶段提交 区别并行数据库系统高速网络连接的目标是充分发挥并行计算机的优势，利用系统中的各个处理机结点并行完成数据库任务，提高数据库系统的整体性能。分布式数据库系统区域网或广域网连接主要目的在于实现场地自治和数据的全局透明共享，而不要求利用网络中的各个结点来提高系统处理性能。