Chapter 1: Introduction

1. 物理数据独立性：在不修改逻辑范式的情况下修改物理范式；逻辑数据独立性：在不修改用户视图范式的情况下…
2. { Storage Manager { File/Buffer Manager / Authoration/Integrity Manager
   Query Processor { DDL Interpreter / DML Compiler：生成执行计划/优化

Chapter 2: Relational Model

1. Attributes 属性 ① Domain：属性取值范围的集合 ② NULL是每个域的成员 ③ INF：属性值是原子/不可分的(多值/复合属性不是原子的) 2. 关系中的元组是无序、不重复、属性值是原子的 3. 主键的值必须唯一且非空 4. 外键指向另一表的主键，其值可以为NULL

二、关系代数基本运算 1. 选择：$\sigma_P(r)$ 2. 投影：$\Pi_{A,B,\dots}(r)$ 3. Union并：$r \cup s$ 4. Difference：$r-s=\{t \mid t\in r \wedge t\notin s\}$ 5. 笛卡尔积：若r、s含同名属性B，则 $r\times s$ 含r.B和s.B 6. Rename：$\rho_{x(A_1,A_2,\dots)}(E)$

三、关系代数衍外运算 1. Intersection $r\cap s = r-(r-s)$ 2. 自然连接 3. θ-Join：$r\bowtie_\theta s=\sigma_\theta(r\times s)$ 4. 除 Division $r\div s=\{t \mid t\in\Pi_{R-S}(r)\wedge[\forall u\in s, 有\ tu\in r]\}$ $r\div s=\Pi_{R-S}(r)-\Pi_{R-S}[(\Pi_{R-S}(r))\times s-\Pi_{R-S,S}(r)]$ 5. Assignment 赋值：← 6. Aggregate Function：
eg. branch-name $\mathcal{G}$ sum(balance) as sum (account)

四、总结 1. 并、差、交的两关系属性的数量和域需相同 2. 运算优先级：Project→Select→笛卡尔积→Join、Division→交→并、差 3. 找最大元素：$\Pi_{balance}(account)-\Pi_{account.balance}(\sigma_{account.balance<d.balance}(account\times\rho_d(account)))$ 4. 数据库操作的实现 ① 删：$r\leftarrow r-E$ ② 插 $r\leftarrow r\cup E$

Chapter 3-4 SQL:

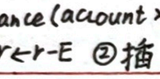
一、SELECT语句 ① SELECT默认不去重，要去重用SELECT DISTINCT ② WHERE后可接AND、OR、BETWEEN…AND，可用NOT取反，判断NULL用IS NULL、IS NOT NULL ③ 在SELECT和FROM中，用AS分别对列和表重命名 ④ 用ORDER BY为输出排序，asc递增(默认)，desc递减 ⑤ 用INTERSECT、UNION、EXCEPT作交、并、差，默认去重，可用INTERSECT/UNION/EXCEPT ALL保留

二、使用聚合函数 ① 默认不去重，可用DISTINCT ② 处理NULL：count不会忽略属性为NULL的元组，其它聚合函数则忽略。

```
SELECT branch-name, count(distinct customer_name)
as num FROM depositor D, account A WHERE
D.account_num=A.account_num GROUP BY branch-
name HAVING count(distinct customer_name)>5
ORDER BY branch-name;
```

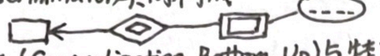
执行顺序：FROM→WHERE→GROUP→HAVING→SELECT→DISTINCT→ORDER BY

Chapter 5: Entity-Relational Model

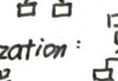
1. 联系集的度 Degree：联系集关联的实体集数目
2. 实体集/联系集都可具有属性 3. Recursive Relationship set 

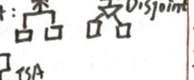
4. 可通过创建新的实体集，将任何非二元关系转化为二元关系 5. 关联数目限制 ① Mapping Cardinalities：限制每个实体参与关联的上限(1或多) ② Total/Partial Participation：下限(0或1)
6. Derived属性：虚线()，多值属性：◎或{}

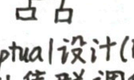
二、Weak Entity Set
1. Identifying Entity set的主键和弱实体集的 Partial Key(Discriminator)共同构成

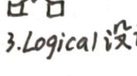


三、泛化 (Generalization, Bottom-Up)与特殊化 Top-Down)
1. 不相交(Disjoint)与可重叠(Overlapping)：设父类为"人"，子类为"老师""学生"，若同一人不能既是老师又是学生，则为不相交 2. Total/Partial Generalization：若要求父类必须特化为子类，则为完全泛化

Overlapping： 

Disjoint： 

Total Generalization： 



四、设计流程
1. 需求分析 2. Conceptual设计(E-R) 3. Logical设计(表) 4. Physical设计(索引、集群、调优)

Chapter 6: Relational Database Design

一、函数依赖(FD) 1. 定义：$\alpha\to\beta$：给出α可唯一确定β。若 $\beta\subseteq\alpha$ 则称其平凡(Trivial) 2. 函数依赖集的闭包：(F^+)若有n个属性，则F^+最多有$2^n\times 2^n$个元素；3. 求F^+的Armstrong's Axioms ① Reflexivity：若 $\beta\subseteq\alpha$，则 $\alpha\to\beta$，②Argumentation：若$\alpha\to\beta$，则$\gamma\alpha\to\gamma\beta$，$\delta\alpha\to\beta$ ③Transitivity：若$\alpha\to\beta$，$\beta\to\gamma$，则$\alpha\to\gamma$ ④Union：若$\alpha\to\beta$，$\alpha\to\delta$，则$\alpha\to\beta\delta$ ⑤Decomposition：若$\alpha\to\beta\delta$，则$\alpha\to\beta$，$\alpha\to\delta$ ⑥Pseudotransitivity：若$\alpha\to\beta$，$\delta\beta\to\delta$，则$\alpha\delta\to\delta$ 4. 属性集的闭包(α^+)：若$R\subseteq\alpha^+$，则α是superkey.(可画图寻找超键) 5. 正则覆盖(Canonical cover)(F_c)：清除了冗余项的函数依赖集。①特点：不存在无关属性(Extraneous Attribute)，每条函数依赖的左侧部分唯一 ②示例：{A→BC, B→C}的正则覆盖是{A→B, B→C} 6. 求正则覆盖的算法 ①先把形如$\alpha\to\beta_1,\alpha\to\beta_2$的项合并为$\alpha\to\beta_1\beta_2$ ②对每个函数依赖，检查$\alpha\to\beta$的α或β中是否可删去无关属性 ③重复做①②两步

二、关系范式的分解 1. 任何关系都可无损连接分解为BCNF或3NF，BCNF可能破坏依赖保持，3NF一定是依赖保持的，但引入冗余 2. Lossless-Join Decomposition ①定义：要求分解后的两表经⋈连接可恢复为原表 ②判定：$\{R_1\cap R_2\}\to R_1$或$\{R_1\cap R_2\}\to R_2$ 3. Dependency Preservation：要求分解后各关系的函数依赖集的并的闭包与原关系的函数依赖集的闭包相同

三、BCNF 1. 定义：要求F^+中每个$\alpha\to\beta$至少满足一条 ①α→β是平凡的 ②α是R的超键 2. 只有2个属性，一定是BCNF 3. BCNF分解算法：对每个不满足BCNF的关系Ri，考察其违背了BCNF的那条函数依赖$\alpha\to\beta$，将Ri分解为两个新关系(α,β)，(Ri-β)并塞回去

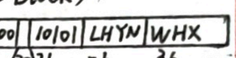
四、3NF 1. 定义：要求F^+中每个$\alpha\to\beta$至少满足一条 ①$\alpha\to\beta$是平凡的 ②α是R的超键 ③(β-α)中的每个属性被R的超键所包含 2. 3NF分解算法：①将F_c中的每个$\alpha\to\beta$分解为子模式(α,β)。②完成①后，若任一子模式都不包含一个完整的Candidate key，则再新建一个子模式，包含R的Candidate key 五、多值依赖与第四范式
1. 定义(多值依赖)：$\alpha\twoheadrightarrow\beta$：给定α，则β的值可以取多个，但这些值和关系中其它属性无关，由于这种独立性，必须在表中存储它们彼此独立组合的所有情况

2. $\alpha\twoheadrightarrow\beta\Leftrightarrow\alpha\twoheadrightarrow R-\beta-\alpha$ 3. 函数依赖是多值依赖的特例，$X\to Y\Rightarrow X\twoheadrightarrow Y$ 4. 平凡的多值依赖：$\beta\subseteq\alpha$或$\alpha\cup\beta=R$ 5. 4NF的定义：D^+中每个$\alpha\twoheadrightarrow\beta$至少满足一条：①$\alpha\twoheadrightarrow\beta$是平凡的 ②α是R的超键。6. 4NF分解算法：同BCNF
7. 严格程度：4NF>BCNF>3NF，函数依赖>多值依赖

Chapter 7: Storage & File Structure

一、存储 1. 层次：①Primary Storage(Cache, Memory) ②Secondary/Online(Flash, 磁盘) ③Tertiary/offline(光盘, Tapes) 2. 磁盘结构：Platter盘片，Track磁道，Sector扇区，Cylinder柱面(所有盘片相同半径的磁道的集合)，Spindle转轴 3. Access Time = Seek Time + Rotational Latency(盘片旋转一圈用时的$\frac{1}{2}$)，表示从I/O请求发出到数据开始传输的时间 4. Data Transfer Rate：MB/s 内侧慢于外侧 5. Buffer维护一个<帧号,页号>的表，替换策略包括LRU、MRU(Most Recently Used)，6. Toss-Immediate 表示对缓冲区内每个块，用后立即丢弃

二、文件组织 1. 定长记录的表示 ① Free List：由被删除记录组成 ② File Header：存放Free List的第一个指针
2. 变长记录的定长表示 ① Reserve Space(要求最大长度已知) ②通过指针表示(Anchor/Overflow Block)
3. 变长记录的表示：



①定长部分：存储定长属性实际值、变长属性的位置、大小 ②BitMap：0表示非空
4. 变长记录在块中的存储 Slotted-Page Structure：包含块头，记录 ①块中记录项的数量 ②指向自由空间末尾的指针 ③一个数组，记录每条记录的位置和大小。

Chapter 8: Indexing (分为Ordered Index / Hash Index)

一、Ordered Index(顺序索引) 1. 索引条目按search-key的排序顺序存储 ①Sequentially Ordered File：文件中的记录按search-key的排序顺序存储 ②Indexed Sequential File：包含主索引的顺序排序文件 2. 主索引与辅助索引 ① Primary Index：又称Clustering Index，与对应的数据文件本身排列顺序相同的索引，既可是稀疏(Sparse)又可为稠密(Dense)索引，主索引的search-key通常是主键，但不一定 ②Secondary Index：又称Non-Clustering Index，用于对非主键的属性进行查询，只能用稠密索引，使用bucket或在search-key中加入record-identifier解决重复项问题 3. 多级索引：既可用于稠密索引，也可用于稀疏索引，Outer Index是Inner Index的稀疏索引 二、B+树索引 1. 

左图 n=4，有3个值
2. 树的高度=从根到叶的路径长，根的深度为0，叶的高度为0。B+树是平衡树 3. 既非根又非叶的节点有$\lceil\frac{n}{2}\rceil$至n个儿子；若叶子节点不是根，则叶子节点具有$\lceil\frac{n-1}{2}\rceil$至(n-1)个值。若叶子节点同时也是根节点，则该节点具有0~(n-1)个值 4. 度为n的B+树包含K个search-key，则树高不大于$\lceil\log_{\lceil n/2\rceil}(K)\rceil$ 5. 记录的插入：向已满的节点插入，先进行节点分裂，再进行数据插入，其中原节点的前$\lceil\frac{n}{2}\rceil$个元素不动，6. 记录的删除：Merge Siblings/Redistribute Pointers

三、Write-Optimized Index (LSM: Log Structured Merge Tree) 优点：插入为顺序I/O；叶子是满的避免空间浪费；与B+树相比，插入的I/O数更少 缺点：查询时需搜索多棵树；每一层级的全部元素被访问多次

Chapter 9: Query Processing

一、查询代价：Seek Time + Block Transfer Time (t_S+t_T)

二、选择操作
1. A1 Linear Search 块传输：b_r(存储关系r所用的块数)，寻道：1 (若检索条件是主键，则平均块传输$\frac{b_r}{2}$，寻道$\frac{1}{2}$)
2. A2 Binary Search 块传输：$\lceil\log_2(b_r)\rceil+\lceil sc(A,r)/f_r\rceil-1$，其中sc(A,r)是满足选择条件的记录数，$f_r$是每个块存储的记录数，-1表示第一个块已在$\lceil\log_2(b_r)\rceil$中完成读取，无需重复记数。寻道：$\lceil\log_2(b_r)\rceil$，仅限等值比较
3. A3 Primary Index, Equality on key/Nonkey。块传输：h_i+b(b为满足选择条件的块的总数，要求连续)，寻道：h_i+1(h_i为索引树的高)
4. A5 Secondary Index, Equality on Nonkey：块传输：h_i+n(n为满足查询条件的元组数，且假设在不同块上)，寻道：h_i+n[进行比较的选择算法] 5. A6 Primary Index, Comparison ①大于：用索引找到第一个A≥v的元组，从此开始顺序扫描 ②小于：不用索引，从头顺序扫描，直到第一个不满足A≤v的元组 6. A7 Secondary Index, Comparison：①大于：用索引找到第一个A≥v的指针，从此开始对B+树的叶子进行扫描，获得指针并访问，②小于：略
7. A8 Conjunction Selection Using one Index：要进行$\sigma_{\theta_1\wedge\theta_2\wedge\dots}(r)$的选择，先执行代价最小的$\theta_i$，存选择结果存入内存，然后重复该过程 8. A10 Conjunction Selection By Intersection of Identifiers：对每个选择条件，使用各自的索引获得对应的指针，再取交集并访问(Disjunction同理，取并集)

三、外部排序
设b_r表示待排序的关系所占的磁盘块数，M表示内存中可用的块数，b_b表示一次性可以读写的块数，越大越好
1. 归并轮数：$\lceil\log_{M-1}(b_r/M)\rceil$
2. 块传输(若最后数据不写回磁盘)：$b_r(2\lceil\log_{M-1}(\frac{b_r}{M})\rceil+1)$，(若写回)：$b_r(2\lceil\log_{M-1}(\frac{b_r}{M})+2)$
3. 寻道(若不写回)：$2\lceil b_r/M\rceil+\lceil b_r/b_b\rceil(2\lceil\log_{M-1}(\frac{b_r}{M})\rceil-1)$ (若写回)：$2\lceil b_r/M\rceil+\lceil b_r/b_b\rceil(2\lceil\log_{M-1}(\frac{b_r}{M})\rceil)$

四、连接操作
1. Nested-Loop Join：$r\bowtie s$，r为outer relation，s为inner relation ①算法：对r中每个元组，扫描s的所有元组 ②若较小的关系能完全放入内存，则令较小的为内关系，块传输：b_r+b_s，寻道：2 ③否则，令较小的为外关系，最坏情形是内存只能各放下两个关系的一个块
块传输：$n_r\times n_s+b_r$，寻道：n_r+b_r
2. Blocked Nested-Loop Join：①算法：对r的每个块，扫描s的所有块，对r块中每个元组，扫描s块中每个元组 ②最好情形和代价同上 ③最坏情形同上，代价：块传输：$b_r\times b_s+b_r$，寻道：$2\times b_r$ ④设M表示可用块数，用M-2个块缓存外关系，剩下2个块分别缓存内关系、输出，块传输：$\lceil b_r/(M-2)\rceil\times b_s+b_r$，寻道：$2\lceil b_r/(M-2)\rceil$
3. Indexed Nested-Loop Join：①适用性：等值连接，内关系的连接属性上具有索引 ②选取具有较少元组数的关系作为外关系 ③最坏情形：内存只能放下r的一个块，块传输=寻道=$b_r+n_r\times c$，c为对r的每个元组通过遍

S的索引找到所有匹配元组的对应成本
4. Merge Join: ①适用性: 等值连接. 要连接的两个关系均已按连接属性进行排序 ②块传输: b_r+b_s. 寻道: $\lceil b_r/b_b\rceil+\lceil b_s/b_b\rceil$ ③Hybrid Merge Join: 要求一个关系已排序. 另一关系在连接属性上有辅助索引. 先归并得到新的关系(已排序的关系元组. 未排序的关系指针), 再按指针的物理地址依序访问
5. Hash Join: ①适用性: 等值连接 ②关系S为构造输入(Build Input). r为探测输入(Probe Input). 选较小的关系作为构造输入 ③要求构造输入的每个分组Hsi都能放入内存中. 即分组数量 $n_h\geq\lceil b_s/M\rceil$ ④若 $n_h>M$, 需要Recursively Partitioning(递归分组), 不需要递归分组的条件是 $n_h\leq M$. 即 $M>n_h+1$. 即 $M>\frac{b_s}{n_h}+1$. 即 $M>\sqrt{b_s}$
[代价分析] ①最好情况: 较小的关系能完全放入内存, 无需分组, $n_h=1$, 块传输 b_r+b_s, 寻道 2 ②若不需递归分组, 块传输 $3(b_r+b_s)+4\times n_h$, 寻道: $2(\lceil\frac{b_r}{b_b}\rceil+\lceil\frac{b_s}{b_b}\rceil)+2\times n_h$ (n_h很小, 若不知道可忽略) ③若需要递归分组: 块传输: $2(b_r+b_s)\lceil\log_{M-1}(b_s)-1\rceil+b_r+b_s$
寻道: $2(\lceil b_r/b_b\rceil+\lceil b_s/b_b\rceil)\times\lceil\log_{M-1}(b_s)-1\rceil$

Chapter 10: Query Optimization
一、关系代数转化的等价规则 $\pi_{name}\bowtie_{table}\sigma_{...=...}$ table
1. R1: 与关系SELECT的拆解: $\sigma_{\theta_1\wedge\theta_2}(r)=\sigma_{\theta_1}(\sigma_{\theta_2}(r))$
2. R2: SELECT的交换律: $\sigma_{\theta_1}(\sigma_{\theta_2}(r))=\sigma_{\theta_2}(\sigma_{\theta_1}(r))$
3. R3: 层层包含的投影, 只有最后一个起作用
4. R4: SELECT融入Join条件: $\sigma_{\theta_1}(E_1\bowtie_{\theta_2}E_2)=E_1\bowtie_{\theta_1\wedge\theta_2}E_2$
5. R5: θ连接和自然连接满足交换律(Commutative)
6. R6A: 自然连接满足结合律(Associative)
R6B: θ连接. 有限的结合律(θ_2只含来自E_2和E_3的属性): $(E_1\bowtie_{\theta_1}E_2)\bowtie_{\theta_2\wedge\theta_3}E_3=E_1\bowtie_{\theta_1\wedge\theta_3}(E_2\bowtie_{\theta_2}E_3)$
7. R7: 先连接后选择→先选择后连接(要求θ_1, θ_2各只包含E_1, E_2的属性): $\sigma_{\theta_1\wedge\theta_2}(E_1\bowtie E_2)=(\sigma_{\theta_1}(E_1))\bowtie_{\theta}(\sigma_{\theta_2}(E_2))$
8. R8: 先连接后投影→先投影后连接(L3、L4分别包含了连接条件) $\pi_{L_1\cup L_2}(E_1\bowtie_{\theta}E_2)=(\pi_{L_1\cup L_3}(E_1))\bowtie_{\theta}(\pi_{L_2\cup L_4}(E_2))$
9. R9/10: 集合的交、并满足交换律、结合律
10. R11: SELECT可以跨越交、并、差运算(以差为例: $\sigma_{\theta}(A-B)=\sigma_{\theta}(A)-\sigma_{\theta}(B)$)
11. R12: 全外连接满足交换律. 左、右外连接互相可交换
二、关系代数转化原则 1. 选择、投影应当尽早, 以减小连接关系的大小 2. 多个同类运算同时出现时, 使临时关系的大小越小越好
三、代价估计的统计数据
n_r: 元组数, b_r: 所占块数, f_r: 每块存储的元组数量. ($b_r=\lceil\frac{n_r}{f_r}\rceil$), l_r: 每个元组所占字节数, $V(A,r)$: 属性A出现不同取值的数量.
[选择运算] 1. 对$\sigma_{A=v}(r)$, 估计大小: $n_r/V(A,r)$
2. 对$\sigma_{A\leq v}(r)$, 估计大小 $n_r\times\frac{v-min}{max-min}$
[复杂选择运算] 1. 称关系中任一元组满足条件θ_i的概率为θ_i的选择性. 设满足θ_i的元组数为S_i, 则θ_i的选择性表示为 S_i/n_r 2. 假设各选择条件独立
$\sigma_{\theta_1\wedge\cdots\wedge\theta_n}(r)$ 估计大小: $n_r\times\frac{S_1\times S_2\times\cdots\times S_n}{n_r^n}$
$\sigma_{\theta_1\vee\cdots\vee\theta_n}(r)$ 估计大小: $n_r\times[(1-\frac{S_1}{n_r})\cdots(1-\frac{S_n}{n_r})]$
[连接运算] 1. 笛卡尔乘积: $r\times s$ 估计大小 $n_r\times n_s$
2. $r\bowtie s$ ($R\cup S=\phi$) 估计大小 $n_r\times n_s$
3. $r\bowtie s$ (R∪S是R的键): 估计大小 $\leq n_s$
4. $r\bowtie s$ (R∪S是S指向R的外键): 估计大小 $=n_s$
5. $r\bowtie s$ ($R\cup S=\{A\}$不是键). 估计大小 $min\{\frac{n_r\times n_s}{V(A,s)},\frac{n_r\times n_s}{V(A,r)}\}$
[其它运算] 1. 投影$\pi_A(r)$估计大小: $V(A,r)$
2. 聚合函数运算$G\gamma_A(r)$ 估计大小: $V(G,r)$
3. 集合运算 ①同一关系上的交/并: 改写后估计 $\sigma_{\theta_1}(r)\cup\sigma_{\theta_2}(r)\Rightarrow\sigma_{\theta_1\vee\theta_2}(r)$
② r∪s 估计大小 n_r+n_s. r∩s 估计大小 $min(n_r,n_s)$. r-s 估计大小 n_r
4. 外连接 ① $r\supset\subset s$ 估计大小: $r\bowtie s$的大小$+n_r$
[V的估计] $r\supset\subset s$ " " $+n_r+n_s$
1. 在选择运算中, 估计 $V(A,\sigma_\theta(r))$
①若θ要求A获取一个具体值, 则 $V(A,\sigma_\theta(r))=1$
② " A从一组具体值中选一个 " 具体值数目
③若θ形如 $A>v$ ($A<v$), 则 $V(A,\sigma_\theta(r))=V(A,r)\times S$, S为θ的选择性 ④其它情形, 估计值 $min(V(A,r),n_{\sigma_\theta(r)})$
2. 在连接运算中, 估计 $V(A,r\bowtie s)$
①若A中所有属性来自r, 则 $V(A,r\bowtie s)=min(V(A,r),n_{r\bowtie s})$
②若A包含来自r的属性A1和s的属性A_2. 估计值 $min(V(A_1,r)\cdot V(A_2-A_1,s),V(A1-A2,r)\cdot V(A_2,s),n_{r\bowtie s})$

Chapter 11: Transaction
一、概念 1. 事务终止的条件: COMMIT/ROLLBACK WORK
2. 事务的状态. (Start)→Active→partially commit→committed; Active→failed→aborted
①Partially commit: 事务完成执行, 但还未写回磁盘
②Aborted: 事务失败且完成回滚
3. ACID性质 ①Atomicity: 所有语句要么都执行要么都不 ②Consistence: 显式约束和隐式逻辑保持正确. 由隔离性加以保证(中间状态可不一致) ③Isolation: 每个执行的事务不受其它正在执行事务的干扰 ④Durability
4. Recovery-Management Component实现了原子性和持久性 5. Schedule调度, N个事务有N!种Serial Schedule 6. 合法的并行调度都是可序列化的.
7. 若属于不同的事务的两条指令访问了同一数据, 且至少其一对该数据进行写操作, 则称两指令冲突.
二、可恢复性 1. 若Tj读取了先前Ti写入的数据, 则Ti的Commit需先于Tj的提交, 以保证可恢复性
2. Cascadeless Schedule: 为避免级联回滚, 要求若Tj读取了先前Ti写入的数据, 则Ti的Commit需先于Tj的读取 三、可序列化的检测: 在Precedence Graph中, 无环(Acyclic) $O(n^2)$, n为事务数

Chapter 12: Concurrency Control
一、Lock-Based Protocol
1. 2-Phase Locking Protocol: ①每个事务分2阶段, 所有锁在Growing Phase获取, Shrinking Phase释放 ②对某个调度的全部事务使用2PL, 可保证冲突可序列化 ③2PL无法消除死锁 ④不是所有冲突可序列化的事务都可使用2PL ⑤变体(Strict 2PL): 要求所有事务在COMMIT/ROLLBACK前不得释放其持有的X锁, 可避免级联回滚 ⑥变体(Rigorous 2PL): 要求所有事务在COMMIT/ROLLBACK前不得释放持有的任何锁, 从而可按COMMIT先后顺序进行序列化 ⑦变体(2PL with Lock Conversions): Growing Phase可将S变X锁, Shrinking Phase可将X变S锁.
2. Graph-Based Protocol: ①在数据集上定义偏序关系, 用树来表示 ②每个事务遵循以下规则: I. 只能加X锁 II. Ti的第一个锁可加在任一数据上, 此后的锁只能加在父节点正被Ti锁定的数据上 III. 事务可随时释放锁, 但释放后不能重新获取. ③优点: 避免死锁, 保证冲突可序列化; 缺点: 不保证Recoverbility、无法消除级联回滚. ④不适用2PL的调度可能适用树协议, 不适用树协议的调度可能适用2PL
二、多粒度 MuTiple Granularity
1. 粒度大小的影响: ①细粒度(Fine): 高并发性, 高上锁开销 ②粗粒度(Coarse): 低并发性, 低上锁开销
2. Intention Lock ①IS/IX: 后代存在S/X锁 ②SIX=S+IX, 以该节点为根的子树存在S锁, 后代存在X锁
3. Lock Compatibility Matrix

一个事务的加锁申请 \ 其它事务已加的锁	IS	IX	S	SIX	X
IS	√	√	√	√	×
IX	√	√	×	×	×
S	√	×	√	×	×
SIX	√	×	×	×	×
X	×	×	×	×	×

4. 加锁和解锁流程 对于事务Ti: ①必须先以任意模式锁定根节点 ②对Q加S或IS锁, 要求Q的父节点正被Ti以IX或IS锁定 ③对Q加X、SIX、IX锁, 要求Q的父节点正被Ti以IX或SIX锁定 ④遵守2PL. 自下而上解锁
三、死锁处理
1. Preemption: Tj向已被Ti加锁的项请求锁时
①Wait-Die: 若Tj开始更早, 则Tj继续等待, 否则回滚Tj
②Wound-Wait: 晚 " " Ti
2. 死锁检测: Wait-For Graph. 每个节点代表一个事务, 有向边表示锁等待关系. 若出现环则代表死锁
四、Index-Locking Protocol
1. 为解决Phantom Phenomenon 2. 要求: ①每张表都有索引 ②遵守2PL ③查询必须通过索引进行, 为索引上S锁 ④插入、删除必须通过索引进行, 为索引上X锁

Chapter 13: Recovery
一、概述
1. 故障分类: Transaction Failure. System Crash. Disk Failure
2. 日志: <Ti Start>, <Ti, X, 旧值, 新值>, {<Ti, commit> / <Ti, 旧值> / <Ti, abort>}
3. 在最基本的实现中事务abort不生成形如<Ti, A, 旧值>的补偿日志记录. 所以abort的事务应出现在Undo-List中
4. 先写日志规则(Write Ahead Logging Rule)
①日志按产生的顺序输出到稳定存储器 ②<Ti, Commit>记录必须在Ti提交之前输出到稳定存储器 ③<Ti, Commit>输出到稳定存储器前, 所有与Ti相关的日志记录都必须已输出 ④在块输出前, 所有与块上数据相关的日志必须已输出
二、ARIES数据结构
LSN、PageLSN、Compensation Log Record(UndoNextLSN)、DirtyPageTable(PageID+PageLSN+RecLSN)、Checkpoint-Log Record(DPT+活跃事务列表, 对每个活跃事务标注LastLSN)
三、ARIES执行阶段
[Analysis Pass]
①将RedoLSN设置为DPT中所有页的RecLSN的最小值, RedoPass将从此开始
②将Undo-List初始化为检查点日志记录中的事务列表. ③然后从检查点开始正向扫描, 每遇到更新日志, 就同步更新DPT. 每遇到不属于Undo-List的事务就将其加入Undo-List, 每遇到COMMIT就将该事务移出Undo-List(abort的事务应出现在Undo-List中)
[Redo Pass]
从RedoLSN开始正向扫描, 遇更新记录时.
[Undo Pass]