第一章

数据（Data）是数据库中存储的基本对象

数据是描述事物的符号记录

信息是具有时效性的，有一定含义的，有逻辑的、经过加工处理的、对决策有价值的数据流。

数据是符号化的信息，信息是语义化的数据

数据的三种分类：结构化数据（符合关系模式）、半结构化数据（模式随意）、非结构化数据

数据库（Database，简称DB）是长期储存在计算机内、有组织的、可共享的大量数据的集合。

数据库基本特征：
1.数据按一定的数据模型组织、描述和储存
2.可为各种用户共享
3.冗余度较小
4.数据独立性较高
5.易扩展

数据库管理系统(DBMS)是位于用户与操作系统之间的一层数据管理软件，用于帮助用户定义、创建、维护和控制数据库访问的软件系统

功能：数据定义、数据组织存储和管理、数据操纵和、数据库事务管理与运行管理、数据库建立和维护、其他（通信、数据转换）

数据库系统（DBS）：
1.数据库(Database)：底层的数据文件、文件结构和存取方式，数据字典（数据库结构/模式描述）
2.数据库管理系统（DBMS及其应用开发工具）
3.应用程序(Application)
4.数据库管理员(DBA)

数据库系统特点：数据结构化、数据的共享性高，冗余度低且易扩充、数据独立性高、数据由DBMS统一管理和控制

数据的整体结构化是数据库的主要特征之一、数据用数据模型描述

物理独立性：应用程序与数据库中数据的物理存储是相互独立的，应用程序不需要关心具体的物理数据结构和存储；当数据的物理存储改变了，应用程序不用改变。
逻辑独立性：应用程序与数据库的逻辑结构是相互独立的。 当数据的逻辑结构改变了，应用

程序不用改变（在某些条件下成立，很多时候应用程序需要改变）。
数据独立性由DBMS的二级映像功能保证。

数据模型是对现实世界数据特征的抽象，是数据库系统的核心和基础
两类模型：概念模型、逻辑模型和物理模型

概念模型也称信息模型，它是按用户的观点来对数据和信息建模，用于数据库的初始概念设计，反映对用户业务需求的基本理解
逻辑模型按计算机系统的观点对数据建模，主要包括网状模型、层次模型、关系模型、面向对象数据模型、对象关系数据模型、半结构化数据模型等
物理模型是对数据最底层的抽象

数据建模的三个层次：概念建模、逻辑建模、物理建模

客观存在并可相互区别的事物称为实体。
实体所具有的某一特性称为属性。
唯一标识实体的属性集称为码。
用实体名及其属性名集合来抽象和刻画同类实体称为实体型（结构，类，class）
同一类型实体的集合称为实体集

现实世界中事物内部以及事物之间的联系在信息世界中反映为实体（型）内部的联系和实体（型）之间的联系。
实体内部的联系通常是指组成实体的各属性之间的联系，实体之间的联系通常是指不同实体集之间的联系

常见数据模型：层次模型、网状模型、关系模型

数据模型的三要素：数据结构（组成对象及其联系）、数据操纵（数据库各种对象的实例允许执行操作的结合）、数据的完整性约束条件

模式是数据库中全体数据的逻辑结构和特征的描述。模式相对稳定、实例相对变动

三级模式：外模式-模式-内模式
模式：数据库中全体数据的逻辑结构和特征的描述，所有用户的公共数据视图
外模式：数据库用户能够看到和使用的描述，数据库用户的数据视图，多为模式的子集
内模式：唯一的存储模式，数据在数据库内部的组织方式

两级映像：外模式/模式，模式/内模式
外模式/模式连接应用和模式，模式修改时只需修改映像，保持应用不变————逻辑独立性
模式/内模式连接模式和内模式，内模式（存储结构）改变时只需修改镜像，模式和应用不变————物理独立性

第二章

关系的三种类型：基本关系、查询表、视图表
关系模式是关系的结构, 关系是关系模式在某一时刻的数据
在一个给定的应用领域中，所有关系的集合构成一个关系数据库

关系数据库的型: 关系数据库模式，是对关系数据库的描述
关系数据库的值: 关系模式在某一时刻对应的关系的集合，通常称为关系数据库

5种基本操作：选择、投影、并、差、笛卡尔积
操作的对象和结果都是集合
关系数据库语言：关系代数、关系演算、SQL

关系三类完整性约束：实体完整性、参照完整性、用户定义完整性

关系代数运算：传统集合运算（并、差、交、笛卡尔积）、专门关系运算（选择、投影、连接、除）

第三章

SQL：数据查询、数据操纵、数据定义、数据控制
SQL主要特点：综合统一、高度非过程化、面向集合的操作方式、以同一种语法结构提供多种使用方式、语言简洁
SQL采用集合操作方式

基本表：
本身独立存在的表
SQL中一个关系就对应一个基本表。
一个（或多个）基本表对应一个存储文件，一个表可以带若干索引

存储文件：
逻辑结构组成了关系数据库的模式，物理结构（内模式）对用户是隐蔽的。
用户修改模式，DBMS实现模式到内模式的自动映射与管理。

视图：
从一个或几个基本表通过查询而导出的表（虚表）
数据库中只存放视图的定义，不存放视图对应的数据
用户可以在视图上再定义视图

SQL数据定义功能：模式定义、表定义、视图和索引的定义
一个RDBMS的实例（Instance）中可以建立多个数据库
一个数据库中可以建立多个模式（取决与具体实现）
一个模式下通常包括多个表、视图和索引等数据库对象

顺序文件上的索引
B+树索引
散列（hash）索引
位图（bitmap）索引
特点：
B+树索引具有动态平衡的优点
HASH索引具有查找速度快的特点

DBA/表属户建立索引，用户不必也不能显式地选择索引

数据字典是关系数据库管理系统内部的一组系统表，它记录了数据库中所有定义信息：关系模式定义、视图定义、索引定义、完整性约束定义、各类用户对数据库的操作权限、统计信息等

Mysql：NULL与空字符串''是不同的
Oracle：NULL与空字符串''是等价的

WHERE子句中是不能用聚集函数作为条件表达式
HAVING短语与WHERE子句的区别：WHERE子句作用于基表或视图，从中选择满足条件的元组，HAVING短语作用于组，从中选择满足条件的组。

子查询不能使用ORDER BY子句（通常把order by 移到最外层）

临时禁止约束：
SET FOREIGN_KEY_CHECKS=0 (禁止)
SET FOREIGN_KEY_CHECKS=1 (恢复)
alter table <table_name> DISABLE/ENABLE constraint <constraint_name>;

有NOT NULL约束条件的不能取空值
加了UNIQUE限制的属性不能取空值
码属性不能取空值

视图的特点：
1.虚表，是从一个或几个基本表（或视图）导出的表
2.只存放视图的定义，不存放视图对应的数据
3.基表中的数据发生变化，从视图中查询出的数据也随之改变

关系数据库管理系统执行CREATE VIEW语句时只是把视图定义存入数据字典，并不执行其中的SELECT语句。
在对视图查询时，按视图的定义从基本表中将数据查出。

若一个视图是从单个基本表导出的，并且只是去掉了基本表的某些行和某些列，但保留了主码，我们称这类视图为行列子集视图。
有些情况下，视图消解法不能生成正确的查询。

多对一视图不能更新：
如果视图的select目标列包含聚集函数，则不能更新
如果视图的select子句使用了unique或distinct，则不能更新
如果视图中包括了group by子句，则不能更新
如果视图中包括经算术表达式计算出来的列，则不能更新
如果视图是由单个表的列构成，但没有包括主键，则不能更新

视图能够简化用户的操作
视图使用户能以多种角度看待同一数据
视图对重构数据库提供了一定程度的逻辑独立性

视图能够对机密数据提供安全保护
适当的利用视图可以更清晰的表达查询

第四章

数据库的安全性是指保护数据库以防止不合法使用所造成的数据泄露、更改或破坏 。
系统安全保护措施是否有效是数据库系统主要的评价指标之一。
数据库管理系统还要进行存取控制，只允许用户执行合法操作
身份鉴别：静态口令/动态口令

- TCSEC/TDI安全级别划分

安全级别	定　　义
A1	验证设计（Verified Design）
B3	安全域（Security Domains）
B2	结构化保护（Structural Protection）
B1	标记安全保护（Labeled Security Protection）
C2	受控的存取保护（Controlled Access Protection）
C1	自主安全保护（Discretionary Security Protection）
D	最小保护（Minimal Protection）

- 计算机信息系统安全保护等级划分准则
  - GB 17859-1999, 1999年发布，2001.1.1实施
  - 第一级：用户自主保护级；
  - 第二级：系统审计保护级；
  - 第三级：安全标记保护级；
  - 第四级：结构化保护级；
  - 第五级：访问验证保护级。
- 等级保护5级大致与C1,C2,B1,B2,B3相对应
  - 掐头(D)去尾(A)

用户权限定义和合法权检查机制一起组成了数据库管理系统的存取控制子系统

DAC：
C2级
用户对不同的数据对象有不同的存取权限
不同的用户对同一对象也有不同的权限
用户还可将其拥有的存取权限转授给其他用户

MAC：
B1级
每一个数据对象被标以一定的密级
每一个用户也被授予某一个级别的许可证
对于任意一个对象，只有具有合法许可证的用户才可以存取

用户权限组成：数据对象+操作类型
定义存取权限称为授权
存取控制对象：数据库模式（模式、基本表、视图、索引）+数据

GRANT发出者：DBA、表owner、有grant权限的人

只有系统的超级用户才有权创建一个新的数据库用户
新创建的数据库用户有三种权限：CONNECT、RESOURCE和DBA
CONNECT：连接
RESOURCE：建立基本表和视图
DBA：超级用户，建立基本表、视图、模式、新用户、授予权限

数据库角色：被命名的一组与数据库操作相关的权限
角色是权限的集合

可以为一组具有相同权限的用户创建一个角色

RBAC三个安全原则：最小权限、责任分离、数据抽象

MAC：
主体(Subject)是系统中的活动实体
数据库管理系统所管理的实际用户
代表用户的各进程
客体(Object)是系统中的被动实体，受主体操纵
文件、基本表、索引、视图
密级：TS>=S>=C>=P
下读（高主体读低客体）
上写（低主体写高客体）

第五章

数据库完整性：数据正确性&&数据相容性
完整性维护：
1.提供定义完整性约束条件的机制
2.提供完整性检查的方法
3.违约处理 NO ACTION/CASCADE/SET NULL

四种破坏参照完整性的情况：加新表、改新表、删旧表、改旧表
新表不让动，旧表都行

CREATE TABLE时定义属性上的约束条件
列值非空（NOT NULL）
列值唯一（UNIQUE）
检查列值是否满足一个条件表达式（CHECK）

触发器（Trigger）是用户定义在关系表上的一类由事件驱动的特殊过程
触发器保存在数据库服务器中
任何用户对表的增、删、改操作均由服务器自动激活相应的触发器
触发器可以实施更为复杂的检查和操作，具有更精细和更强大的数据控制能力

表的拥有者才可以在表上创建触发器
触发器名可以包含模式名，也可以不包含模式名
同一模式下，触发器名必须是唯一的
触发器名和表名必须在同一模式下

触发器只能定义在基本表上，不能定义在视图上
当基本表的数据发生变化时，将激活定义在该表上相应触发事件的触发器

第七章

表达计算机世界的模型称数据模型
表达信息世界的模型称概念数据模型， 简称概念模型
数据库设计是指对于一个给定的应用环境，构造（分析、设计）优化的数据库逻辑模式和物理结构，并据此建立数据库及其应用系统，使之能够有效地存储和管理数据，满足各种用户应用需求，包括信息管理要求和数据操作要求。

数据库设计的目标是为用户和各种应用系统提供一个信息基础设施和高效率的运行环境

数据库设计分6个阶段
需求分析
概念结构设计
逻辑结构设计
物理结构设计
数据库实施
数据库运行和维护
需求分析和概念设计独立于任何数据库管理系统
逻辑设计和物理设计与选用的数据库管理系统密切相关

需求分析是设计数据库的起点
分成2大类
功能需求：重点关注
非功能需求，或者技术需求：了解，后期关注
调查的重点是“数据”和“处理

数据字典是关于数据库中数据的描述，即元数据，不是数据本身
数据字典在需求分析阶段建立，在数据库设计过程中不断修改、充实、完善
数据字典是进行详细的数据收集和数据分析所获得的主要结果

数据字典的内容
数据项
数据结构
数据流
数据存储
处理过程
数据项是数据的最小组成单位
若干个数据项可以组成一个数据结构
数据字典通过对数据项和数据结构的定义来描述数据流、数据存储的逻辑内容

数据项是不可再分的数据单位
数据结构反映了数据之间的组合关系。
一个数据结构可以由若干个数据项组成，也可以由若干个数据结构组成，或由若干个数据项和数据结构混合组成。
数据流是数据结构在系统内传输的路径。
数据存储是数据结构停留或保存的地方，也是数据流来源和去向之一。

概念模型反映现实世界
同一个实体集内的各实体之间也可以存在一对一、一对多、多对多的联系。

数据库在物理设备上的存储结构与存取方法称为数据库的物理结构
数据库物理设计的步骤
确定数据库的物理结构
在关系数据库中主要指存取方法和存储结构;
对物理结构进行评价
评价的重点是时间和空间效率

关系数据库物理设计的内容
为关系模式选择存取方法（建立存取路径）
设计关系、索引等数据库文件的物理存储结构

数据库管理系统常用存取方法

- B+树索引存取方法
- Hash索引存取方法
- 聚簇存取方法

B+索引选择一般规则：查询条件、聚集函数参数、连接条件、排序
Hash索引选取一般规则：等值连接、等值比较
关系大小可变可不变，可变需满足动态Hash存取，Hash字段具有良好的离散性

为了提高某个属性（或属性组）的查询速度，把这个或这些属性（称为聚簇码）上具有相同值的元组集中存放在连续的物理块中称为聚簇。
聚簇索引
建立聚簇索引后，基表中数据也需要按指定的聚簇属性值的升序或降序存放。也即聚簇索引的索引项顺序与表中元组的物理顺序一致。
在一个基本表上最多只能建立一个聚簇索引
聚簇索引的适用条件
很少对基表进行增删操作
很少对其中的变长列进行修改操作
聚簇可以节省存储空间，但只能提高某些特定应用的性能，建立与维护聚簇的开销相当大
当SQL语句中包含有与聚簇码有关的ORDER BY, GROUP BY, UNION, DISTINCT等子句或短语时，使用聚簇特别有利

确定数据库物理结构主要指确定数据的存放位置和存储结构
三要素：存取时间、存储空间利用率和维护代价

第八章

嵌入式SQL：预编译 过程化SQL基本结构是块
定义的变量、常量等只能在该基本块中使用
当基本块执行结束时，定义就不再存在
命名块

编译后保存在数据库中，可以被反复调用，运行速度较快，过程和函数是命名块
匿名块
每次执行时都要进行编译，它不能被存储到数据库中，也不能在其他过程化SQL块中调用

存储过程：由过程化SQL语句书写的过程
函数和存储过程的异同
同：都是持久性存储模块
异：函数必须指定返回的类型

第九章

查询优化分类：
代数优化：指关系代数表达式的优化
物理优化：指存取路径和底层操作算法的选择

关系数据库管理系统查询处理阶段：

- 查询分析
- 查询检查
- 查询优化
- 查询执行

查询检查的任务
合法性检查
视图转换
安全性检查
完整性初步检查

选择操作：全表扫描、索引扫描
连接操作：
（1）嵌套循环算法(nested loop join)
（2）排序-合并算法(sort-merge join 或merge join)
（3）索引连接(index join)算法
（4）Hash Join算法

类别	Nested Loop	Hash Join	Merge Join
使用条件	任何条件	等值连接（=）	等值或非等值连接(>，<，=，>=，<=)，‘<>’除外
相关资源	CPU、磁盘I/O	内存、临时空间	内存、临时空间
特点	当有高选择性索引或进行限制性搜索时效率比较高，能够快速返回第一次的搜索结果。	当缺乏索引或者索引条件模糊时，Hash Join比Nested Loop有效。通常比Merge Join快。在数据仓库环境下，如果表的纪录数多，效率高。	当缺乏索引或者索引条件模糊时，Merge Join比Nested Loop有效。非等值连接时，Merge Join比Hash Join更有效
缺点	当索引丢失或者查询条件限制不够时，效率很低；当表的纪录数多时，效率低。	为建立哈希表，需要大量内存。第一次的结果返回较慢。	所有的表都需要排序。它为最优化的吞吐量而设计，并且在结果没有全部找到前不返回数据。

集中式数据库I/O代价是最主要的 分布式数据库总代价=I/O代价+CPU代价+内存代价 + 通信代价

有选择和连接操作时，先做选择操作

代数优化策略：通过对关系代数表达式的等价变换来提高查询效率
连接、笛卡尔积可以交换+结合
多重投影等同于最小子集投影
多重选择等于交选择，选择条件可以合并
选择投影可以交换
选择与笛卡尔积按照属性分配交换
选择和并分配、选择与差分配、选择与自然连接分配
投影与笛卡尔积、并分配
投影与差没有分配律

启发式规则：
1.选择运算应尽可能先做
2.把投影运算和选择运算同时进行
3.把投影同其前或其后的双目运算结合起来
4.把某些选择同在它前面要执行的笛卡尔积结合起来成为一个连接运算
5.找出公共子表达式

物理优化就是要选择高效合理的操作算法或存取路径
基于规则的启发式优化、基于代价估算的优化

第十章

事务(Transaction)是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。
事务是恢复和并发控制的基本单位
事务的ACID特性：
原子性（Atomicity）
一致性（Consistency）
隔离性（Isolation）
持续性（Durability ）

数据库管理系统必须具有把数据库从错误状态恢复到某一已知的正确状态

故障种类：
事务内部的故障
2.系统故障
3.介质故障
4.计算机病毒

事务内部更多的故障是非预期的，是不能由应用程序处理的。恢复：UNDO
运算溢出
并发事务发生死锁而被选中撤销该事务
违反了某些完整性限制而被终止等

系统故障称为软故障，是指造成系统停止运转的任何事件，使得系统要重新启动。
不破坏数据库、内存中数据库缓冲区的信息全部丢失

恢复操作的基本原理：冗余
利用存储在系统别处的冗余数据来重建数据库中已被破坏或不正确的那部分数据

如何建立冗余数据
数据转储（backup）
登记日志文件（logging）

重装后备副本只能将数据库恢复到转储时的状态
要想恢复到故障发生时的状态，必须重新运行自转储以后的所有更新事务

静态转储
在系统中无运行事务时进行
开始时数据库处于一致性状态
转储期间不允许对数据库的任何存取、修改活动
得到的一定是一个数据一致性的副本

动态转储
后备副本加上日志文件

转储操作与用户事务并发进行
转储期间允许对数据库进行存取或修改

海量转储: 每次转储全部数据库
增量转储: 只转储上次转储后更新过的数据

日志文件(log file)是用来记录事务对数据库的更新操作的文件
格式：以记录为单位、以数据块为单位
进行事务故障恢复
进行系统故障恢复
协助后备副本进行介质故障恢复

静态转储方式中，也可以建立日志文件

日志文件原则：
登记的次序严格按并发事务执行的时间次序
必须先写日志文件，后写数据库

恢复策略：
事务故障：UNDO，系统自动完成，反向扫描文件日志+逆操作

系统故障：Undo 故障发生时未完成的事务，Redo 已完成的事务，由系统在重新启动时自动完成，正向扫描日志文件+建立队列

介质故障：重装数据库、重做已完成的事务，装入最新的后备数据库副本与有关的日志文件副本。
数据库管理员重装+给出指令，恢复由系统来做

在日志文件中增加检查点记录（checkpoint）
检查点记录的内容
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
重新开始文件的内容
记录各个检查点记录在日志文件中的地址

周期性地执行如下操作：建立检查点，保存数据库状态
步骤：
1.日志缓冲区写入磁盘日志文件
2.日志文件写入检查点记录
3.数据缓冲区写入数据库
4.检查点记录地址写入重新开始文件

恢复子系统可以定期或不定期地建立检查点
步骤：正向扫描，建立UNDOLIST和REDOLIST记录，最后UNDO没完成的，REDO已提交的

第十一章

在单处理机系统中，事务的并行执行是这些并行事务的并行操作轮流交叉运行
并行事务并没有真正地并行运行，但能够减少处理机的空闲时间，提高系统的效率
多处理机系统中，每个处理机可以运行一个事务，多个处理机可以同时运行多个事务，实现多个事务真正的并行运行

事务并发执行带来的问题
会产生多个事务同时存取同一数据的情况
可能会存取和存储不正确的数据，破坏事务隔离性和数据库的一致性

事务是并发控制的基本单位，也是数据恢复的基本单位
并发控制机制的任务
对并发操作进行正确调度
保证事务的隔离性
保证数据库的一致性

数据恢复保证原子性、一致性、持续性
并发控制保证隔离性、一致性

并发操作带来的数据不一致性
1.丢失修改（Lost Update）：两边一起改，只改变了一次
2.不可重复读（Non-repeatable Read）：一个读完另一个修改，再读发现对不上
3.读“脏”数据（Dirty Read）：一个修改完另一个读，结果修改被撤销了

数据不一致性：由于并发操作破坏了事务的隔离性

并发控制的主要技术
封锁(Locking)
时间戳(Timestamp)
乐观控制法
多版本并发控制(MVCC)

封锁就是事务T在对某个数据对象（例如表、记录等）操作之前，先向系统发出请求，对其加锁
在事务T释放它的锁之前，其它的事务不能更新此数据对象。

基本封锁类型
排它锁（Exclusive Locks，简记为X锁）
共享锁（Share Locks，简记为S锁）

X锁（我要改，你们谁都不许读不许改）
若事务T对数据对象A加上X锁，则只允许T读取和修改A，其它任何事务都不能再对A加任何类型的锁，直到T释放A上的锁
其他事务在T释放A上的锁之前不能再读取和修改A

S锁（都可以读，但是我不去锁你们都只能读不能更改）
事务T对数据对象A加上S锁，则事务T可以读A但不能修改A，其它事务只能再对A加S锁，而不

能加X锁
保证其他事务可以读A，但在T释放A上的S锁之前不能对A做任何修改

封锁协议：
何时申请X锁或S锁
持锁时间
何时释放

一级：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（COMMIT/ROLLBACK）才释放。
可防止丢失修改，并保证事务T是可恢复的。
如果仅仅是读数据不对其进行修改，是不需要加锁的，所以它不能保证可重复读和不读“脏”数据

二级：一级封锁协议+事务T在读取数据R之前必须先对其加S锁，读完后即可释放S锁
可以防止丢失修改和读“脏”数据。
由于读完数据后即可释放S锁，所以它不能保证可重复读。

三级：一级封锁协议+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。
三级封锁协议可防止丢失修改、读脏数据和不可重复读。

活锁：多事务同时封锁，解锁后随即分配，某个事务出现长时间等待
活锁解决：先来先服务
死锁：多事务交叉封锁，T1等待T2的锁，T2等待T1的锁（互斥、占有并等待、非抢占、循环等待）

预防死锁：
（1）一次封锁法：每个事务必须一次将所有要使用的数据全部加锁（一次性锁完，别分开申请）
问题：降低系统并发度，且封锁对象是动态的，不能事先确定

（2）顺序封锁法：预先对数据对象规定一个封锁顺序，所有事务都按这个顺序实行封锁。
问题：维护顺序成本高，且封锁对象是动态的，难以按照规定顺序执行

死锁的诊断：
（1）超时法：一个事务的等待时间超过了规定的时限，就认为发生了死锁
优点：实现简单
缺点：有可能误判死锁，时限若设置得太长，死锁发生后不能及时发现

（2）等待图法
事务等待图动态反映所有事务的等待情况
如果发现图中存在回路，则表示系统中出现了死锁
解决：选择一个处理死锁代价最小的事务，将其撤消，释放资源

可串行化：并发执行结果和某一种串行执行结果相同————可串行化
并发执行只有是可串行化调度才是正确的

冲突：调度中一对连续的动作，它们满足：如果它们的顺序交换，那么涉及的事务中至少有一个事务的行为会改变。
有冲突的两个操作是不能交换次序的，没有冲突的两个事务是可交换的
可交换：两个事务读一个元素，两个事务操作不同元素
保证冲突操作的次序不变的情况下，通过交换两个事务不冲突操作的次序得到另一个可串行的调度————冲突可串行化

并发调度的正确性 ⊂ 可串行性冲突 ⊂ 可串行性

可串行性保障：两段锁协议
在对任何数据进行读、写操作之前，事务首先要获得对该数据的封锁
在释放一个封锁之后，事务不再申请和获得任何其他封锁
事务遵守两段锁协议是可串行化调度的充分条件，而不是必要条件。

封锁的对象：逻辑单元，物理单元。对象的大小称为封锁粒度
粒度与并发性（成本）成反比
多粒度树：
显式封锁: 直接加到数据对象上的封锁
隐式封锁:是该数据对象没有独立加锁，是由于其上级结点加锁而使该数据对象加上了锁
对某个数据对象加锁，要检查：该对象、所有上级节点、所有下级结点

意向锁：提高对某个数据对象加锁时系统的检查效率
对一个结点加意向锁，则说明该结点的下层结点正在被加锁
对任一结点加基本锁，必须先对它的上层结点加意向锁
分类：IS（子节点加S锁）、IX（子节点加X锁）、SIX（读该对象，且要修改某些子节点）

T_1 \ T_2	S	X	IS	IX	SIX	–
S	Y	N	Y	N	N	Y
X	N	N	N	N	N	Y
IS	Y	N	Y	Y	Y	Y
IX	N	N	Y	Y	N	Y
SIX	N	N	Y	N	N	Y
–	Y	Y	Y	Y	Y	Y

Y=Yes，表示相容的请求　　N=No，表示不相容的请求

(a) 数据锁的相容矩阵

强度：X>SIX>S、IX>IS
封锁自上而下，解锁自下而上