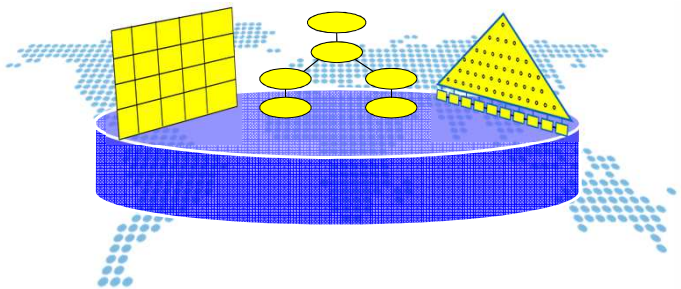
数据库系统复习课

陈世敏
（中科院计算所）



课程相关

- 考核方式：闭卷考试+大作业
  - 实验大作业：40%
    - 实验1：5%，实验2：15%，实验3：20%
    - 实验在各位同学的笔记本电脑（虚拟机）上进行
  - 期末闭卷考试：60%
  - 课堂表现：+3%
- 成绩
  - 数据库系统：闭卷（60%）+大作业（40%）+课堂表现（3%）
  - 数据库系统研讨课：大作业
- 期末考试
  - 2019年1月16日（周三）上午9：00-11:00，阶二2教室

本堂课的组织

- 复习课程的内容
  - 精通：选择、填空、问答
  - 了解：选择、填空
- 答疑

回顾课程安排

周次	内容	
第1周(9月6日)	概述	
第2周(9月13日)	ER模型；关系模型与SQL（1）	
第3周(9月20日)	关系模型与SQL（2）	
第4周(9月27日)	关系模型与SQL（3）	
第5周(9月29日)	关系模型与SQL（4）	
第6周(10月11日)	实验1：数据库安装与使用（独立完成）	5%
第7周(10月18日)	实验2：数据库应用设计（每组3人）	15%
第8周(10月25日)	数据存储与访问路径（1）	
第9周(11月1日)	数据存储与访问路径（2）	
第10周(11月8日)	数据存储与访问路径(3)；查询处理(1)	

回顾课程安排

周次	内容	
第11周(11月15日)	查询处理（2）	
第12周(11月22日)	实验2验收	
第13周(11月29日)	实验3：分析型数据库系统实现（每组3人）	20%
第14周(12月6日)	查询优化	
第15周(12月13日)	事务处理	
第16周(12月20日)	数据仓库；并行/分布式数据库	
第17周(12月27日)	大数据初步	
第18周(1月3日)	复习	
第19周(1月10日)	实验3验收	
第20周(1月16日)	期末考试	60%

1、概述

- 数据库的概念（精通）
- 计算机硬件的发展（了解）
- 数据库系统的发展（了解）

数据库概念

- 什么是关系模型？

数据库概念

- 什么是关系模型？
  - 列(Column)：一个属性，有明确的数据类型
    - 例如：数值类型（e.g., int, double），字符串类型(varchar)，类别类型(有些像程序语言中的enum)
    - 必须是原子类型，不能够再进一步分割，没有内部结构
  - 行(Row)：一个记录（tuple, record）
    - 表是一个记录的集合
    - 记录之间是无序的
  - 通常是一个很瘦长的表
    - 几列到几十列
    - 成千上万行，很大的表可以有亿/兆行

数据库概念

- 表的数学定义？

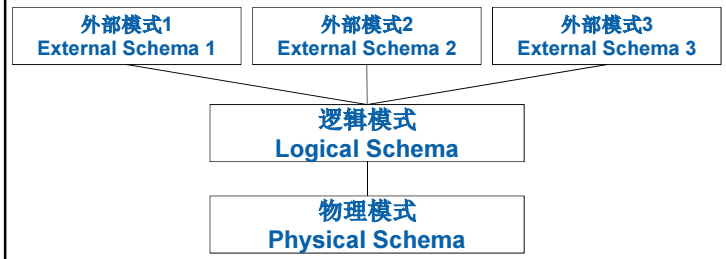
表的数学定义

- K列的表：$\{<t_1, \ldots, t_k> \mid t_1 \in D_1, \ldots, t_k \in D_k\}$
  - 整个表是一个集合$\{<t_1, \ldots, t_k>\}$
  - 集合的每个元素有这样的形式$<t_1, \ldots, t_k>$
    - 第j个部分t_j
  - D_j是第j列的类型所对应的所有可能取值的集合（域）

数据库概念

- 其他概念
  - 数据库（Database）
  - 数据库系统（Database Management System）
  - 关系数据库（Relational Database）
  - 关系数据库系统（Relational Database Management System）
  - 关系模型（Relational Model）
  - Schema vs. Instance
  - 数据独立性

抽象层次



2、ER模型（精通）

- 数据库设计过程
  - 需求分析
  - 概念设计（ER模型）
  - 逻辑设计（ER模型 关系模型）
  - 模式细化（规范化等）
  - 物理设计（物理模式，性能等）
  - 应用与安全设计（定义外部模式等）
- ER模型
  - 实体、实体集、属性、键
  - 联系、联系集、属性
  - 加箭头，加粗线
  - 弱实体、部分键、类层次
- 应用举例+Lab2

3、关系模型与SQL（1）（精通）

- SQL 初步
  - 表的定义、增删改
- 逻辑设计：ER图到关系模型
- 我们在Lab2中练习了，在考试中是必考的

创建表：Create Table

Student

ID	Name	Birthday	Gender	Major	Year	GPA
…	…	…	…	…	…	…

```
create table Student (
    ID integer,
    Name varchar(20),
    Birthday date,
    Gender enum(M, F),
    Major varchar(20),
    Year year,
    GPA float
    );
```

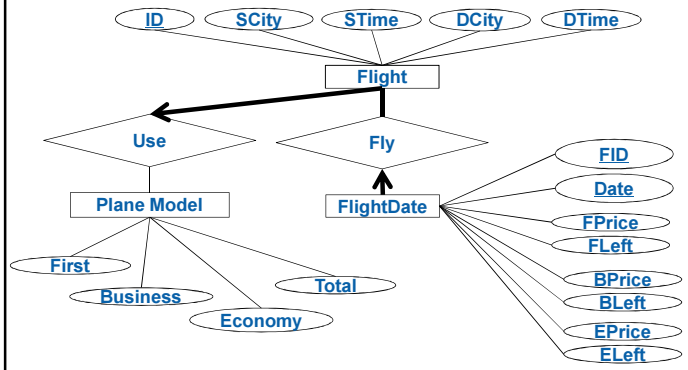
```
create table 表名 (
    列名  类型,
    列名  类型,
    列名  类型,
    ……
    );
```

有些像程序语言中的函数定义

约束条件

- 主键
- 外键
- Check

航班



3、关系模型与SQL（2）（精通）

- 记录的增删改
- 简单的查询
  - 集合操作：并、交、差
  - 选择行或列：选择、投影
  - 两个关系元组之间的操作：笛卡尔积、连接
  - 其它：重命名、除
- 关系代数
- 会写SQL语句，会写对应的关系代数形式

4、关系模型与SQL（3）（精通）

- 丰富的SQL Select功能
  - 扩展关系代数
  - 单个Select语句：aggregation, group by, having, order by
  - 嵌套Select语句：in, exists, unique, op ANY/ALL
- 完整性约束
  - domain, unique, primary key, not null
  - foreign key, 执行
  - check
  - Assertion, trigger（了解）
- 会写SQL语句，会写对应的关系代数形式，会写完整性约束

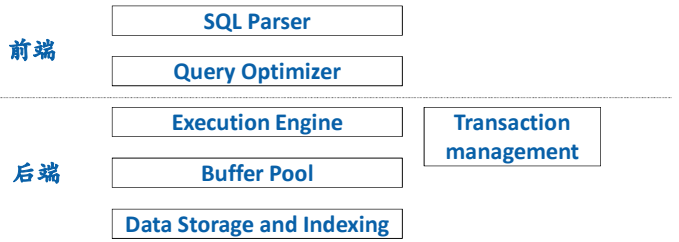
5、关系模型与SQL（4）

- 模式细化：范式
  - 数据冗余的问题（精通）
  - 范式介绍（精通）
  - 函数依赖与2NF,3NF,BCNF（精通）
  - 分解为BCNF（精通）
  - 多值依赖和连接依赖（了解）
- 物理设计：索引的增删（精通）
- 外部模式设计：视图（精通）
  - Grant, revoke

举例

- 假设Sailors, Boats, 和Reserves表
- Joe是Sailors表的创建所有人
- Joe把Sailors的读权限和对读的授权权限授予Art
  (Joe) grant select on *Sailors* to *Art* with grant option;
- Art把Sailors的读权限和对读的授权权限授予Bob
  (Art) grant select on *Sailors* to *Bob* with grant option;
- 一段时间之后，Joe收回Art的读权限和Art进一步的授权给Bob的衍生读权限
  (Joe) revoke select on *Sailors* from *Art* cascade;
  - 这里Bob的读权限也被收回了

RDBMS的系统架构(单机)



6、数据存储与访问路径(1)

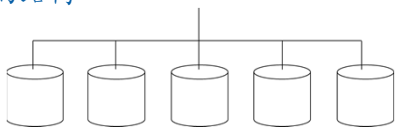
- 数据库系统内部架构概述（了解）
- 数据存储与访问路径概述
  - 存储层次（了解）
  - 存储介质：磁盘（精通）、固态硬盘（了解）等
  - 磁盘阵列（精通）
  - 操作系统支持（了解）
- 磁盘空间管理：工作原理（了解）
- 记录文件格式（精通）
  - 行式文件页结构
  - 行式记录结构
  - 列式文件结构
  - 顺序读和I/O模型
- 缓冲区管理：工作原理，替换算法（精通）

随机访问vs.顺序访问

- 访问时间=$T_{seek} + T_{rotate} + T_{transfer}$
- 随机访问：读取很少量的数据，例如4KB
  - $T_{seek} + T_{rotate}$起主要作用
  - 每次随机访问~10ms，每秒进行100次访问
  - 例如：随机访问4KB数据，那么400KB/s
- 顺序访问：读大量的数据，例如>1MB
  - $T_{transfer}$起主要作用
  - 速度取决于盘片转速、磁介质密度、硬盘接口带宽限制
  - ~100MB/s
- 优化目标：尽量使用顺序访问

RAID

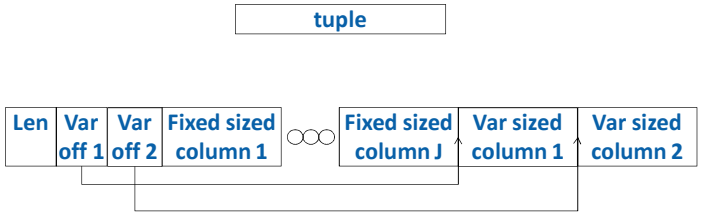
- RAID levels：不同的结构
  - RAID 0
  - RAID 1
  - RAID 5
  - RAID 6（了解）



  - RAID 0+1
  - RAID 10 (即RAID 1+0)
  - RAID 50 (即RAID 5+0)

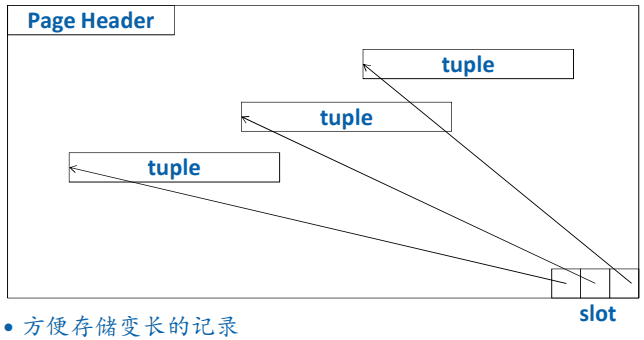
两种RAID的复合

Tuple的结构



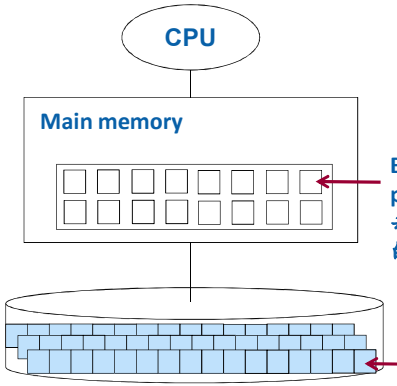
- 举例：有两个变长的列

Page内部结构（Slotted Page）



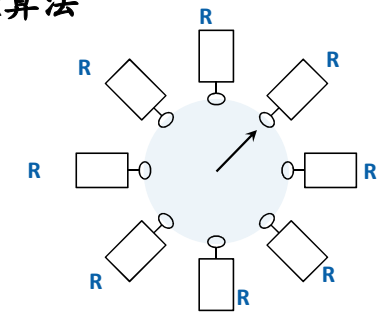
- 方便存储变长的记录
- 记录超出页面大小就需要特殊处理
- Page Header包含：PageID, 校验和, LSN(事务日志序号)等

Buffer Pool的组成



Buffer pool的内存空间分成page大小的单元（frame）
每个frame可以缓冲硬盘中的一个page

Clock算法



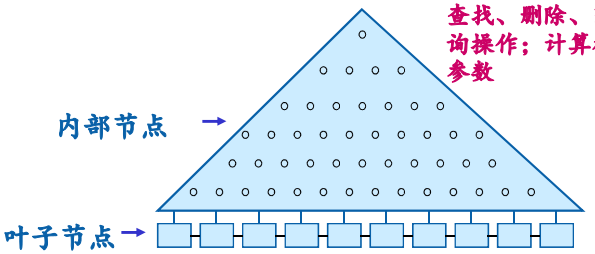
- 数据结构：Buffer head记录R，取值为0或1
  - R=1表示访问时间比较近期
  - R=0表示很长时间没有访问了

7、数据存储与访问路径(2)

- 索引的概念（了解）
- 树结构索引（精通）
  - B+-Trees
  - 索引访问代价
  - 内存优化的B+-Tree
- 哈希索引（精通）
  - Hash function
  - Chained hashing
  - Extendible hashing
- 其他索引（了解）
  - ISAM
  - 位图索引
  - 倒排索引

B+-Trees

给定一组key，可以建树；对树进行插入、查找、删除、范围查询操作；计算树高等参数

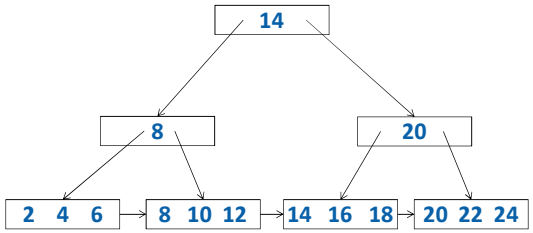


- 每个节点是一个page
- 所有key存储在叶子节点
- 内部节点完全是索引作用

练习1

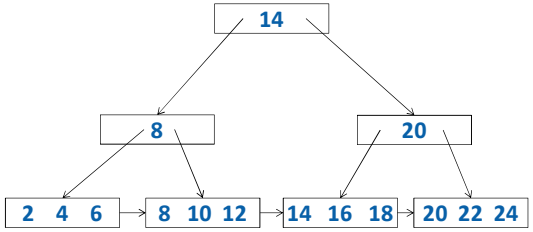
- 假设每个节点的child/pointer个数最多为B=3
- 树中包括下述key:
  2,4,6,8,10,12,14,16,18,20,22,24
- 要求
  - 节点中尽量放满
  - 内部节点至少有2个孩子
- 请画出B+-Tree

练习1解答

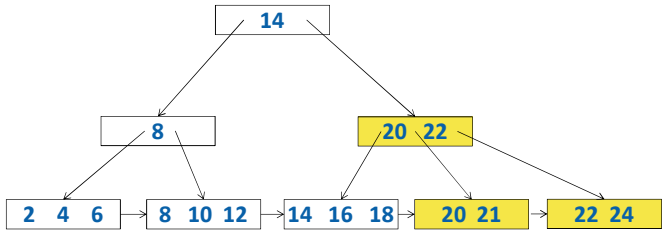


练习3

- 假设每个节点的child/pointer个数最多为B=3
- 请画出insert(21)之后的B^+-Tree

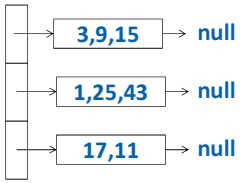


练习3解答



举例

- Key: 1, 3, 25, 17, 9, 11, 43, 15
- h(key) = key
- Size = 3



8、数据存储与访问路径(3)（了解）

- 多维索引
  - 概念
  - 多维散列索引
    - 网格文件（Grid File）
    - 分段散列（Partitioned Hashing）
  - 多维树结构索引
    - 四叉树（Quad Tree）
    - R树（R-Tree）
- 物理数据库设计
  - 索引的选择
  - 只需索引的查询
  - 索引选择的辅助工具

9、查询处理（1）：精通

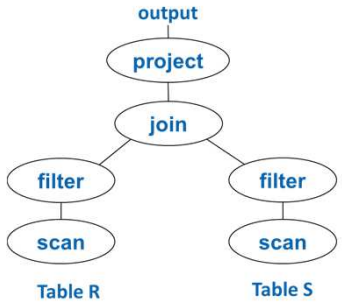
- 查询处理概述
  - 系统目录（System Catalog）
  - 查询执行方式
  - 关系操作实现的常见方式
- 排序和外排序
  - 排序的应用场景
  - 内存排序回顾
  - 外存排序
  - 使用B^+-Tree获得排序数据

方法2：Tuple-at-a-time 迭代求解

- 又称为Volcano方式
- 每个Operator设计成为具有统一的函数接口
  - Open( )：初始化，分配资源，建立数据结构等
    - 进一步调用子Operator的open( )
  - GetNext( )：获得下一条Operator的处理结果
    - 调用子Operator的GetNext( )，在此基础上，进行本Operator所设定的关系运算，得到一条结果
  - Close( )：结束，释放资源
    - 调用子Operator的close( )

方法2：Tuple-at-a-time 迭代求解

- 执行方式是Pull方式
- 建立了Operator tree之后
- 调用根的Open()初始化
- 循环调用根的GetNext( )
  - 父结点将调用子结点的GetNext( )
  - 生成一条条的结果
- 最后，Close( )



10、查询处理（2）：精通

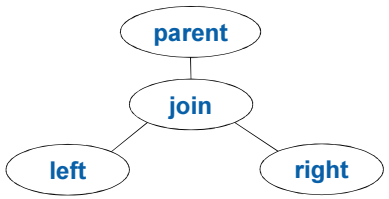
- 选择
- 投影
- 连接
- 去重
- 分组+聚集
- 集合操作
- 内存数据库（了解）

迭代求解实现 Join Operator

- Open( )：初始化，分配资源，建立数据结构等
  - 初始化孩子
  - 完成准备工作：可能包括I/O partitioning+设置第一个Simple Hash Join, Run generation+初始化Run Merge等
- GetNext( )：获得下一条Operator的处理结果
  - 调用子Operator的GetNext( )
  - 进行simple hash join的一步，或者Run merge的一步
  - 对于Hash join，一对Rj和Sj处理完后，初始化下一对的处理
- Close( )：结束，释放资源
  - 调用子Operator的close( )
- 注意Join Operator是Blocking operator，open阶段对一个孩子已经处理完毕

多个Join Operator

- 只需要知道left, right是operator，都提供open, close, getNext接口
- 不需要管具体是什么operator
- 于是可以left或right也可以是join operator



11、查询优化

- 查询优化概述
- 将SQL查询转换成关系代数表达式（精通）
- 关系代数的等价变换（精通）
- 运算代价的估计（精通）
- 基于成本的计划选择（了解）
- 多个查询块：嵌套子查询、视图、集合运算（了解）

转化一个查询块（无嵌套）

select　投影π，去重δ，聚集γ

from　选择σ，或者连接⋈　（先用X表示）

where　选择σ，或者连接⋈

group by　分组γ

having　选择σ

order by　排序τ

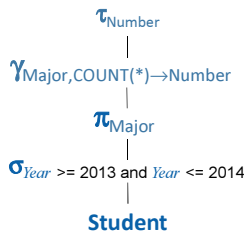
通常使用一个事先定义好的基本方法产生关系代数表达式

举例

统计各系2013-2014年入学的学生人数，并按照人数排序

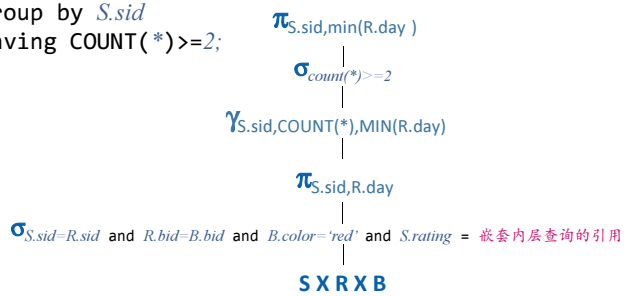
```
select Major, count(*) as Number
from Student
where Year >= 2013 and Year <= 2014
group by Major
order by Number;
```

注意：统一地在选择投影之后使用group-by等，投影的列为所有输出列和中间使用的列



举例

```
select S.sid, MIN(R.day)
from Sailors S, Reserves R, Boats B
where S.sid=R.sid and R.bid=B.bid and B.color='red'
  and S.rating = 嵌套内层查询的引用
group by S.sid
Having COUNT(*)>=2;
```



关系代数等价变换

- 选择
  - AND选择条件的分解
  - 选择+集合运算的等价变换
  - 选择+叉积/连接的等价变形
- 投影
- 叉积和连接
- 去重
- 分组聚集
- 选择下推、投影下推

结果大小估算

- 投影
- 选择
  - Column=value
  - Column < value
  - Column != value
  - Column1=Column2
  - Column in (…)
  - AND, OR, NOT
- 连接
- 并交叉
- 去重
- 分组聚集

举例

- 已知
  - R(a,b)记录数为1000条，b列取值有20个
  - S(b,c)记录数为2000条，b列取值有50个，c列取值有100个
  - T(c,d)记录数为5000条，c列取值有500个
- 估计 $R\bowtie_{R.b=S.b}S\bowtie_{S.c=T.c}T$的结果记录数？
  - 1000*2000*5000/(50*500) = 400,000

12、事务处理

- 事务的概念和ACID（精通）
  - SQL语句表达事务Transaction
  - ACID
- Concurrency Control (并发控制)
  - 数据冲突和可串行化（精通）
    - Isolation level
  - 加锁的并发控制（精通）
    - 2PL
  - 乐观的并发控制（了解）
- Crash Recovery (崩溃恢复)
  - WAL（精通）
  - 崩溃恢复（精通）

Transaction

成功的事务

```
begin transaction;
……
commit transaction;
```

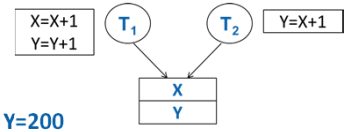
可以用rollback回卷事务

```
begin transaction;
……
rollback transaction;
```

ACID：DBMS保证事务的ACID性质

- Atomicity（原子性）
  - all or nothing
  - 要么完全执行，要么完全没有执行
- Consistency（一致性）
  - 从一个正确状态转换到另一个正确状态
    （正确指：constraints, triggers等）
- Isolation（隔离性）
  - 每个事务与其它并发事务互不影响，好像在独立执行
- Durability（持久性）
  - Transaction commit后，结果持久有效，crash也不消失

Serializable?



初始：X=100, Y=200

T1	T2
Read(X)	
	Read(X)
	Write(Y)
Write(X)	
Read(Y)	
Write(Y)	

Y=102

T1	T2
Read(X)	
Write(X)	
	Read(X)
	Write(Y)
Read(Y)	
Write(Y)	

Y=103

T1	T2
Read(X)	
	Read(X)
Write(X)	
Read(Y)	
Write(Y)	
	Write(Y)

Y=101

……

Isolation Level

	Read uncommitted data （写读）	Unrepeatable Read （读写）	Overwrite uncommitted Data （写写）
Serializable	no	no	no
Repeatable Read	no	no	possible
Read committed	no	possible	possible
Read uncommitted	possible	possible	possible

2PL (2 Phase Locking)

- Pessimistic concurrency control
- 对每个访问的数据都要加锁后才能访问
- 有一个集中的加锁阶段和一个集中的解锁阶段
- 算法如下
  - 在Transaction中，对每个需要访问的数据加锁
    - 如果不能加锁，就等待，直到加锁成功
  - 在Transaction commit前，集中进行解锁
  - Commit

一旦解锁开始，就不会加锁了

实现细节1：读写的锁是不同的

- Shared lock(S)：保护读操作　　（共享锁）
- Exclusive lock(X)：保护写操作　　（互斥锁）

Lock Compatibility Matrix

	Shared Lock(S)	Exclusive Lock(X)
Shared Lock(S)	√	X
Exclusive Lock(X)	X	X

实现细节2： Lock Granularity

Lock Compatibility Matrix

	IS (intent shared)	IX (intent exclusive)	S (shared)	X (exclusive)
IS	√	√	√	X
IX	√	√	X	X
S	√	X	√	X
X	X	X	X	X

什么是Write-Ahead Logging?

- Write-Ahead
  - Logging 总是**先于**实际的操作
  - Logging 相当于意向，先记录意向，然后再实际操作
- 写操作
  - 先Logging
  - 然后执行写操作
- Commit
  - 先记录commit 日志记录到外存的事务日志文件
  - 然后commit

Crash Recovery

- Crash后重新启动
- ARIES算法
  - 分析阶段
  - redo阶段
  - undo阶段

13、数据仓库（了解）

- 数据仓库简介
  - 数据仓库vs.事务处理（精通）
  - Star Schema：数据仓库中常见
  - 常见的query形式
- OLAP与Data Cube
  - Roll up / drill down
  - Slice, dice
- 实现介绍
  - Group by … with cube （精通）
  - 固化视图（Materialized View）

14、并行分布数据库（了解）

- 系统架构和关键技术
  - 三种架构
  - Horizontal partitioning
- 分布式查询处理
  - 并行选择、投影
  - 并行连接
- 分布式事务处理
  - 2 Phase Commit