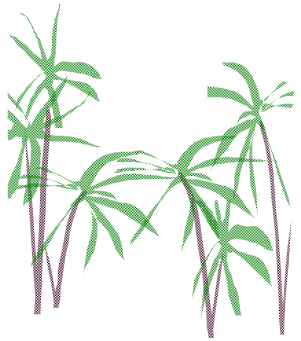
高级数据库技术
Advanced Database technology

可靠性、性能提升
安全、分布式数据库、数据仓库

湖南大学信息科学与工程学院

杨金民

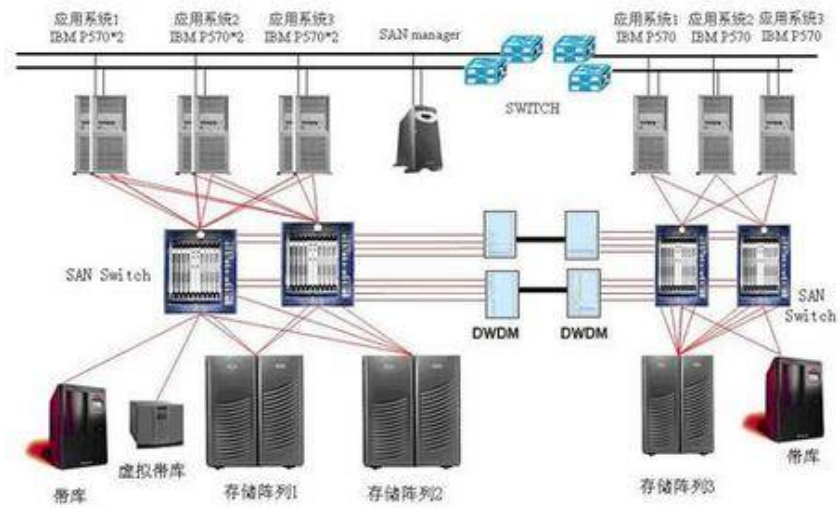
2015. 09



Course Information

- **上课老师**: 杨金民 教授
- **办 公 室**: 湖南大学软件大楼403
- **电　　话**: 13975896967, 0731-8711167
- **Email**: rj_jmyang@hnu.edu.cn
- **QQ**: 909485030;
- **网站**:
  http://ss.hnu.cn/newweb/teachers/zhuanzhijiaoshi/yangjinmin/index.htm

企业数据中心



数据就是**企业**：是企业最核心的内容

企业就是**数据**：全浓缩在拥有的数据中

要解决的问题：

高可靠：**数据一致和不丢失；**

高性能：**速度快，吞吐量大；**

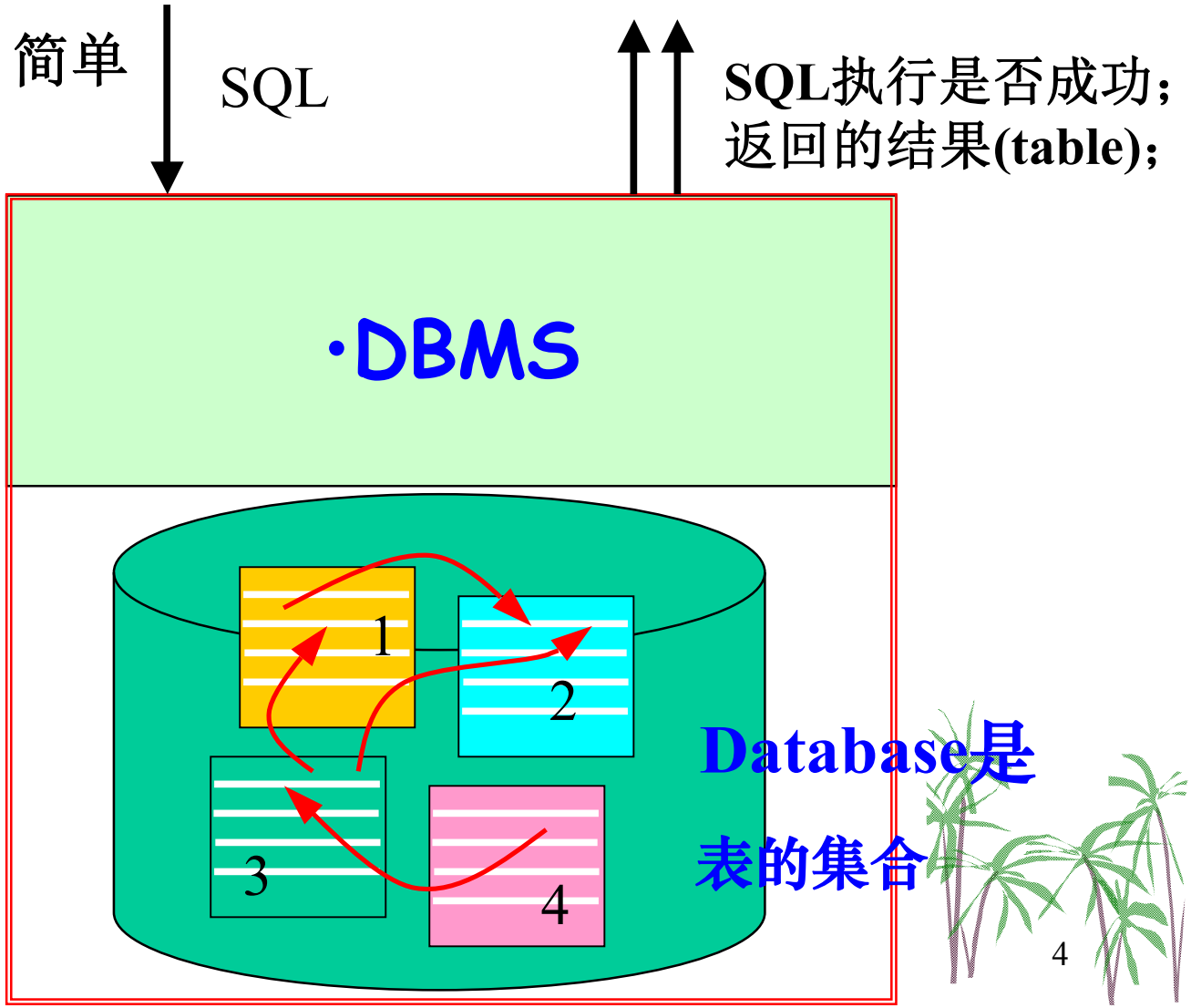
高安全：**不泄露，不可抵赖；**

高体验：**通俗，简单，友好**

关系型数据库

使用二维表结构来组织和存储数据。

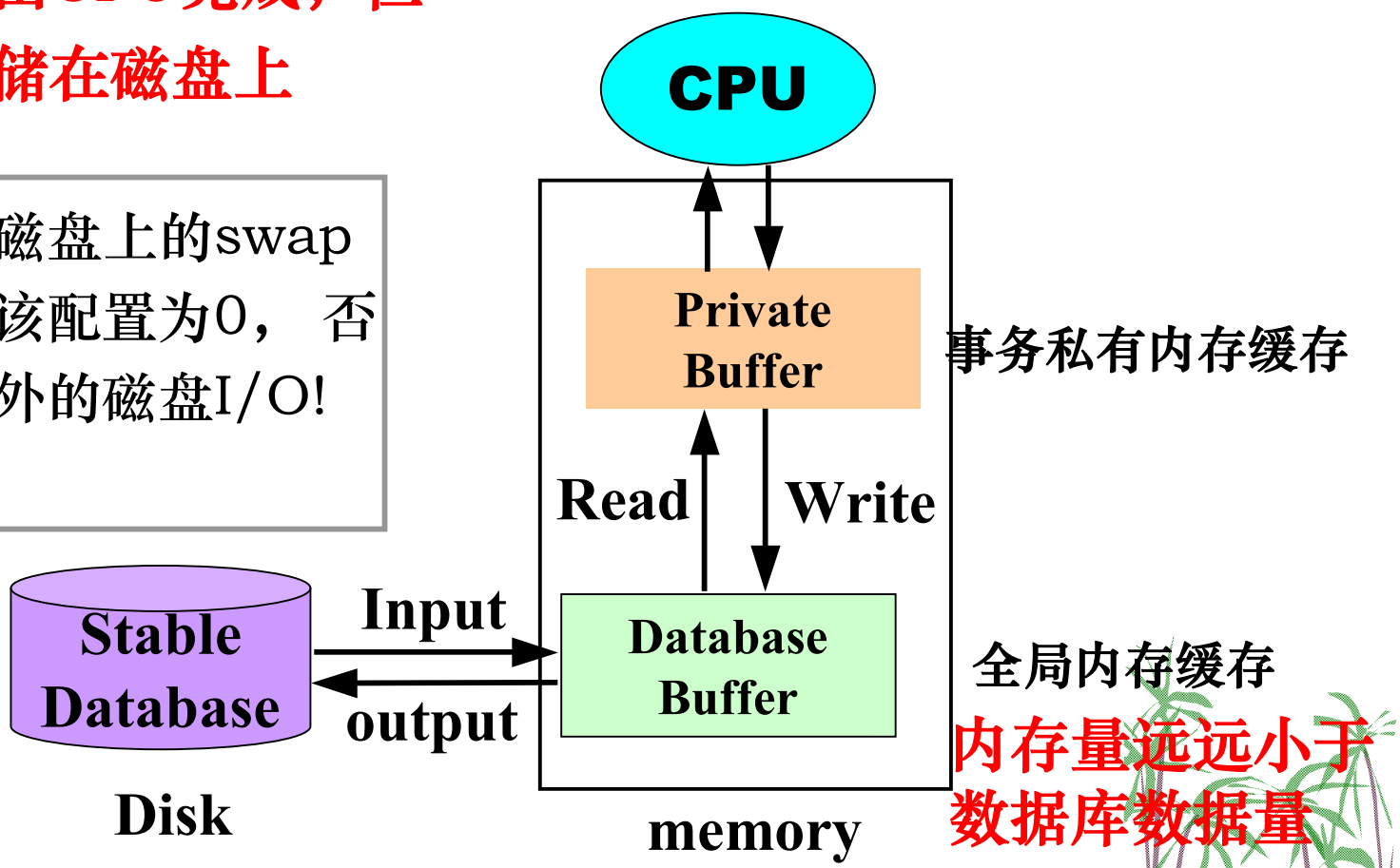
现有数据库系统都是关系型数据库；



DBMS的数据处理模型

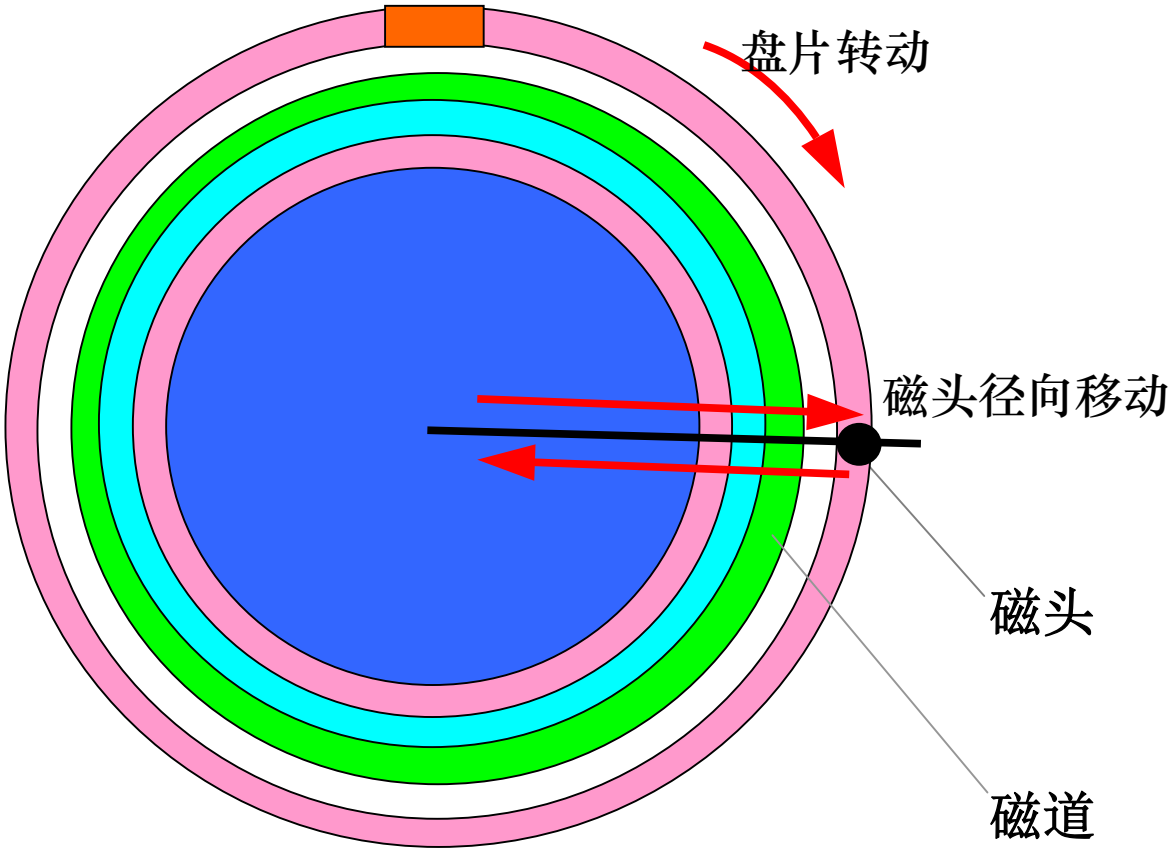
所有处理要由CPU完成，但
数据全部存储在磁盘上

虚拟内存在磁盘上的swap空间大小应该配置为0， 否则会引起额外的磁盘I/O!
Why?



数据量巨大，散布在磁盘盘面空间中

磁盘原理



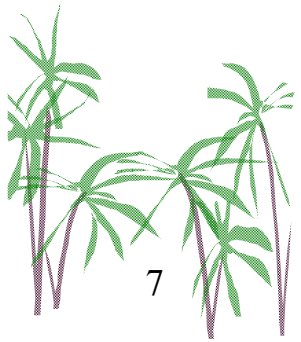
磁盘：

不可替代的优点：容量大，数据不受停电和系统故障的影响

不可忽视的缺点：访问速度慢；

数据库管理系统中的基本问题

- 数据正确性问题；
- 数据处理性能问题；
- 数据操作简单性问题；
- 数据安全问题；
- 数据完整性问题。



第一部分

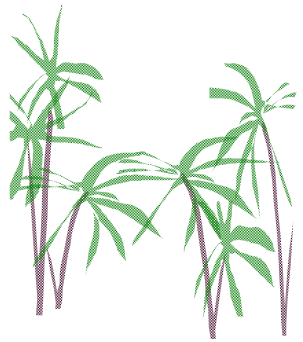
可靠性技术

故障时的数据正确性保证技术

保证故障发生时数据不丢失和一致

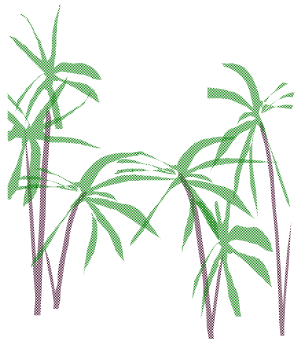
杨金民

2015. 09



主要教学内容

- 事务属性；
- 事务状态；
- 数据处理模型；
- 磁盘原理；
- 系统故障；
- 日志；
- 检查点；
- 备份；
- 容灾；



事务概念

银行交易系统

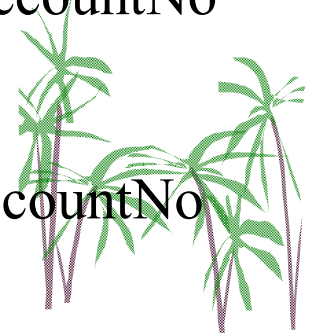
account

Name	accountNo	identityNo	balance
周山	2008043101	430104198008064311	1200
汪兵	2008043214	430104197612274810	80,000
张珊	2008043332	430104196902114933	137,000

例子: 对于银行交易系统来说，假定从汪兵的帐上划转1万元钱到周山的帐上，那么有两步操作，分别为汪兵的帐上减1万，周山的帐上加1万，**假定刚完成第一步操作时，停电，那么数据就会不一致**;

UPDATE account **SET** balance = balance + 10000 **WHERE** accountNo ='2008043101';

UPDATE account **SET** balance = balance - 10000 **WHERE** accountNo ='2008043214';

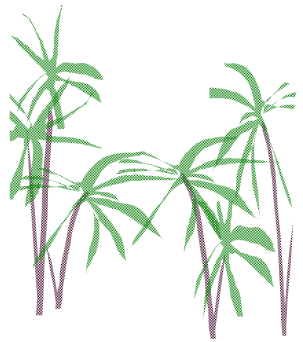


事务特征

事务是指对数据库的一组操作，这些操作涉及对多个数据项进行更新/修改；

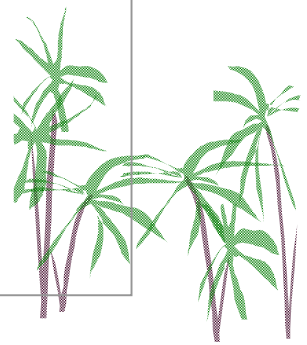
要求：中间状态对外部不可见；

```
TRANSACTION BEGIN
    UPDATE account SET balance = balance + 10000
    WHERE accountNo =‘2008043101’;
    UPDATE account SET balance = balance - 10000
    WHERE accountNo =‘2008043214’;
COMMIT;
```

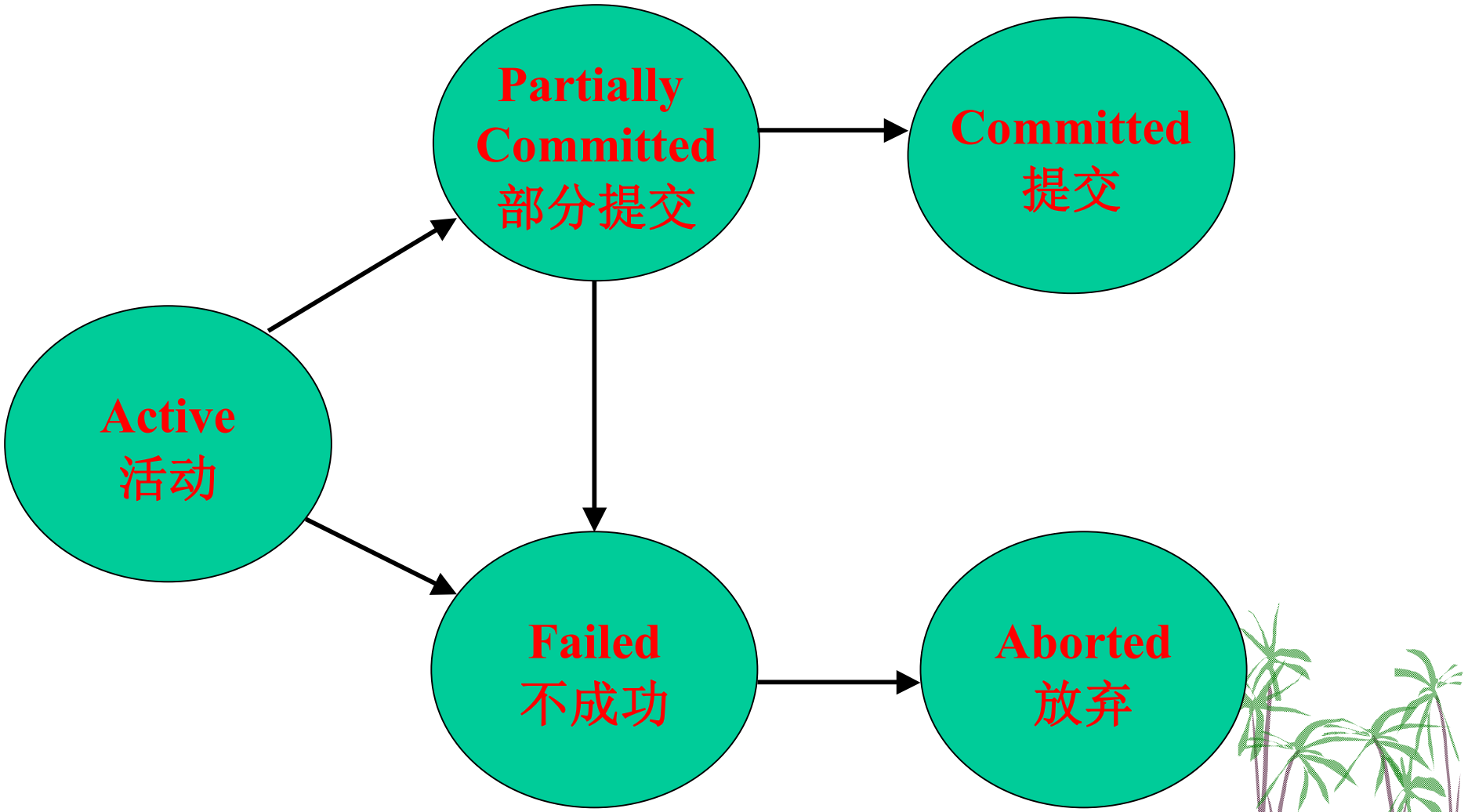


系统故障

- 事务故障
  - 逻辑故障， 例如: 除以 0.
  - 余额不允许为负；
- 系统崩溃故障
  - 停电、硬件故障，蓝屏死机故障；
- 磁盘故障
  - 数据丢失
- 灾难故障
  - 例如：地震，恐怖袭击，火灾；



事务的五个状态


Partially
Committed
部分提交
Committed
提交
Active
活动
Failed
不成功
Aborted
放弃

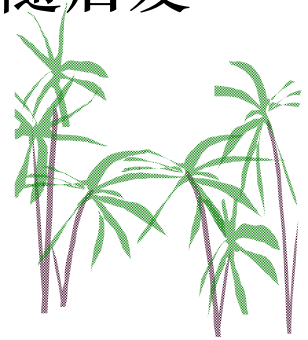
事务的ACID属性

原子性（Atomicity）： 一个事务中的操作要求要么全部执行，要么全部不执行.

一致性（Consistency）： 在外部看来，数据库中的数据总是正确的.

隔离性（Isolation）： 尽管多个事务在并发执行，但从外部看来，具有多个事务串行执行的效果；

持久性（Durability）： 一个事务一旦提交了，即使随后发生故障，其结果在数据库中不会丢失；



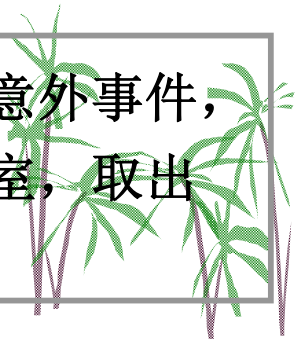
故障恢复

在有故障可能的情况下，如何保证事务的四个属性；

故障恢复包括2个部分：

- **正常执行时**的**防备措施**，为故障恢复做准备；
- 在**故障发生后**的**故障恢复措施**，保证事务的四个属性；

日常生活也是如此，例如为了防备发生钥匙丢失而不能进家门的意外事件，你事先配一把钥匙存放在办公室。一旦发生丢失，你就跑到办公室，取出备匙，意外便得到低代价处理，否则就进不了门，或者要砸门。



第一种方法：分页方法

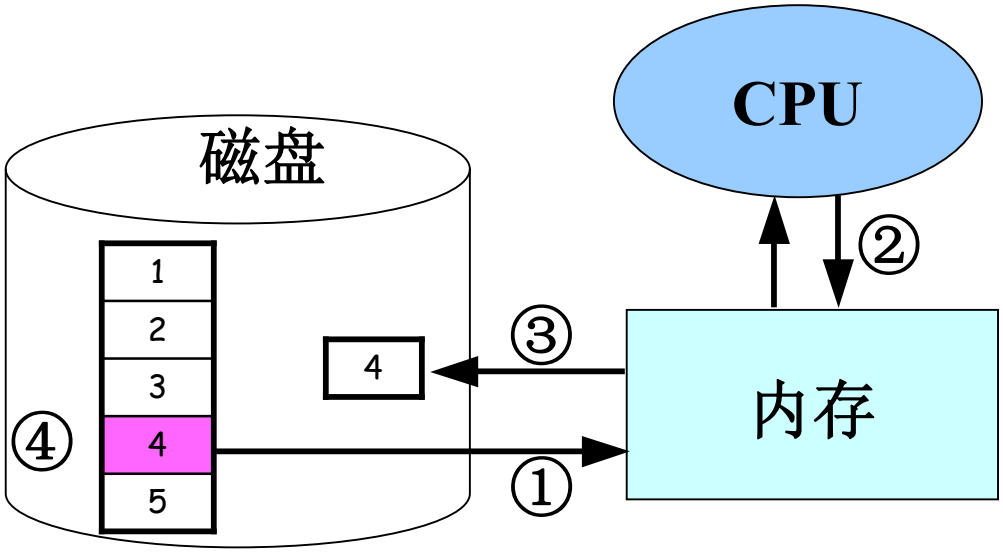
对每个事务：

1) 把要访问的数据页读入内存；

2) 数据处理；

3) 把修改了的数据页作为一个新页写回磁盘，注意不要覆盖旧页；

4) 完成第3步操作后，在磁盘上删除过时了的旧页；



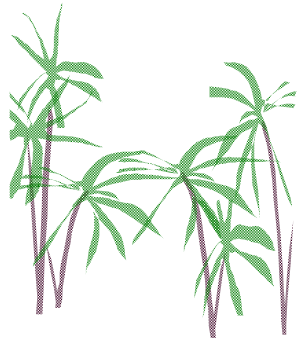
分页方法的特征

- 优点：**简单**.

  故障恢复方法：如果因故障导致第3步没有完成，那么原数据页还在，就当该事物没有发生看待，保证了事物的原子性；

- 缺点：**效率不高**；

  原因是：磁头要在磁盘盘面上到处来回移动。因为磁头一会要读事物要处理的数据页，一会要去磁盘上的哪个地方找一个空闲页来写修改后的数据，还要回过头来删除旧页。



第二种方法：日志方法(广泛采用)

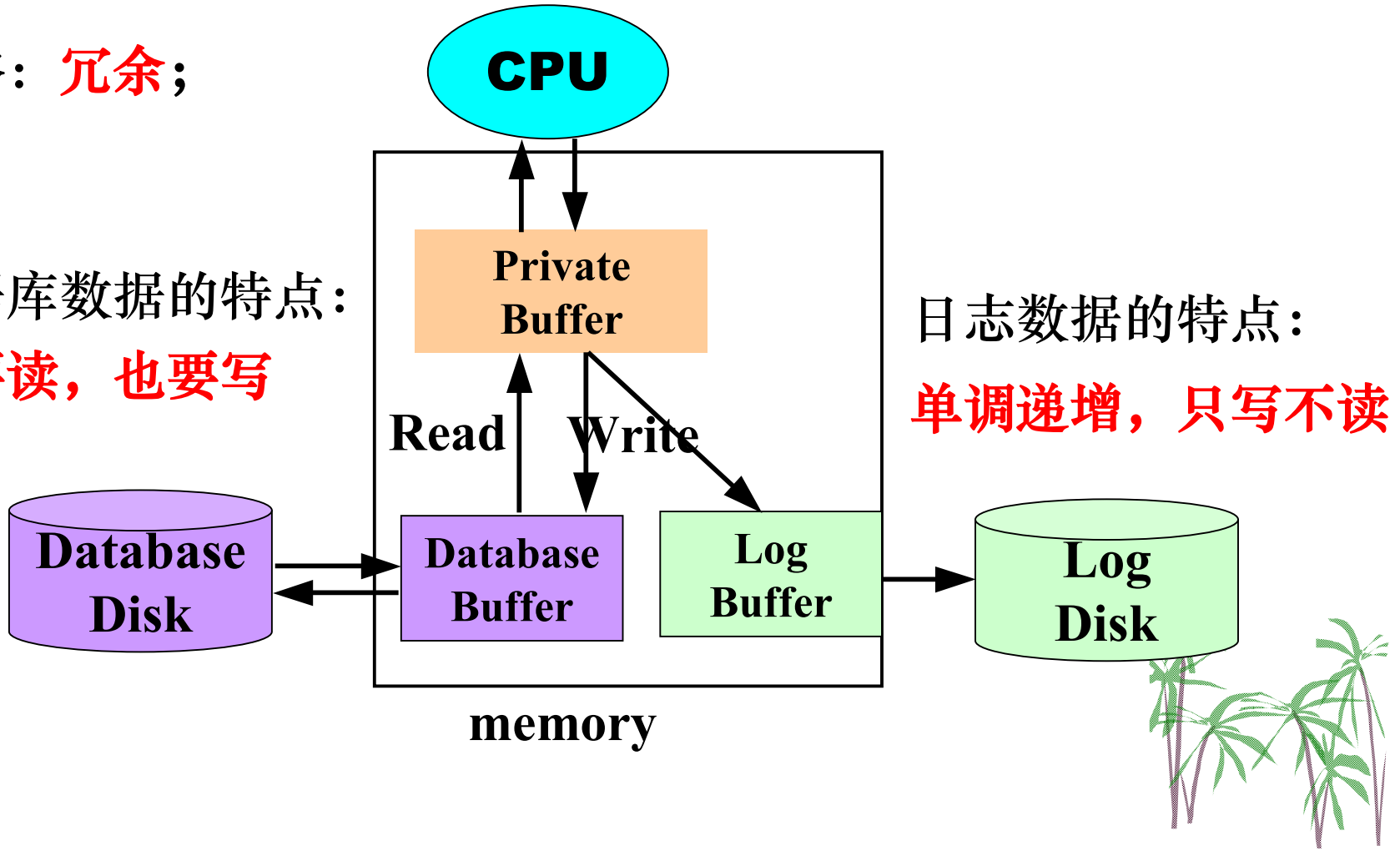
策略：冗余；

数据库数据的特点：

既要读，也要写

日志数据的特点：

单调递增，只写不读



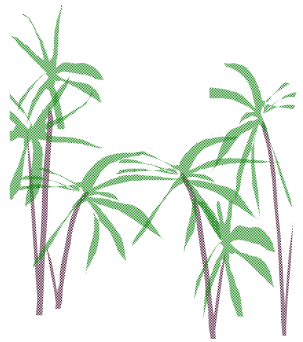
把日志缓冲区的内容输出到日志磁盘

有两种同步约束:

1）当数据库缓冲区中的某个内容要输出到数据库磁盘时，要先把与其相关的日志缓冲区内容输出到日志磁盘之后，才允许(WAL）；

- 数据库缓冲区中数据要输出到磁盘，是因为当要从磁盘中读数据到内存空间来进行处理时，必须先为其先腾出空间来；

2）当事务 T_i 完成，遇到<T_i commit>记录时，要把日志缓冲区的内容输出到日志磁盘，以便保证事物的四个属性；



日志内容

T_0: **Read** (A);
$A = A - 50$;
Write (A);
Read (B);
$B = B + 50$;
Write (B);

T_1: **Read** (C);
$C = C - 100$;
Write (C);

<T_0 **start**>
<T_0, A, 1000, 950>
<T_o, B, 2000, 2050>

(a)

<T_0 **start**>
<T_0, A, 1000, 950>
<T_o, B, 2000, 2050>
<T_o, commit>
<T_1 **start**>
<T_1, C, 700, 600>

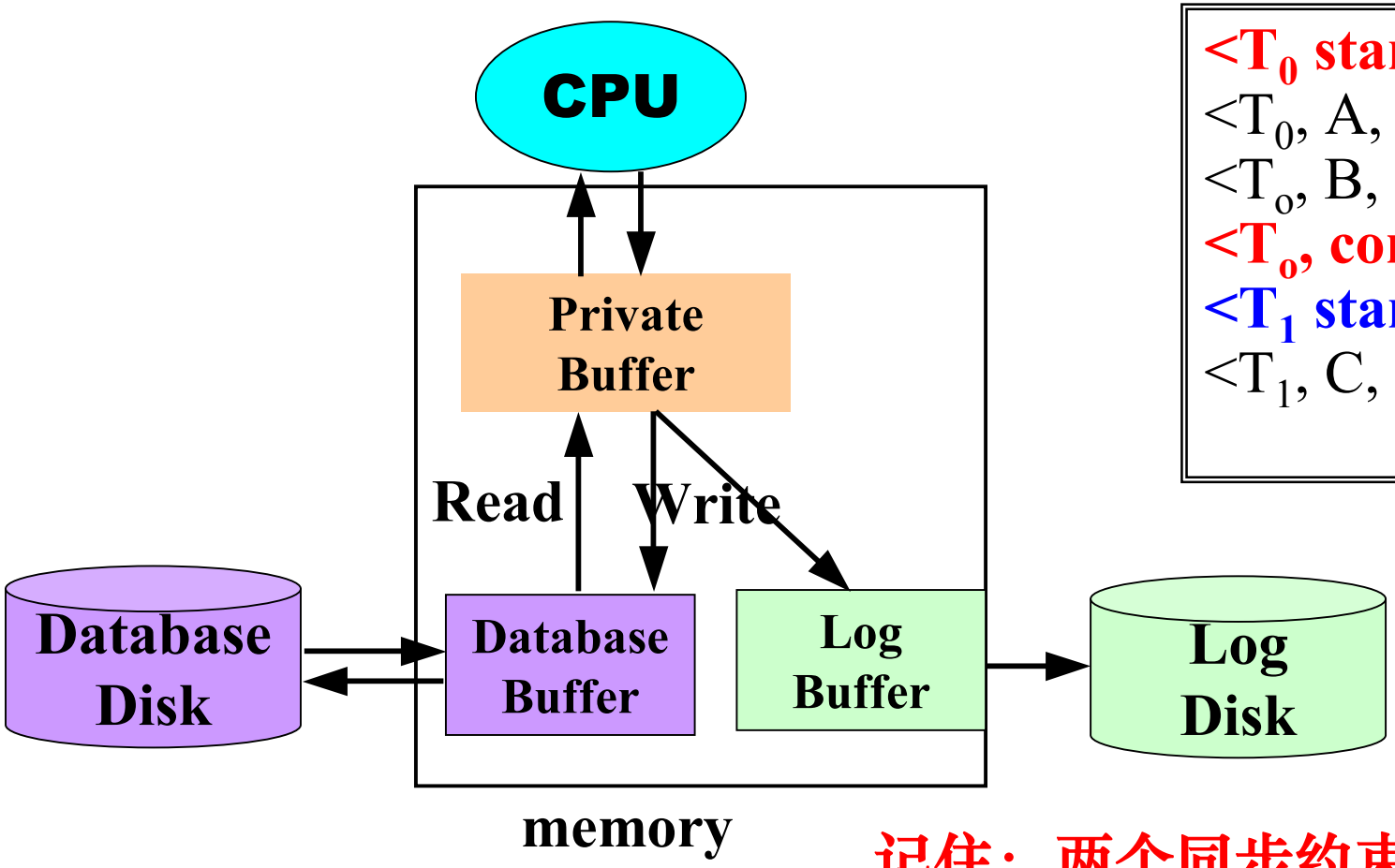
(b)

<T_0 **start**>
<T_0, A, 1000, 950>
<T_o, B, 2000, 2050>
<T_o, commit>
<T_1 **start**>
<T_1, C, 700, 600>
<T_1 **commit**>

(c)

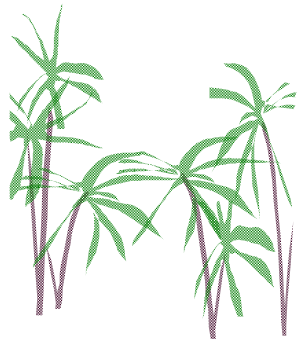
当重启数据库系统时，检查日志内容，(a)/(b)/(c) 是可能出现的场景

日志方法实例



<T_0 start>
<T_0, A, 1000, 950>
<T_o, B, 2000, 2050>
<T_o, commit>
<T_1 start>
<T_1, C, 700, 600>

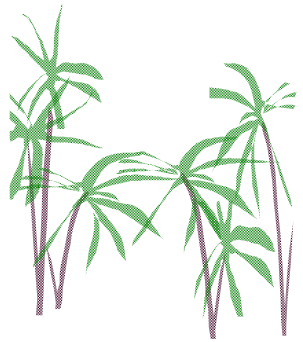
记住：两个同步约束



事务故障的恢复方法

特征：事务不能执行下去，未完成；

恢复方法：执行**回卷(Rollback)操作**：当事物T_i要撤销时，**反向**扫描日志内容，对T_i的**每项**数据操作记录，执行undo(T_i) 操作，使用旧值恢复数据项的原有值，即撤销事务所做的数据操作；**直至**遇到<T_i start>记录**为止**，然后**放弃**T_i；



系统崩溃故障的恢复方法

① 重启数据库管理系统；

② 从日志磁盘读取日志文件；

③ 反向扫描日志，即从日志文件的结束位置开始后向扫描。对于在日志记录中**没有**$<T_i$ commit$>$ **记录**的事物，执行**回卷操作(Rollback)**，使用旧值恢复数据项，**做撤销处理**:

④ 然后从日志文件的开始位置前向扫描。 对日志记录中**含有**$<T_i$ commit$>$的事务，执行redo(T_i) 操作，使用新值赋值数据库中的数据项，**确保事物的生效性**；

检查点(Checkpoint)概念

做检查点的目的：**加快系统崩溃故障恢复过程** 。基于如下观察：

- 扫描整个日志文件很费时；
- 对已经输出到了数据库磁盘的事务数据项没有必要再做redo操作；

周期性地做**检查点**(checkpointing)：

① 暂停所有当前活动事务；

② 把日志缓冲区中的所有日志记录输出到日志磁盘；

③ 把数据库缓冲区中的所有修改输出到数据库磁盘；

④ 写一条 < **checkpoint**, < 当前活动事务的标识号列表 >> 日志记录到日志磁盘；

有检查点时的系统崩溃故障恢复

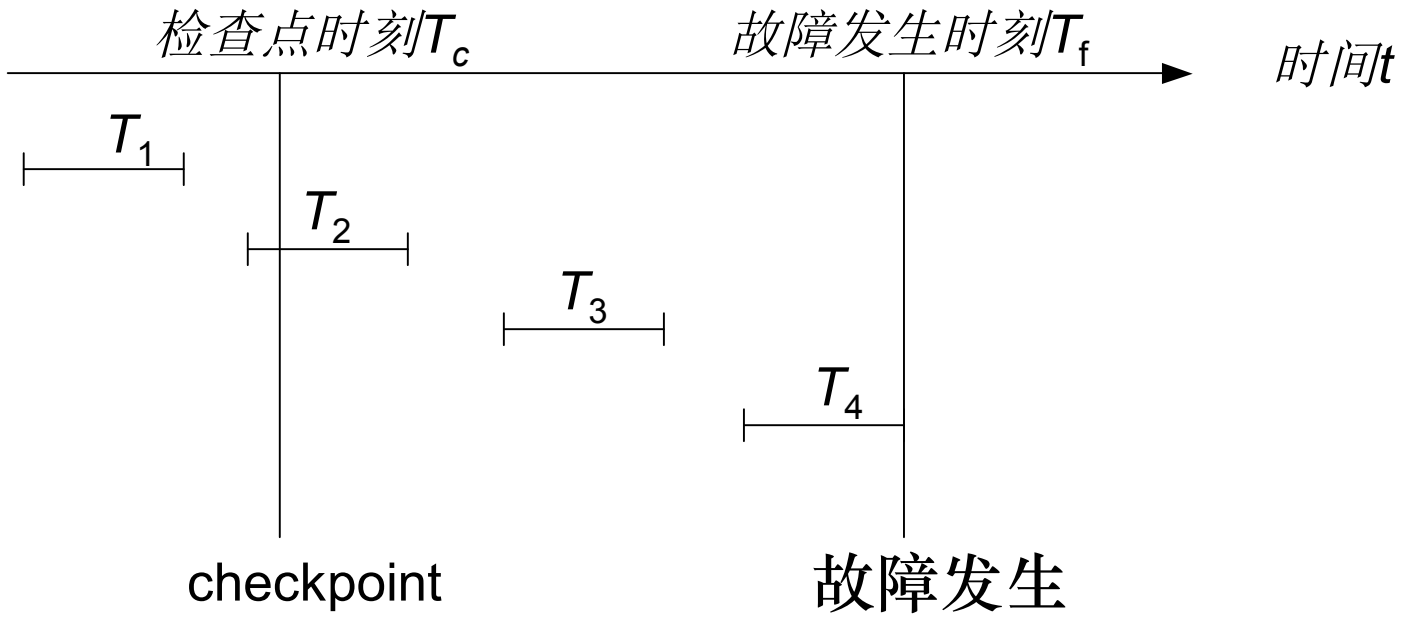
① 从日志文件的结尾处反向扫描，直至遇到最近的 <checkpoint> 记录为止；

② 对反向扫描中只有<T_i start> 而没有<T_i commit>的事务，执行回卷操作(Rollback (T_i))，做撤销处理；

③ 从<checkpoint>开始前向扫描日志记录，对有<T_i commit>的事务，执行redo(T_i)操作，保证其生效；

- 没有检查点时，日志文件反向扫描要到起始位置。检查点加快了故障恢复过程，缩短了恢复时间；

检查点加快故障恢复的例子



故障恢复时，反向扫描至checkpoint即可：

- T_1 可以忽略；
- Rollback(T_4);
- redo T_2 and T_3;

备份(Dump)操作—应对数据库磁盘故障

周期性地执行**备份(dump)** 操作，对磁盘数据库进行磁盘备份:

① 不再接收客户事务请求，让当前所有活动事务执行完毕;

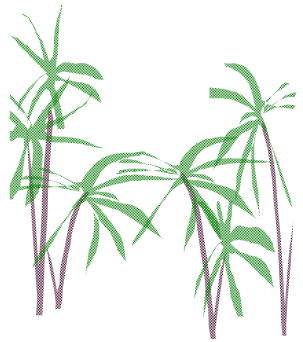
② 输出日志缓冲区中的日志记录到日志磁盘中;

③ 输出数据库缓冲区中的缓冲数据到数据库磁盘中;

④ 把数据库磁盘中的数据库文件拷贝到另一个磁盘上;

⑤ 往日志磁盘中写入一条 **<dump>** 日志记录;

⑥ 接收客户事物请求，恢复正常处理;



数据库磁盘故障的恢复方法

特征：数据库数据全部丢失；

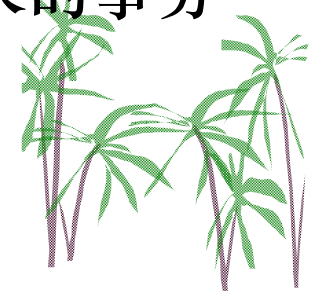
恢复方法：

① 用最近备份的数据库磁盘替换掉失效的数据库磁盘；

② 重启数据库管理系统；

③ 读日志文件，从文件末尾反向扫描直至<dump>记录；

④ 再顺向扫描日志记录，对有<T_i commit>记录的事务做redo(T_i)操作；



日志磁盘故障的恢复方法

日志本来就是冗余的，只要数据库不发生故障，日志就没用场；

恢复方法：

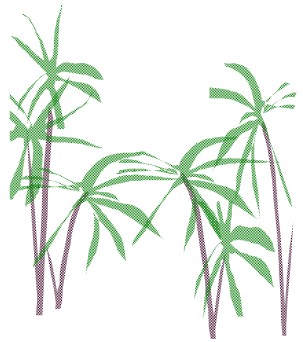
① 不再接收客户事务请求，让当前的所有活动事务执行完毕；

② 输出数据库缓冲区中的缓冲数据到数据库磁盘中；

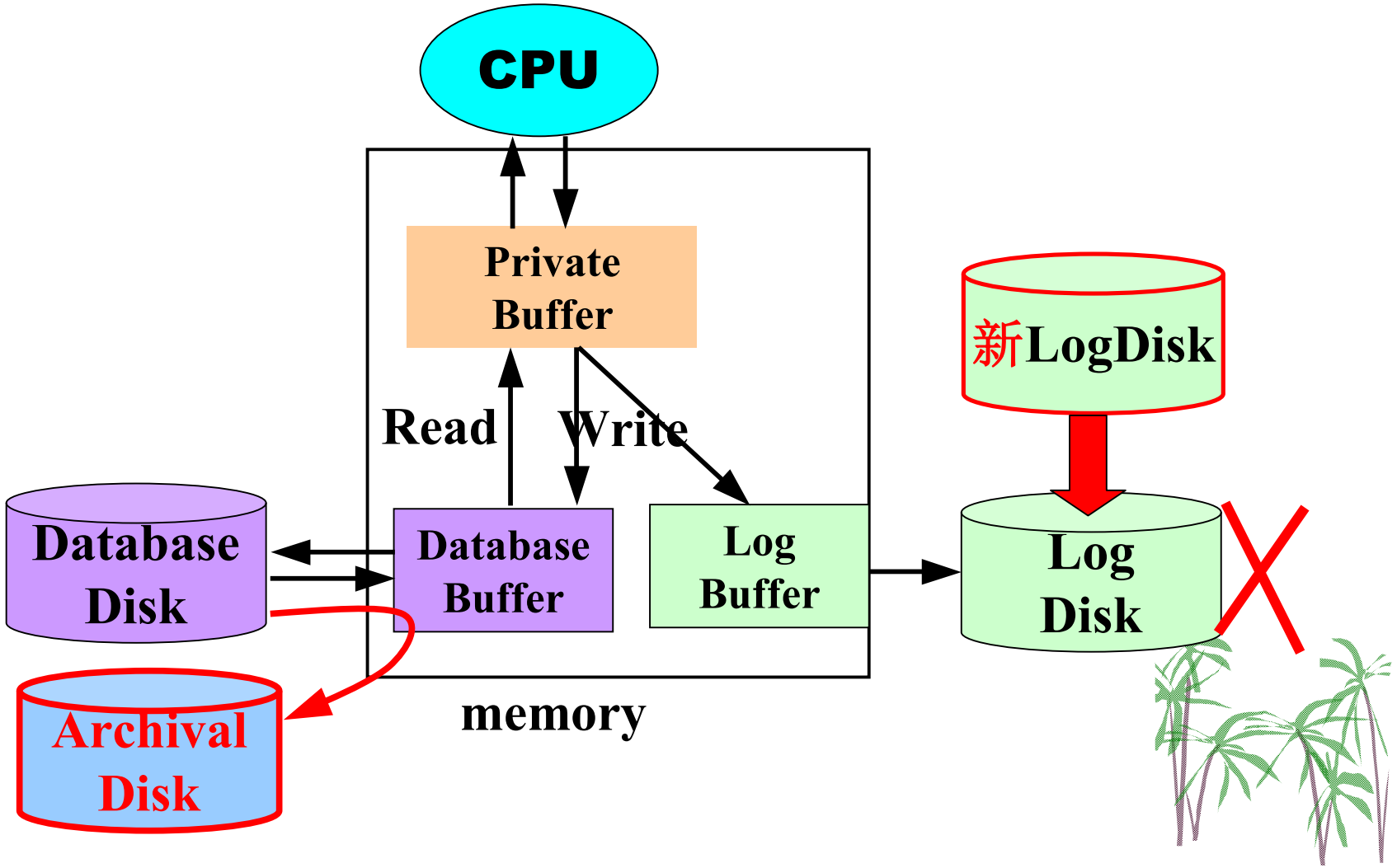
③ 执行备份(Dump)操作，把磁盘中的数据库文件拷贝到另一个磁盘上；

④ 用一个好的磁盘更换日志磁盘；

⑤ 恢复正常处理；

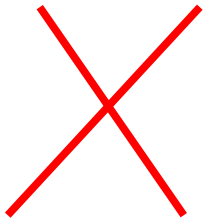


日志磁盘故障的恢复方法



日志磁盘和数据库磁盘同时故障

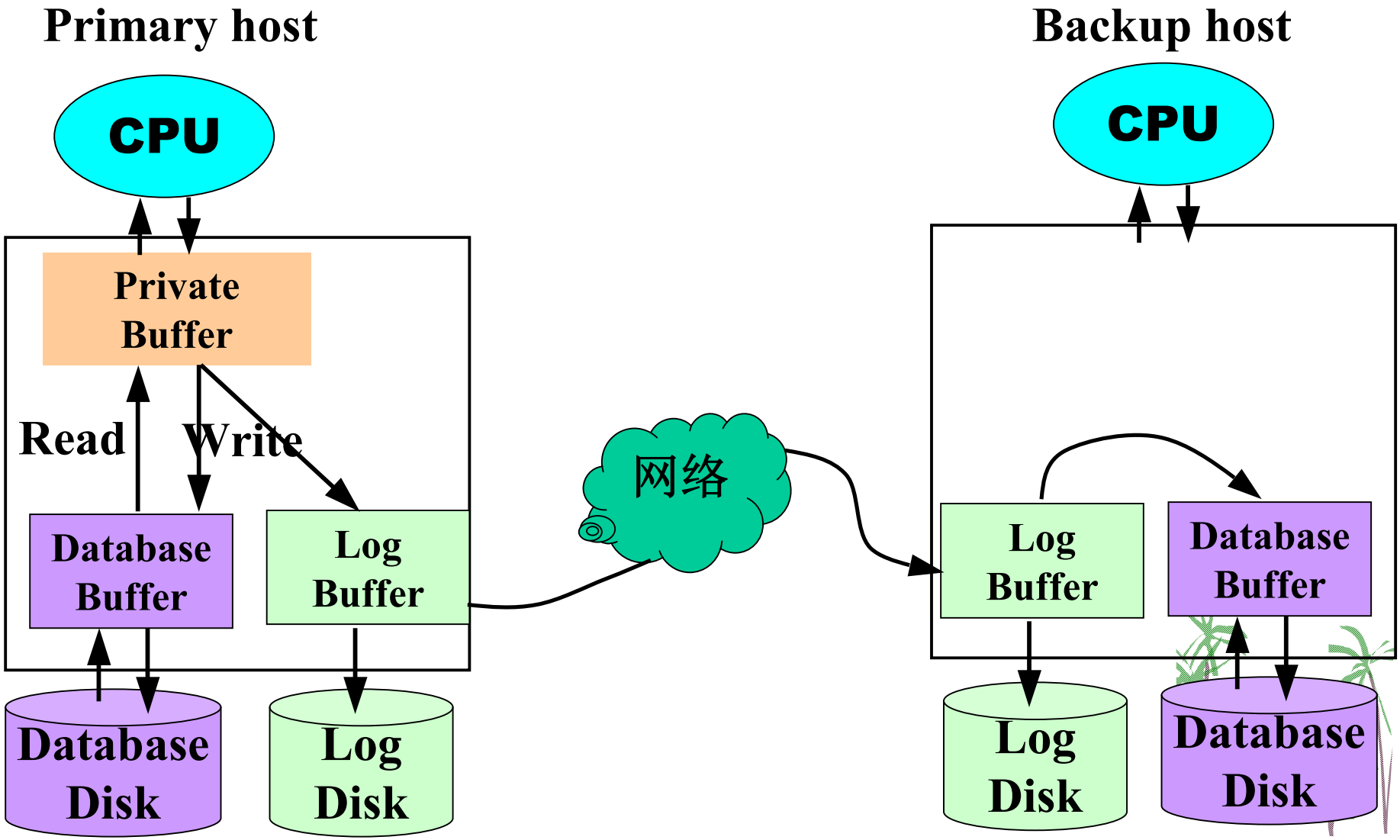
- **无法恢复**；



- 这是一个**小概率事件**：假定一个磁盘故障的概率为p，那么两个磁盘同时故障的概率为p^2；

例如，磁盘的正常工作时间可达5年，即故障概率为10^{-3}级，p^2为10^{-6}级，即200年左右。也就是说，**可靠性从5年提高到了200年**。抗洪能力从5年一遇到200年一遇的防洪大堤修建也就是这个道理。

容灾—远程备份

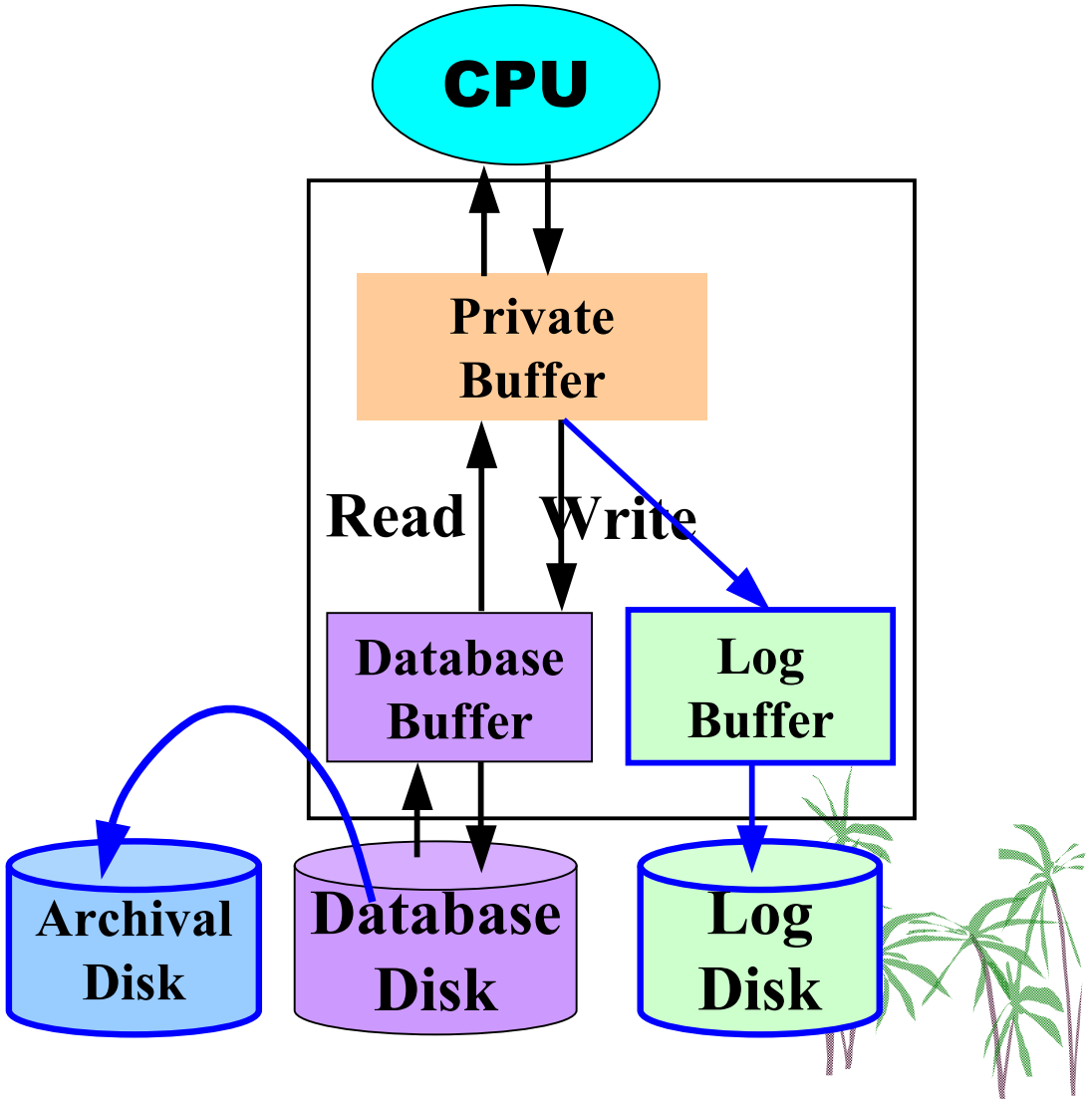

Primary host
Backup host
CPU
CPU
Private Buffer
Read
Write
网络
Database Buffer
Log Buffer
Log Buffer
Database Buffer
Database Disk
Log Disk
Log Disk
Database Disk

性能与可靠性的折中平衡

- 要完全保证事物的四个属性，应该是primary机在收到backup机对 <T_i commit > 日志记录的接受响应时，才能提交给客户；
- 这样提交就要等待，性能就会受到很大影响。于是，把约束放宽到本机把<T_i commit>写入本机的日志磁盘时，即可提交事务；
- 这样，当发生灾难时，有可能个别或者少数提交的事物不能得到恢复，但能接受，因为灾难很少见。既然发生了，丢失一点交易，客户也能谅解 。

故障恢复总结

- 事务故障: 回滚;
- 系统崩溃故障;
- 数据库磁盘故障;
- 日志磁盘故障;
- 灾难故障;



日志方法容错的本质

- 数据库磁盘中，由于数据量巨大，数据库中的数据散布在整个磁盘面上，而且既要读，又要写。数据写磁盘具有随机性。如果每次都把改动写回到磁盘，磁头就会在整个盘面上到处移动：一会到这里读，一会又到那里写。磁头移动是机械运动，因此性能会很低；
- 日志数据具有完全不同的特性，系统正常时，日志是只添加，没有读取，因此不存在磁头来回移动的问题；性能就会大大提高；
- 这一实质告诉我们，在安装数据库系统时，**日志和数据库千万不能配置在一个物理磁盘上，一定要分开，否则性能将剧降；**

高级数据库技术

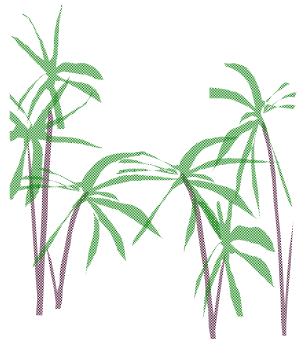
Advanced Database technology

(二）数据处理性能提升技术

湖南大学
信息科学与工程学院

杨金民

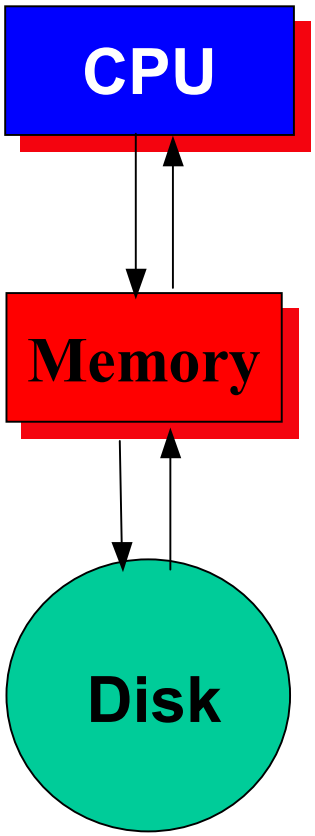
2015. 09



数据处理性能问题

- 企业的数据都集中存储在数据库中，**海量的数据**。从海量的数据中查找和定位数据**非常耗时**。
- 数据集中存储后，所有用户都要来访问数据库，因此数据库服务器的**负载非常重**。而且是**很多用户同时访问**数据库。
- 目前很多企业的数据库系统都因处理性能低下，超出用户容忍范围而受到用户的非议与责难，影响企业的正常运作和发展。
- 数据分类、数据排序、数据索引，都是提升数据处理性能的有效方法。
- 提升数据处理性能可以从挖掘和**利用数据特性**，**硬件特性**，以及**访问特性**入手寻找办法。

数据处理过程



- 所有处理要由CPU完成，但数据全部存储在磁盘上。CPU和磁盘很不匹配：CPU速度很快，磁盘速度很慢，相差天文数字（10^6）个数量级；
- 磁盘有不可替代的特质：存储容量大；其上的数据不受系统故障影响；
- 内存速度比磁盘高很多，起到缓冲作用，但其容量比磁盘小很多，其上的数据在发生死机或停电之类的故障时，数据丢失；
- **数据库数据量通常比内存容量大很多；**

数据库性能

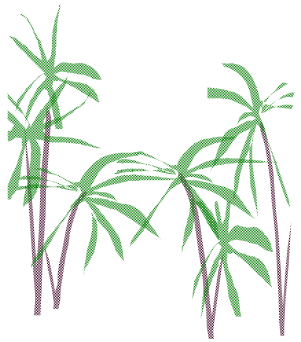
度量指标：

- **事务吞吐量**(Transaction throughput)：单位时间中能够处理的交易(事务)数量.
- **响应时间**(Response time)：完成单个交易所用的时间.

两个指标必须相提并论，不能单独来提。
如果只说单一指标，就很容易解决。两个都保证，就难

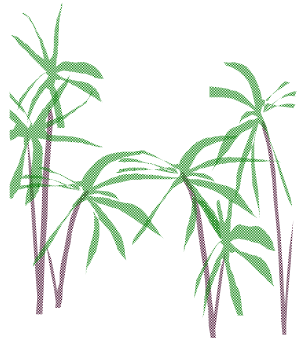
提高数据库性能的策略

- 挖掘和利用：
  - **数据特性**，
  - **硬件特性**，
  - 以及**数据访问特性**；



提高数据库性能的方法

- 方法1: 排序;
- 方法2: 索引 （哈希索引）；
- 方法3: 连续的磁盘存储;
- 方法4: 分类、聚簇;
- 方法5: 内存缓冲;
- 方法6: 并发执行;
- 方法7: 查询优化;
- 方法8: 日志和数据分盘存储;

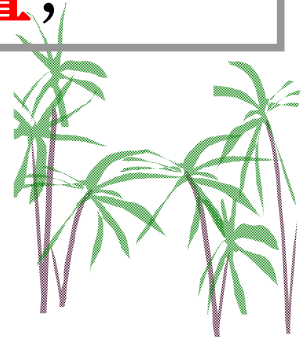


其中与数据库设计有关的方法

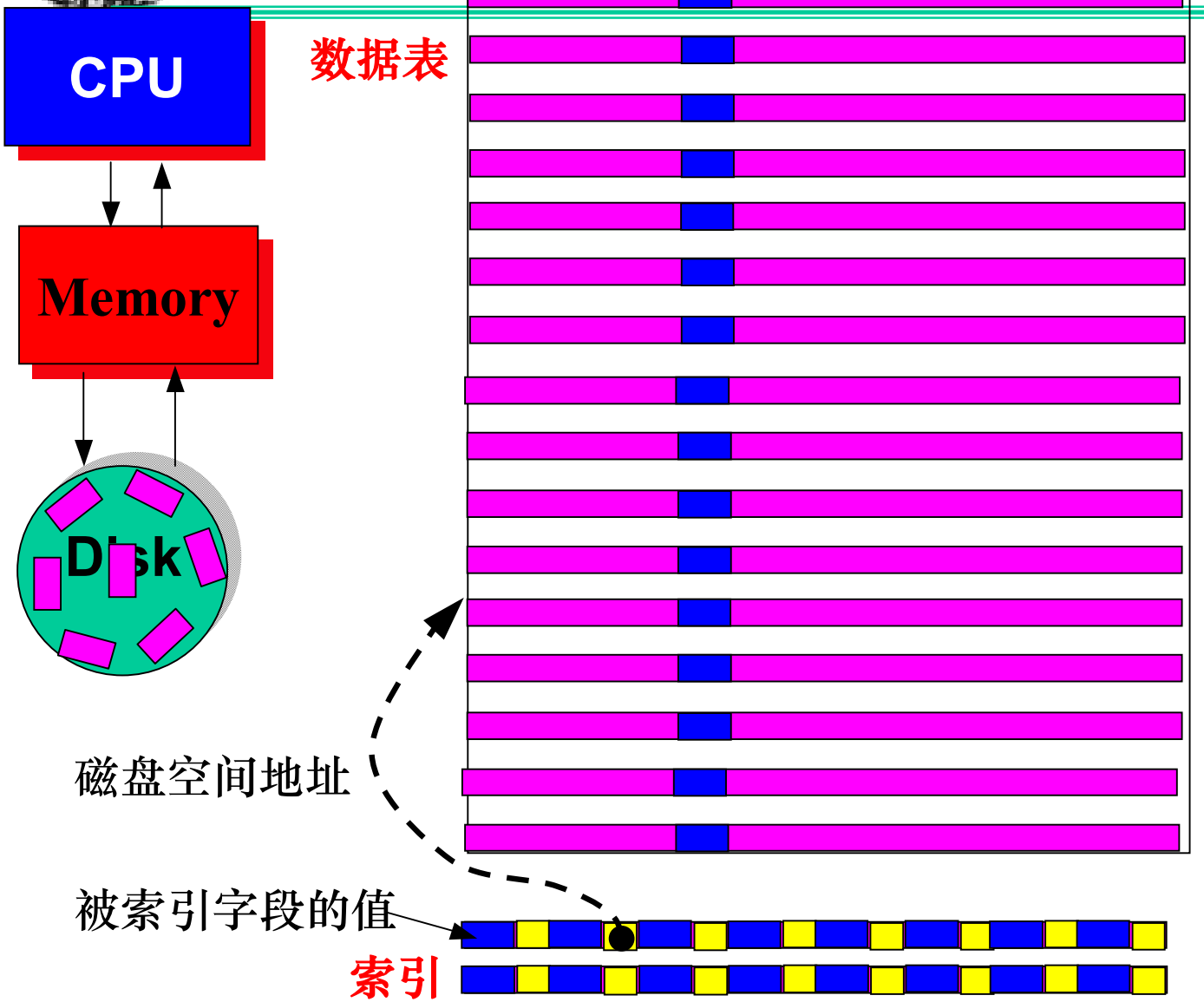
- 方法1: 排序;
- 方法2: 索引 ;
- 方法3: 连续的磁盘存储;
- 方法4: 分类、聚簇;
- 方法5: 内存缓冲;
- 方法8: 日志和数据分盘存储;

方法6“并发执行”和方法7“查询优化”，**完全封装在DBMS中，对数据库设计者和DBA透明**;

其它的方法都需要**数据库设计者和DBA进行配置**;



提高表数据查询性能—创建索引（Index）



大块的数据变成了很小的索引（仅2行），缩小了很多很多倍，可以一次性加载到内存里，迅速地找到想要的行，然后将行数据从磁盘读入内存。

没有索引，则要把大块数据全运输到内存，一个一个地比对，仅只极少的行有用。

创建索引时的注意事项

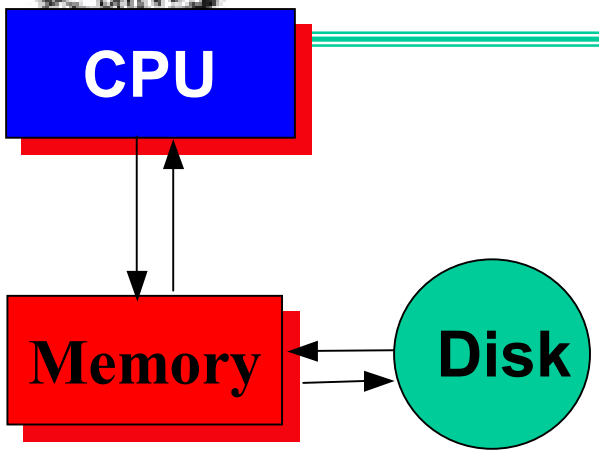
语法：

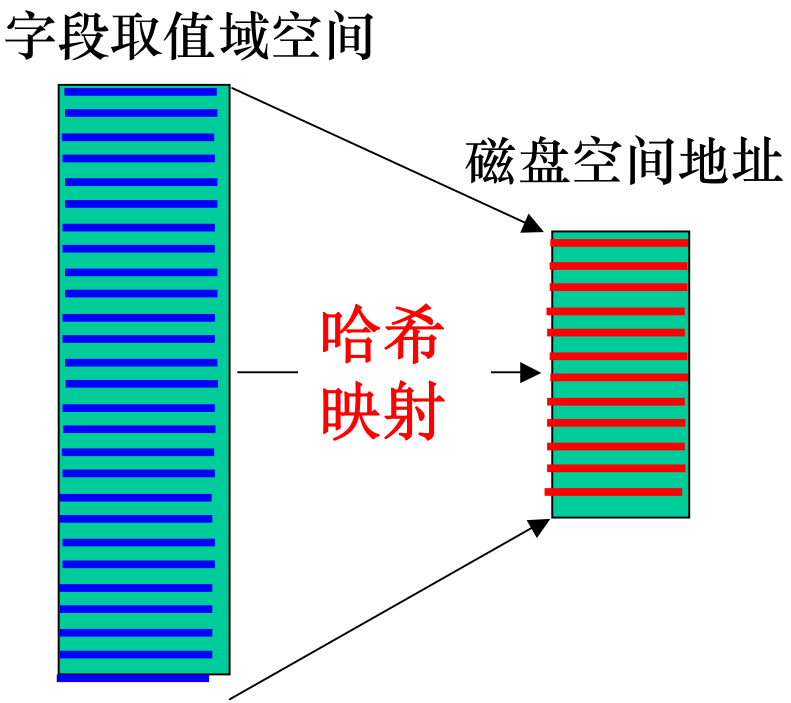
CREATE INDEX studentIndex **ON** student(dno, name);

正确地任用索引：

- **不要对数据量少的表创建索引**。因为读磁盘是以页（8k)为单位进行，如果表数据量只有几个页，索引就没意义。就像很少几页的文章，再给它搞个目录页也没有意义；
- **对长字符串的字段，例如备注字段，不要创建索引，因为压缩比会小**；
- 对访问频繁的外键（作为查询条件，或者联接运算），应对其创建索引；
- (3) 对经常作为查询条件、联接运算、排序、分组、UNION，DISTINCT的字段，对其创建索引；

哈希索引(Hashing index)



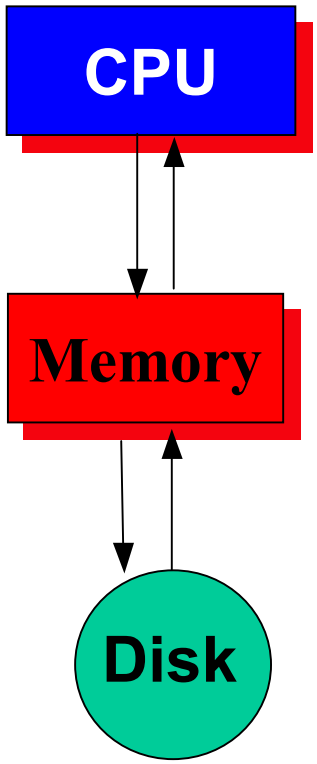


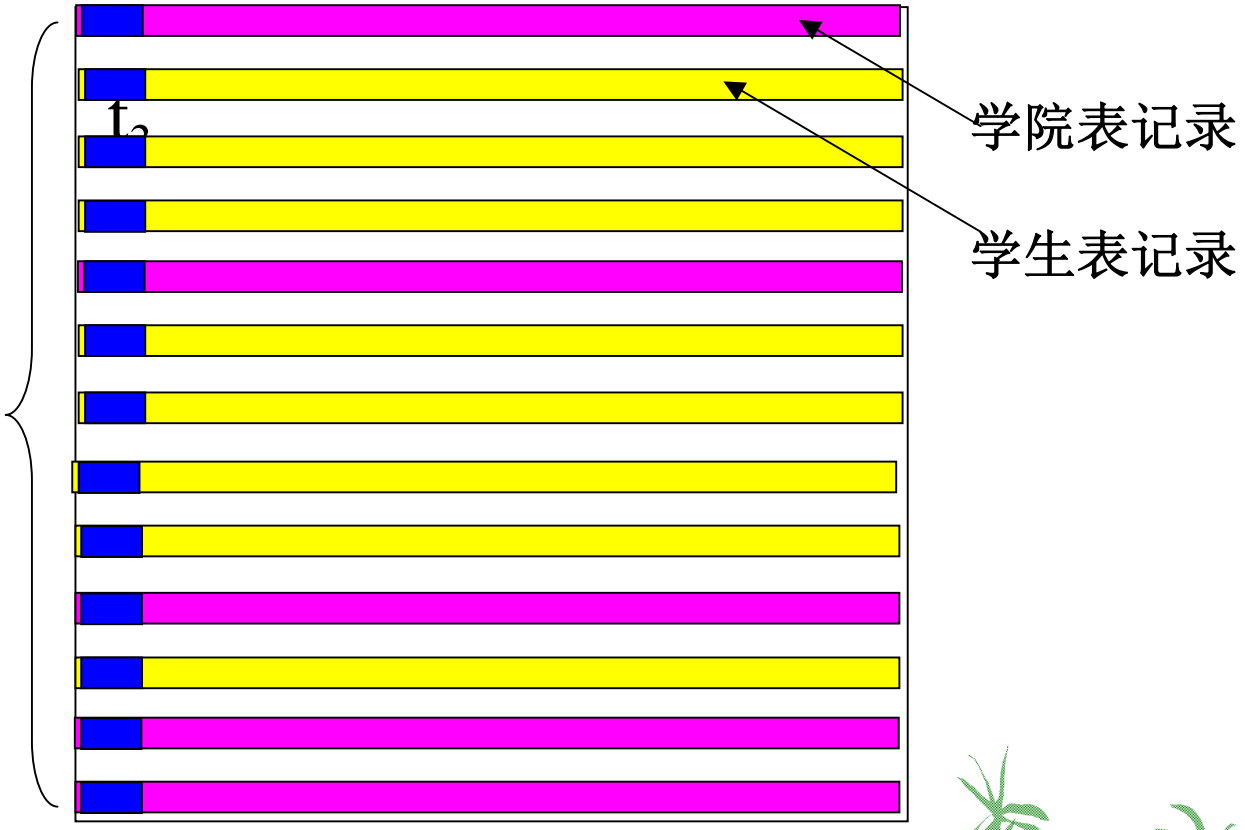
当往表中添加一行记录时，对要哈希的字段计算哈希值，然后把该记录存储在磁盘空间中磁盘地址为该哈希值的地方。

查询数据行时，用户给出字段值，通过哈希计算，就可发现该记录的磁盘空间地址，直接读到对应的记录，不须要一行一行地去比对；

对分布均匀特性的字段，哈希索引可行；例如“学号”字段；

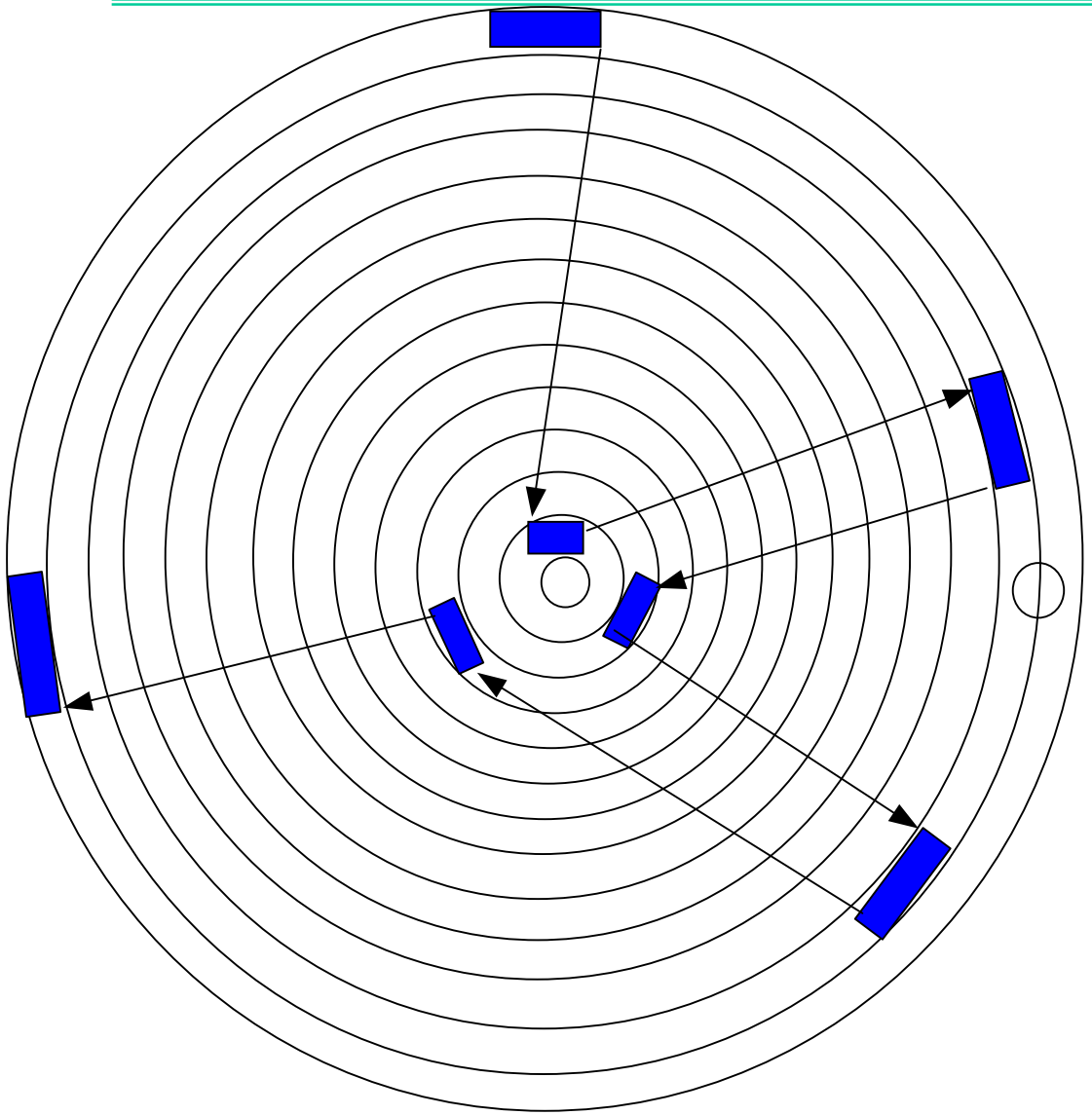
提高表数据查询性能—聚簇(Clustering)





把关系非常紧密，但位于不同表中的行记录，在磁盘上临近存储。当它们做联接运算时，就能迅速得到结果；

提高表数据查询性能—连续空间存储

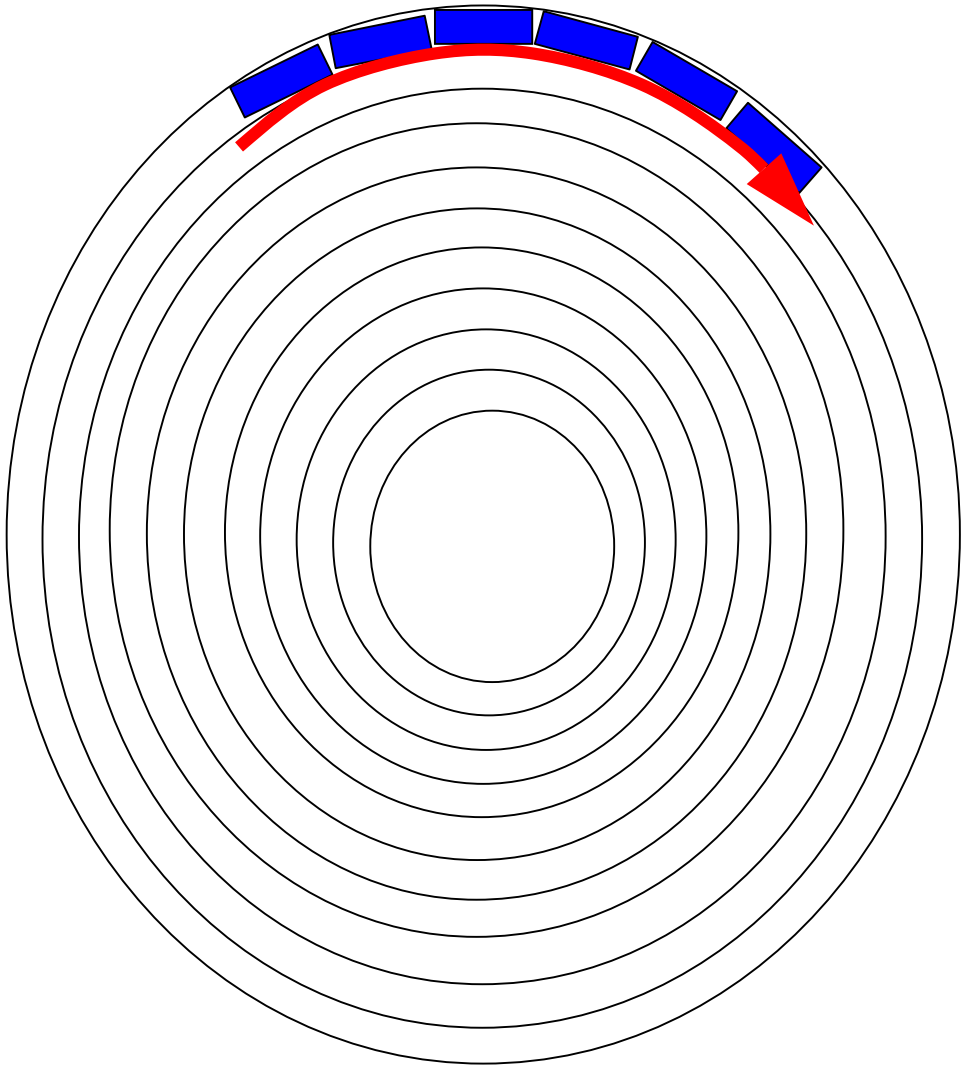


磁盘以片段为单位来存储数据；

一个文件在磁盘上被划分成多个片段，磁盘选取空闲片段来存储表记录；

如果这些片段散布在整个盘面上，**磁头就要反复来回走动，效率低**！

提高表数据访问性能—连续空间存储



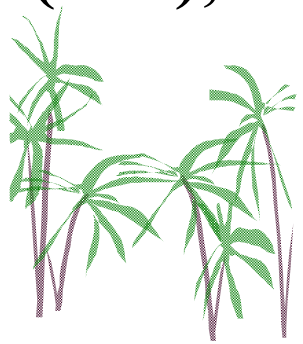
对于表记录，如果**连续地存储**在一个磁道上，当读数据时，数据就可**一个紧挨一个地读到**，访问速度就会大大加快；

你的计算机使用一段时间后，感觉明显变慢，也是这个道理；

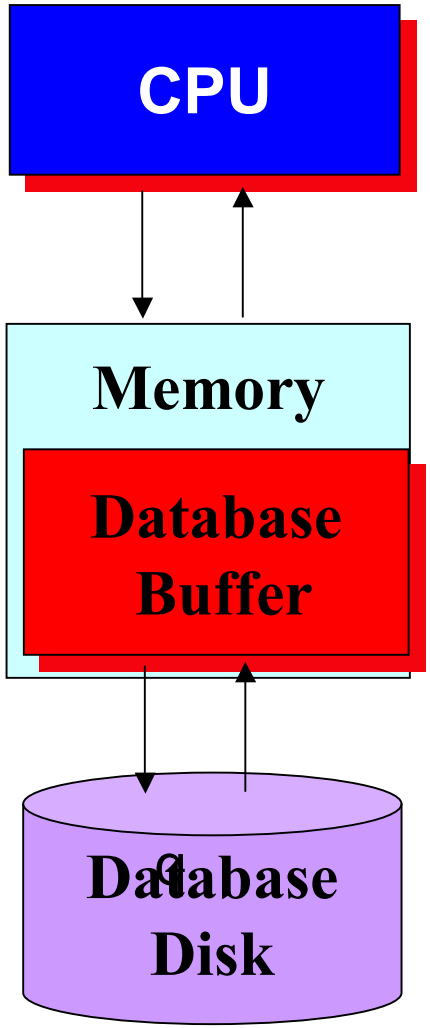
你可以做**磁盘碎片整理**，让每个文件的数据在磁盘上连续存储，这样就克服了**磁头在盘面空间中到处来回移动**，速度大大加快；

连续的磁盘空间存储实现方法

```
CREATE TABLESPACE  basicdata DATAFILE
'c:\oracle\data\basicdata.dbf'  SIZE 100M;

CREATE TABLE employee (
        eno CHAR(5) NOT NULL,
        ename VARCHAR(30),

        PRIMARY KEY (eno),
        FOREIGN KEY (dno)  REFERENCES Dept(dno),

        TABLESPACE  basicdata,
        STORAGE (INITIAL 10M NEXT 10M)
)
```



提高数据处理性能—缓存(buffering)

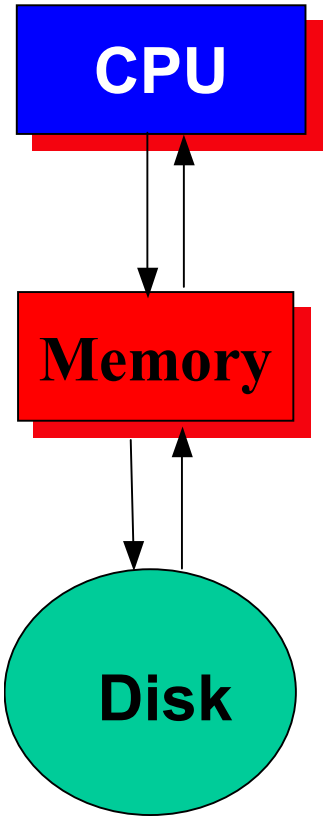


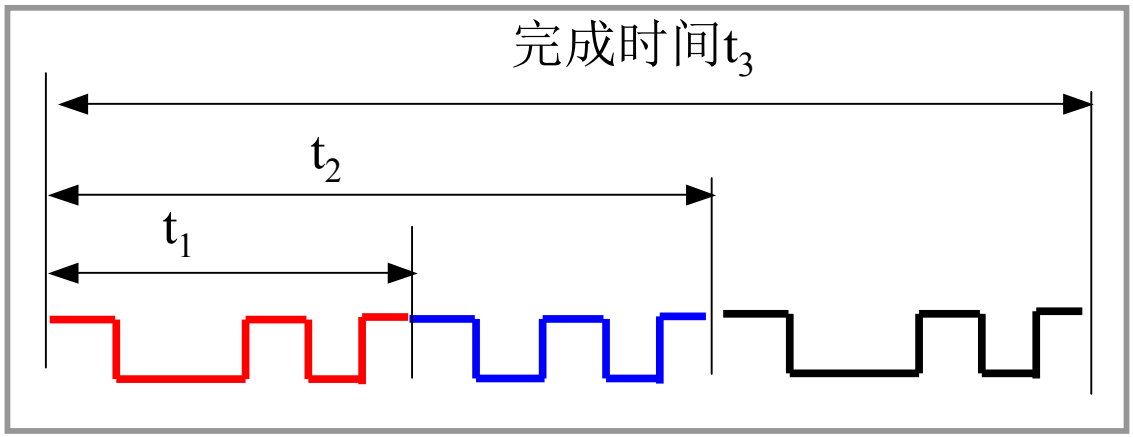
我们知道，我们的钱放在银行最可靠，小偷偷不走，也不会丢失。但是，如果每次用钱时，都到银行去取，很费时间（至少要1小时），效率很低。

为了提高效率，我们会一次到银行取2000放在口袋里，要用钱时，马上就可拿到，效率极高。

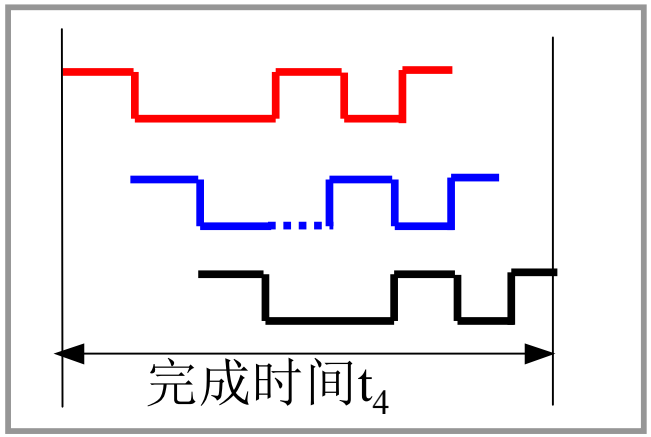
数据存在磁盘上可靠。访问磁盘也是这个道理，很费时间。可事先把数据缓存在内存中，这样数据访问效率和性能就会极大地提高；

提高数据处理性能—并发执行
Concurrent Execution





三个事务串行执行



三个事务并发执行

因为3个组件的速度很不协调，CPU发出一个数据访问指令，内存和磁盘要半天才有响应结果，这段时间CPU就空闲下来了。

我们在家里做菜也是这样，一个人同时做几样菜：一样菜要煮一段时间时，人就可去切另一样菜。效率大大提高。

提高数据处理性能—查询优化 (Query Optimizing)

对于处理一个查询请求任务，就像做一个数学题一样，有多种方法来解答，有的方法简单高效，有的方法复杂，代价大；查询优化就是要找到并使用那种最简单最高效的方法，来处理用户的查询请求。

例如用户的请求是：查出在北京的所有营业点的负责人：

SELECT * FROM emp e, dept d **WHERE** e.dno = d.dno **AND**
(e.title = 'Manager' **AND** d.city = '北京');

有3种代数运算处理方法：

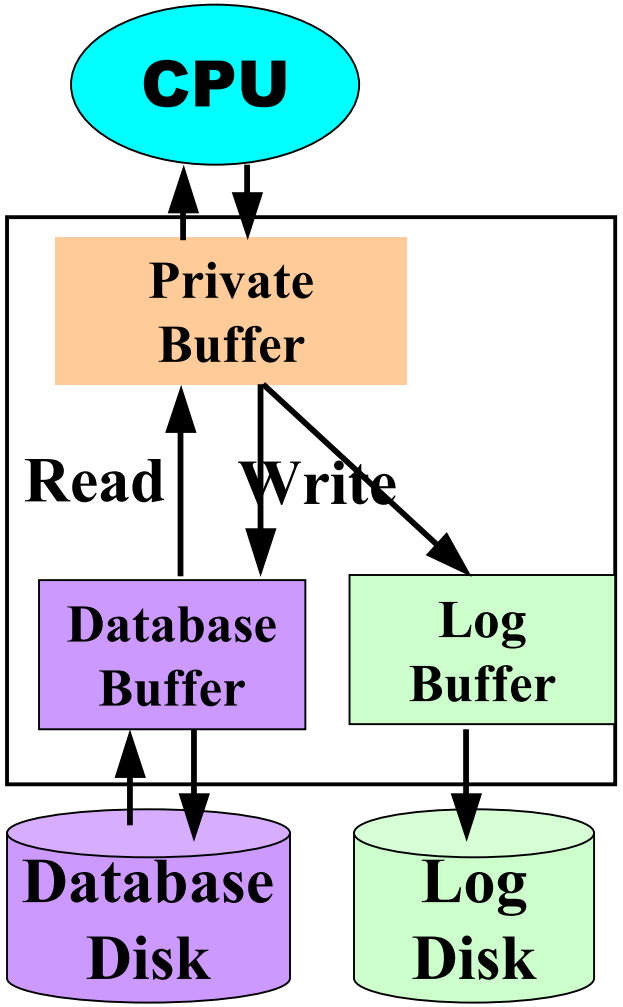
(1) $\sigma_{(title='Manager') \wedge (city='London') \wedge (emp.dno=dept.dno)}$ **(emp × dept)**

(2) $\sigma_{(title='Manager') \wedge (city='London')}$**(emp** $\bowtie_{emp.dno=dept.dno}$ **dept)**

(3) **(**$\sigma_{title='Manager'}$**(emp))** $\bowtie_{emp.dno=dept.dno}$ **(**$\sigma_{city='London'}$ **(dept))**

我们可以看到第1种方法代价最大，第3周方法代价最小，最简单。

提高数据处理性能—配置单独专用的日志磁盘



当日志和数据库配置在一个物理磁盘上时，磁头一会要去数据库区读写要处理的数据，马上又要跳到日志区写日志，接着又要折回数据库区读写要处理的数据。**磁头来回长距离倒腾，疲于奔命，效率极低，性能极差**。

当配置单独的日志磁盘时，正常运行时，日志具有只写不读的特点，而且是递增，**磁头可一个挨一个地写，不用来回移动，性能得到极大提高**。

高级数据库技术

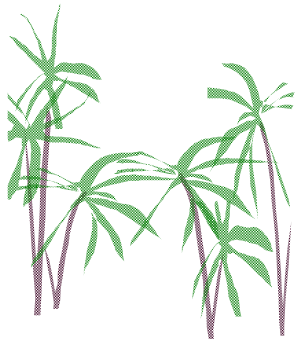
Advanced Database technology

(三）安全技术

湖南大学
信息科学与工程学院

杨金民

2015. 09



用户的数据操作

银行数据库表

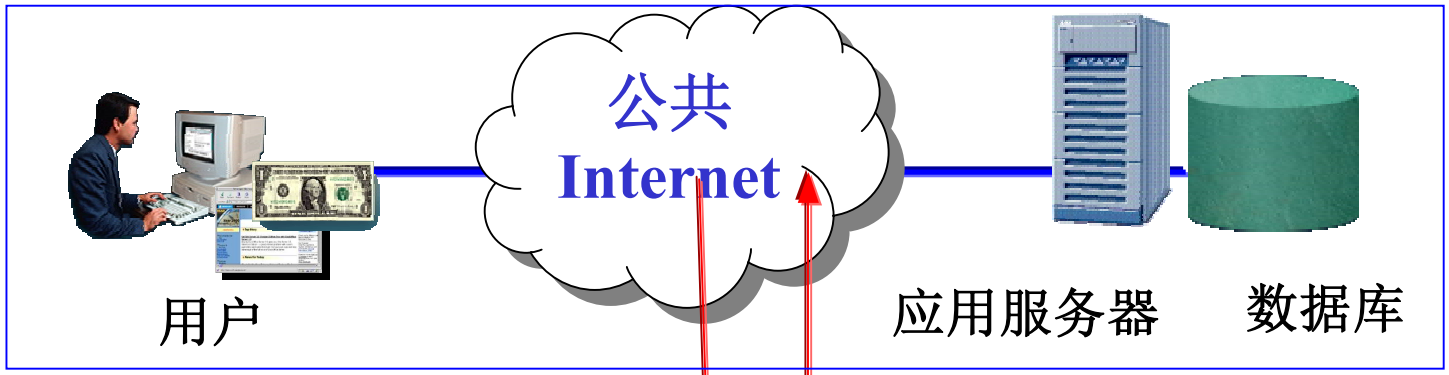
account

Name	accountNo	identityNo	balance
周山	2008043101	430104198008064311	1200
汪兵	2008043214	430104197612274810	80,000
张珊	2008043332	430104196902114933	137,000

例子:对于网银系统，汪兵要转账1万元给周山，从汪兵的电脑上发给银行数据库系统的操作请求为：

```
TRANSACTION BEGIN
      UPDATE account SET balance = balance + 10000 WHERE accountNo
    =‘2008043101’;
      UPDATE account SET balance = balance - 10000 WHERE accountNo
    =‘2008043214’;
COMMIT；
```

数据库安全问题及技术



抵赖

偷看

假冒

篡改

加密技术

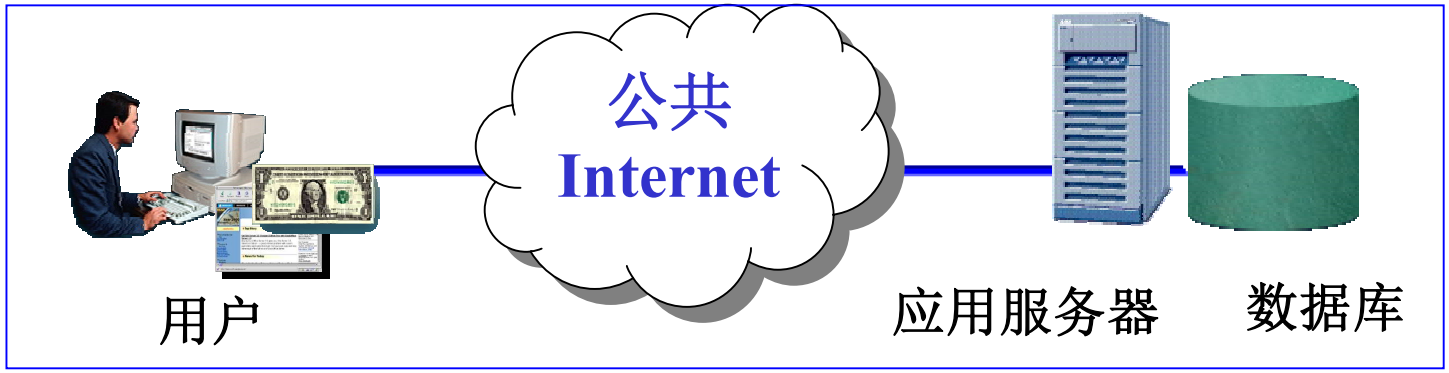
非法数据访问：

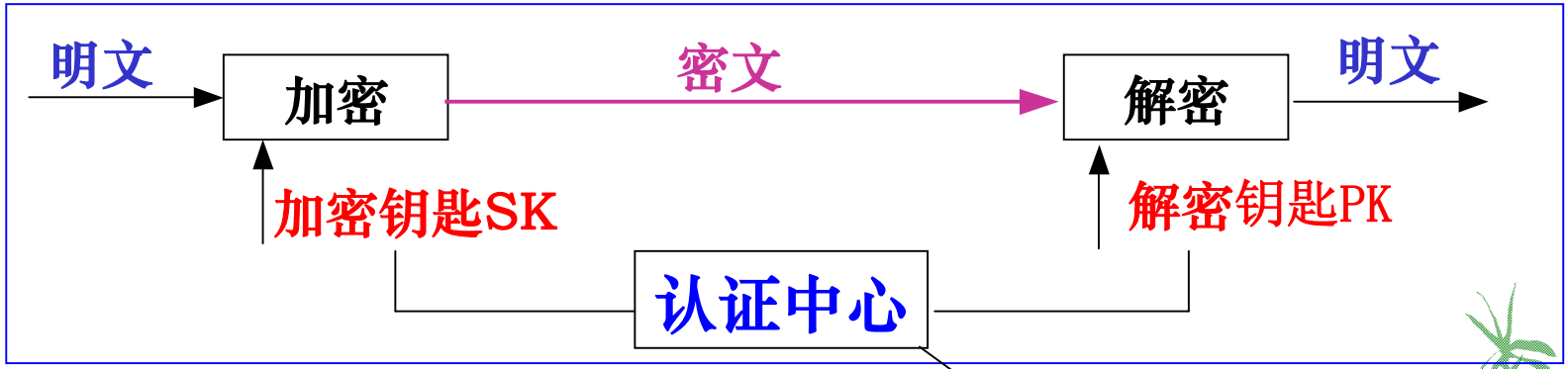
- 读取非允许的数据；
- 改/删/加非允许的数据；

用户、权限管理，审计

安全管理—运输中泄密问题

解决：抵赖、偷看、假冒、篡改这四个问题。





非对称技术：

用一个东东(SK)加密的密文,要使用另一个不同的东东（PK）来解密；

起到公证处的作用

安全管理—密码技术

- (n,e)构成**公钥PK**，可以告诉别人；
- (n,d)构成**私钥SK**，自己保留，不让任何人知道；

给别人发送的信息使用SK加密，别人能用PK解开就证明信息是由你发送的，构成了**签名机制**；

别人给你发送信息时使用PK加密，这样只有拥有SK的你能够对其解密。

安全性在于：对于一个大数n，已知公钥PK(n,d)，但**无法获得私钥**SK(n,e)；

密码约束条件：

- 找两素数p和q，取n = p * q，再取t=(p - 1) * (q - 1)
- 取任何一个数e, 要求e < t并且e与t互素，取d使得d * e % t == 1

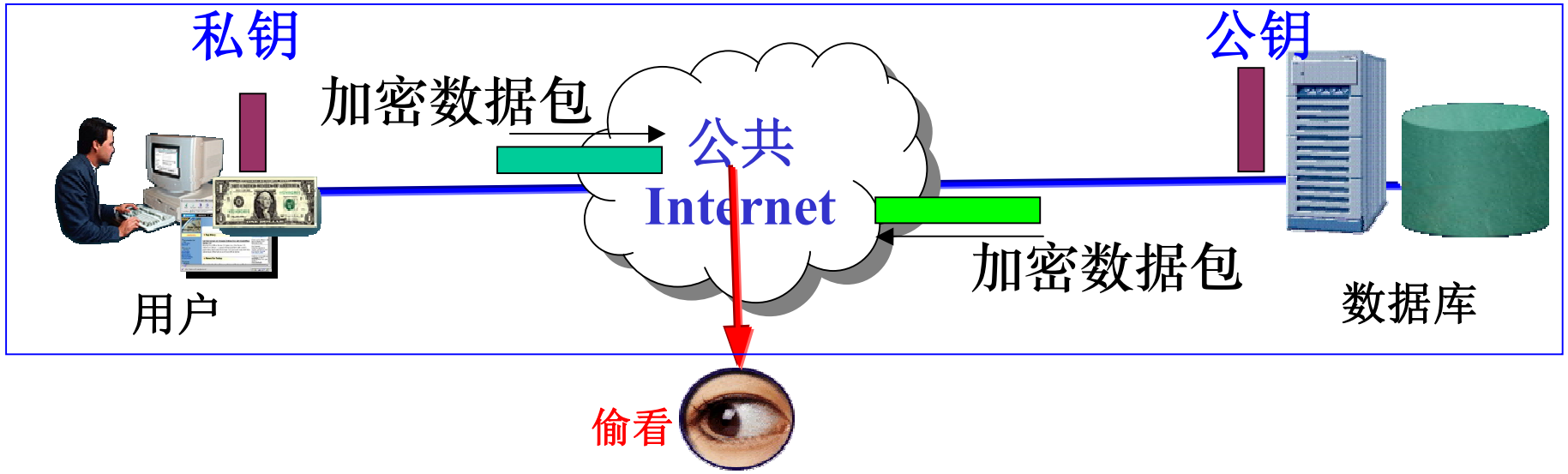
✓ **加密**：密文 = (明文**d) % n;

✓ **解密**：明文 = (密文**e) % n;

例子：PK(33,3); SK(33,7); m=7;

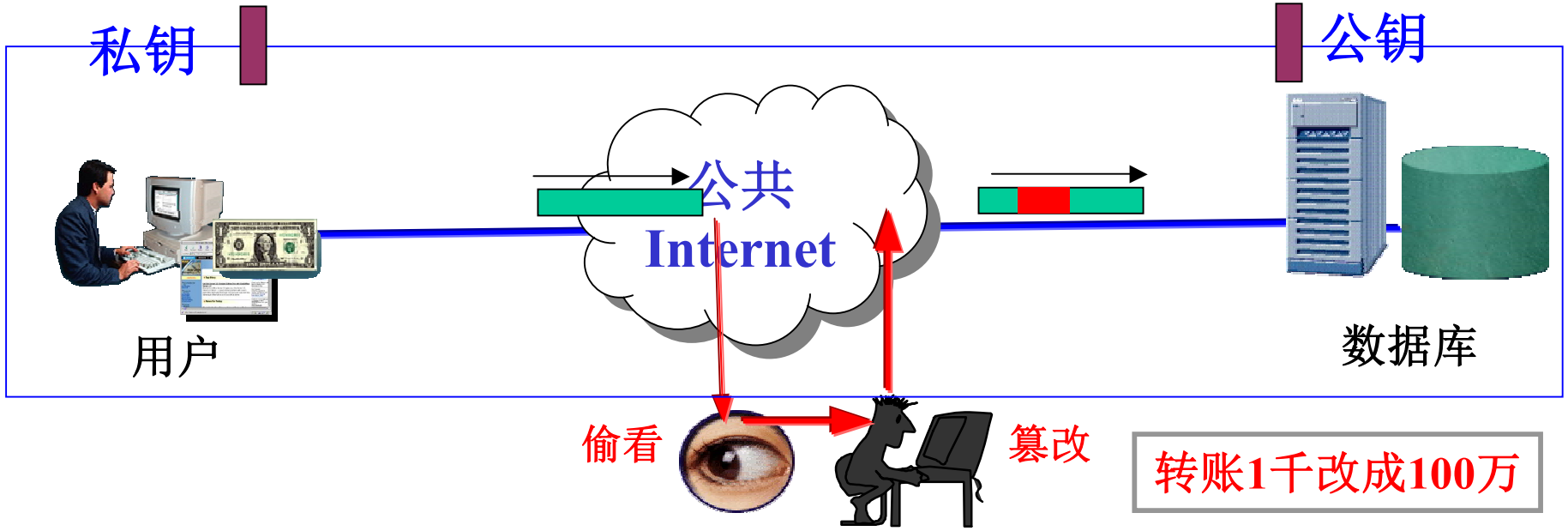
b • c % n = (b % n) • (c % n) % n

安全管理—偷看问题的解决



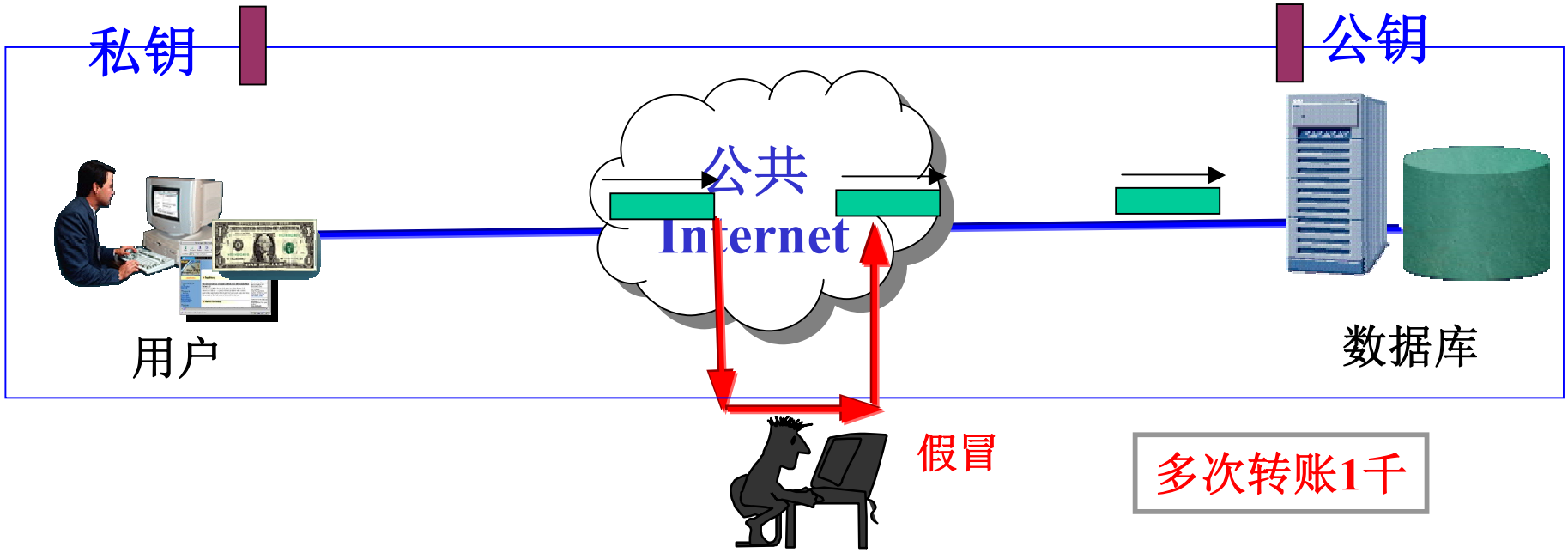
用户对要发送给数据库的请求使用私有秘钥加密，仅仅只有持有公有秘钥的数据库管理系统能解密。反过来也是如此，数据库管理系统对发给某用户的数据，使用该用户的公有秘钥加密，仅仅只有持与之对应的私有秘钥的用户能够解密，于是解决了数据在互联网上传输过程中被偷看的问题。

安全管理—篡改问题的解决



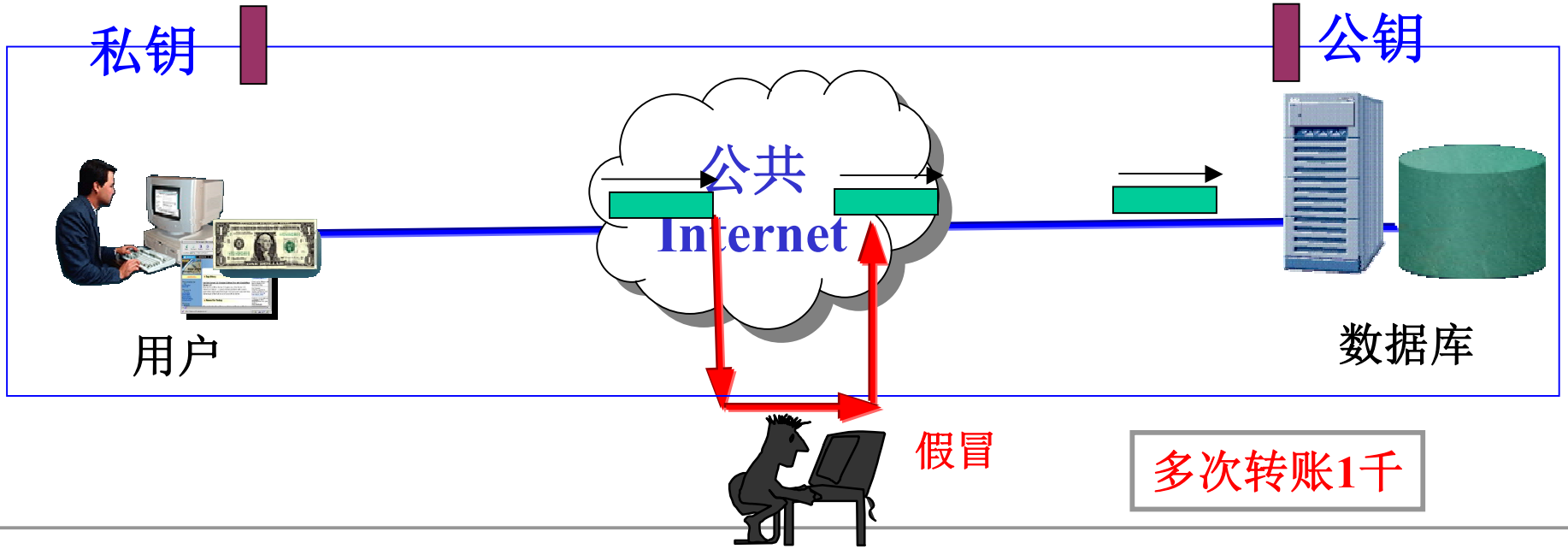
使用某用户的公有秘钥对数据加密，使用公有秘钥并不能对其解密，于是，即使数据库管理员把用户的公有秘钥泄露出去了，不法分子虽然能够解密用户发给数据库管理系统的消息，也没有办法篡改用户的消息，因为篡改后因为没有用户的私有秘钥，也就没有办法加密，于是解决了篡改问题。

安全管理—假冒问题



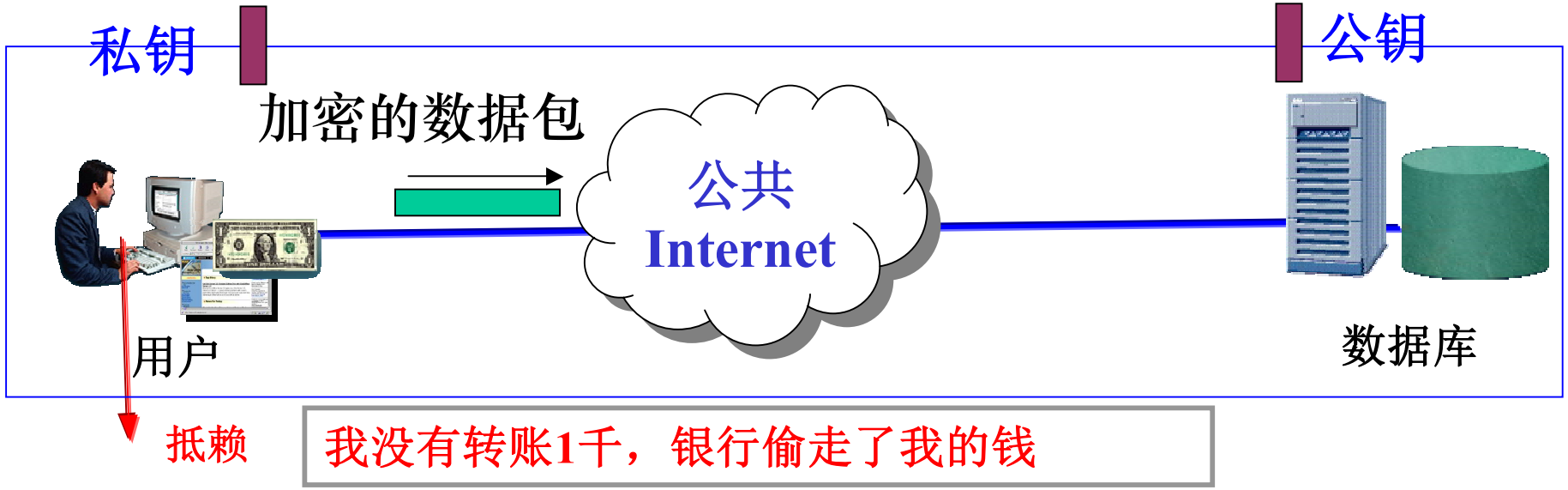
这样即使不法分子在internet上截获了用户的数据操作请求包，并进行复制，然后再多次发给数据库，企图让数据库多次重复执行该请求，达到其不可告人的牟利目的。

安全管理—假冒问题的解决



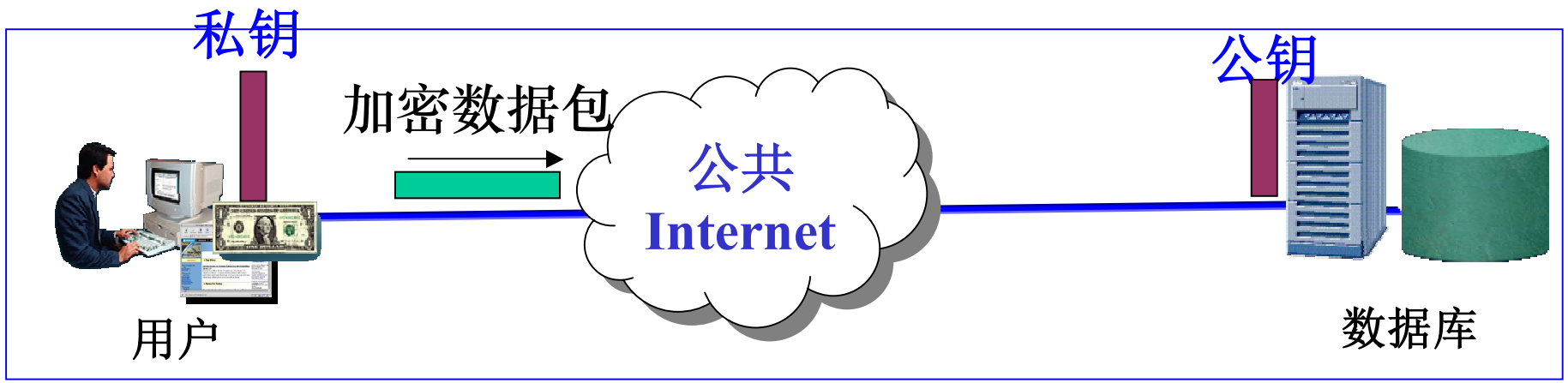
假冒问题的解决则是是使用验证码，用户每次访问数据库，都是先请求数据库给定一个验证码，然后在请求中附加上这个数据库给定的验证码打包加密发送给数据库。数据库收到用户请求之后，解析出其附带的验证码，在自己维护的有效验证码表中查找是否存在该验证码，如果有，则认定该请求合法有效，处理该请求，然后在有效验证码表中删除该验证码。但是当数据库第二次收到该请求时，该请求附带的验证码已被删除，于是，就可认定该请求重复无效。

安全管理—抵赖问题的解决



私有秘钥加密的数据，仅只有使用公有秘钥才能对其解密。因此，对于某个用户，凡是能够用其公有秘钥能解密的数据，肯定是由该用户发送过来的，于是解决了用户对自己所做操作进行抵赖的问题。

传输安全问题的解决的实际方式

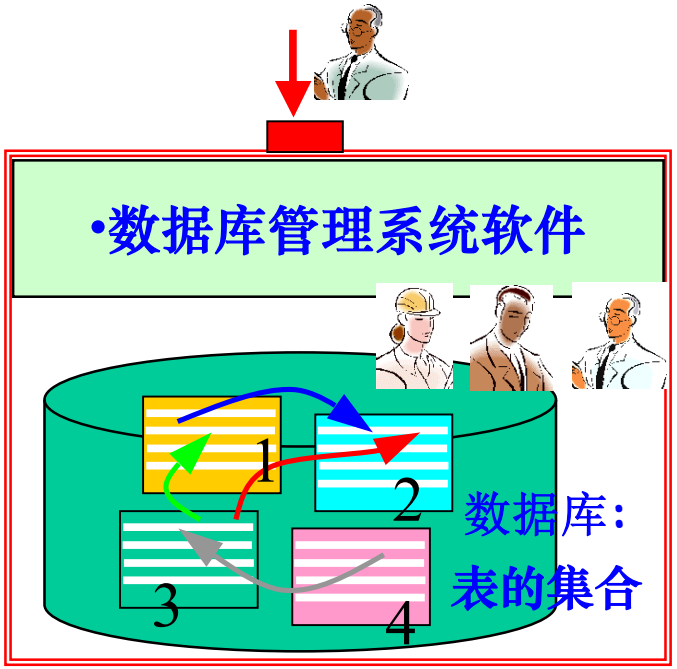


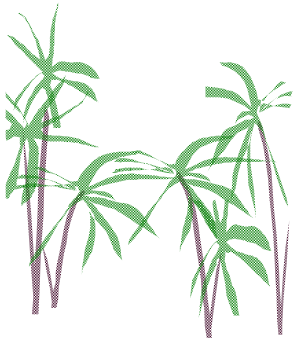
U盾

私钥太长，不好记。于是，采用U盾，把私钥存在U盾（U盘）上，实现了简单性。这就要求用户必须保管好U盾，不能借给别人，也不要砸死外面别人的机器上使用，以防泄露私钥，给不法分子以可乘之机。银行事先把你的公钥在认证中心进行了登记，以防后面的法律纠纷。

安全管理—数据库内部安全管理

- **联接控制**：只允许**授权的用户**与数据库建立联接，访问数据库；
- **权限控制**：只允许**用户**访问已**被授权的数据**；
- **审　　计**：对用户访问数据库的过程进行**跟踪记录**，一旦发生了安全问题，便可追踪**溯源**，破案；





安全管理—数据库访问权限控制

安全机制的三个概念:

- **用户**：（用户名，口令）；
- **权限**： 读，修改，删除，添加；
- **数据对象**：表（列）；

- **授权操作**：对于某个**数据对象**，将**某种权限授予**给**某个用户**；
- **收权操作**：将**某个用户**对**某个数据对象**拥有的**某种权限收回**；

员工表emp

name	eno	Birthdate	title	salary	dno
周　山	E62	1-Jun-81	EE	5000	D01
杰克孙	E63	1-Aug-82	PR	4500	D07
菲利普	E64	4-May-83	ME	4000	D07
奥巴马	E72	1-May-71	SA	8500	D01

用户表

name	pwd
Philip	123
Machel	tree

工程公司的数据库样例

部门表dept

dNName	dno	deanNo
工程1部	D01	E72
设计部	D07	E62
工程2部	D04	E63
财务部	D05	E64

项目表proj

pName	pNo	budget	dNo
湘湖大桥	P01	25000	D01
明珠广场	P03	789	D02
星明大夏	P17	6500	D02

关系

关系

员工表emp

name	eno	Birthdate	title	salary	dno
周　山	E62	1-Jun-81	EE	5000	D01
杰克孙	E63	1-Aug-82	PR	4500	D07
菲利普	E64	4-May-83	ME	4000	D07
奥巴马	E72	1-May-71	SA	8500	D01

工程公司的数据库样例

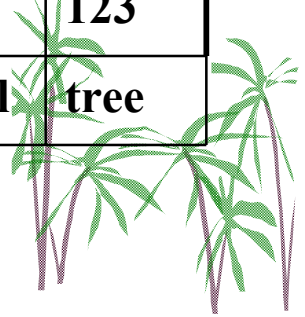
员工参加项目情况表workon

eNo	pNo	duty	hours
E63	P17	水电监理	90
E62	P03	钢材供应	56
E72	P02	施工指挥	56
E64	P17	木工	77
E62	P02	门窗设计	87
E78	P17	外部协调	56

外键：1）eNo;员工工号
2）pNo;项目编号

用户表

name	pwd
Philip	123
Machel	tree



数据库安全访问权限控制机制

- 数据库访问安全控制模型中包含有4个概念:
  - **注册用户;**
  - **数据库中的数据对象：表和视图;**
  - **数据对象的属主;**
  - **访问权限；.**
- 用户标识包括（用户 id, 和密码password）;
- 当一个用户创建一个表或者视图时，他便成为了这个被创建的对象的属主，拥有全部权限，成为**授权树**的”**根节点**”;
- 对象的属主可以把权限授予给他人，也可把权限收回;

权限

权限是指对数据库中的表的操作权限，包括：

1) SELECT – 查询表中的数据

2) INSERT – 往表中添加行记录

3) UPDATE – 修改表中数据

4) DELETE – 删除表中行记录

5) REFERENCES – 在完整性控制中引用其他表；

其中：

- INSERT，UPDATE 和 REFERENCES 能进一步限定到表中的字段一级；
- 而其它权限只能限定到表一级；

SQL授权例子

- 对于“部门表dept”,将“查询”权限授予给“所有用户”:

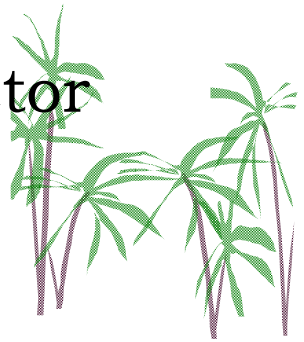
```
GRANT SELECT ON dept TO PUBLIC;
```

- 对于“员工表Emp”的工资字段salary，将其“查询和修改”权限仅仅只授给角色“经理”或者“主任”:

```
GRANT SELECT, UPDATE(salary) ON Emp TO Manager,Director;
```

- 对于“项目表proj”，将其”全部权限”授给角色 “主任”，并允许主任对其拥有进一步授权给他人的权限:

```
GRANT ALL PRIVILEGES ON Proj TO Director
WITH GRANT OPTION;
```



通常，权限是不能够传递授予的;

数据操作的权限要求

对于下面的数据操作，需要什么样的权限?

```
UPDATE Emp SET salary=salary*1.1 WHERE eno IN (
SELECT eno FROM WorksOn WHERE hours > 30);
```

```
DELETE FROM dept WHERE dno NOT IN
(SELECT dno FROM WorksOn);
```

```
INSERT INTO WorksOn (eno,pno) VALUES ('E5','P5');
```

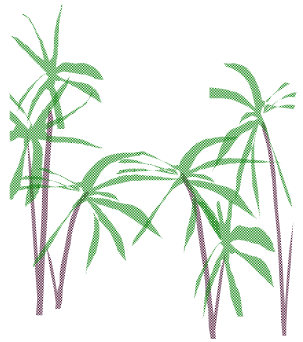
收回权限操作的SQL

禁止"所有用户"对"部门表dept"执行"查询"操作:

REVOKE SELECT ON dept **FROM PUBLIC**;

对于"员工表emp"，收回"用户Joe"的"任何访问权限":

REVOKE ALL PRIVILEGES ON Emp **FROM** Joe;



视图能用来增强权限控制

对于普通员工角色" Staff", 就员工表emp, 可以对birthdate和salary两个字段之外的所有字段拥有查询权限:

如何来实现这一规则?

1）就员工表emp创建一个视图:

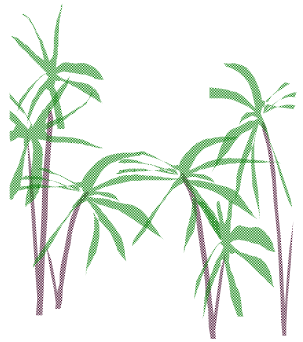
```
CREATE VIEW EmpView AS
SELECT eno, ename, title, supereno, dno FROM emp;
```

2） 对于EmpView，将查询权限授给角色" staff"

```
GRANT SELECT ON EmpView TO Staff;
```

3）收回角色" staff"对emp表的查询权限:

```
REVOKE SELECT ON Emp From Staff;
```



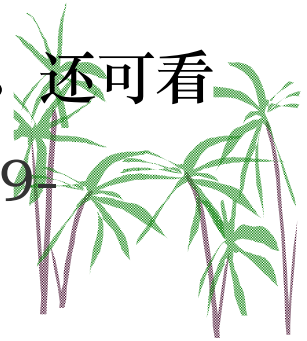
安全管理—数据库访问的审计

对用户访问数据库的过程进行**全程跟踪记录。**

审计记录表：

返回码	操作码	用户名	操作用户名	访问主机	时间
1017	100	SCOTT	WPRATA-BR	10.126.53.159	2004-09-22:11:08:00
1017	100	SCOTT	WPRATA-BR	10.126.53.159	2004-09-22:11:08:03
1017	100	SCOTT	WPRATA-BR	10.126.53.159	2004-09-22:11:08:09
0	101		WPRATA-BR	10.126.53.159	2004-09-22:11:09:21

通过检查审计表中的记录，1017的含义为错误的用户名口令，可以清楚地看出用户WPRATA-BR尝试破译用户SCOTT的口令。还可看到，用户是在 IP地址为10.126.53.159的机器上于2004-09-22:11:08:00干的。



高级数据库技术

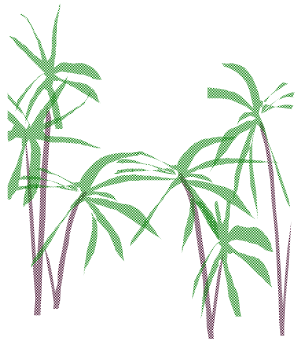
Advanced Database technology

(四)　分布式数据库

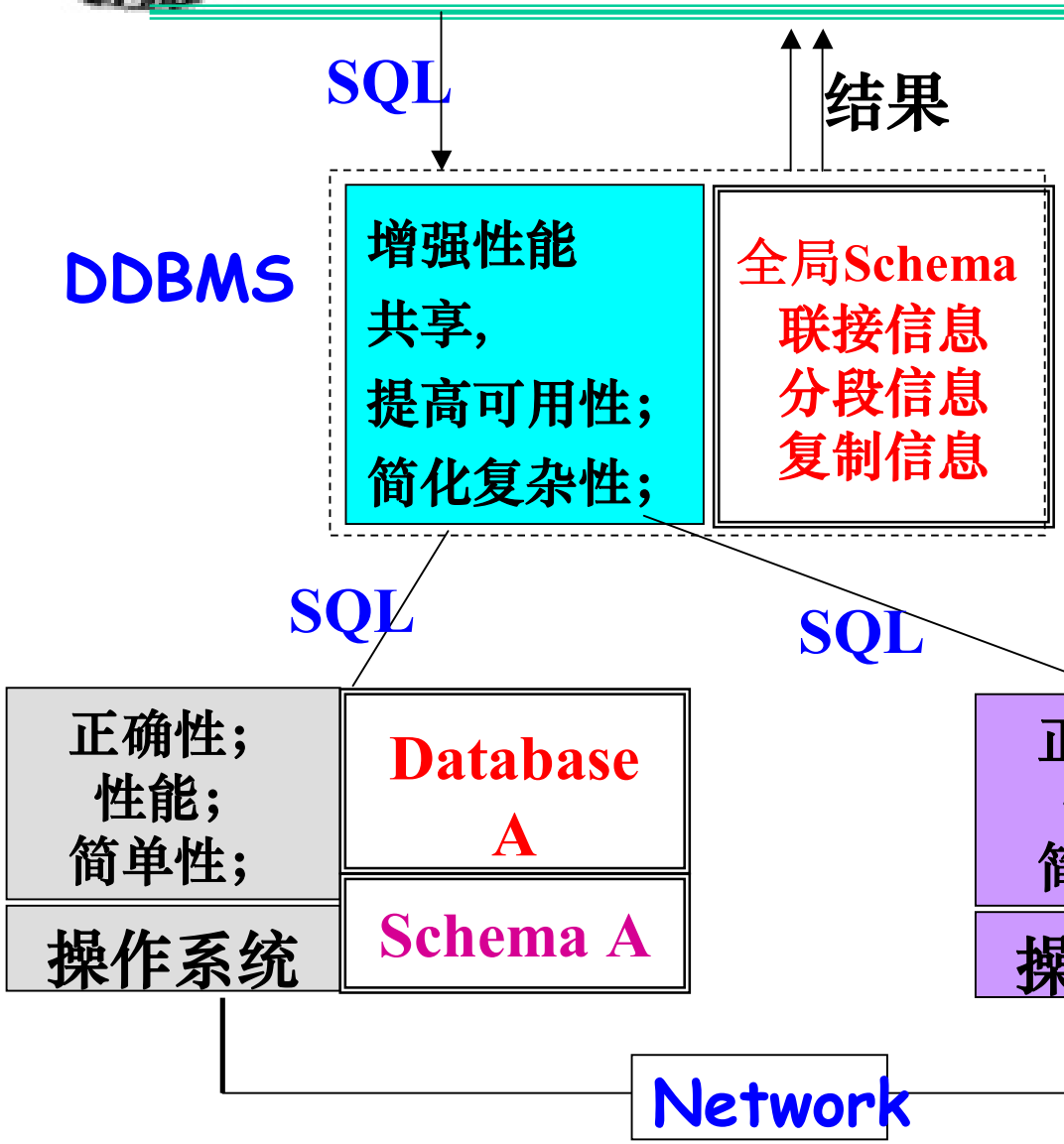
湖南大学
信息科学与工程学院

杨金民

2015. 09



4.1 分布式数据库DDB



- DDB其实质是一个中介；实体数据由实体数据库维护；
- 对用户来说，它与通常的数据库没有丝毫差异；
- 内部维护着实体数据库的语义信息，其功能就是分解和分发任务，收集和组合结果；

4.2 分布式数据库中数据分段

PROJ

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	P2 Database Develop	135000	
P3	CAD/CAM	250000	
P4	Maintenance	310000	Paris
P5	CAD/CAM	500000	Boston

PROJ <$200000

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	P2 Database Develop	135000	

PROJ2 >=$200000

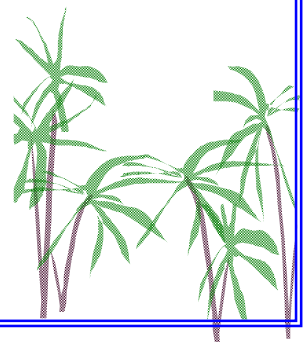
PNO	PNAME	BUDGET	LOC
P3	CAD/CAM	250000	
P4	Maintenance	310000	Paris
P5	CAD/CAM	500000	Boston

PROJ2

PNO	PNAME	LOC
P1	Instrumentation	Montreal
P2	P2 Database Develop	
P3	CAD/CAM	
P4	Maintenance	Paris
P5	CAD/CAM	Boston

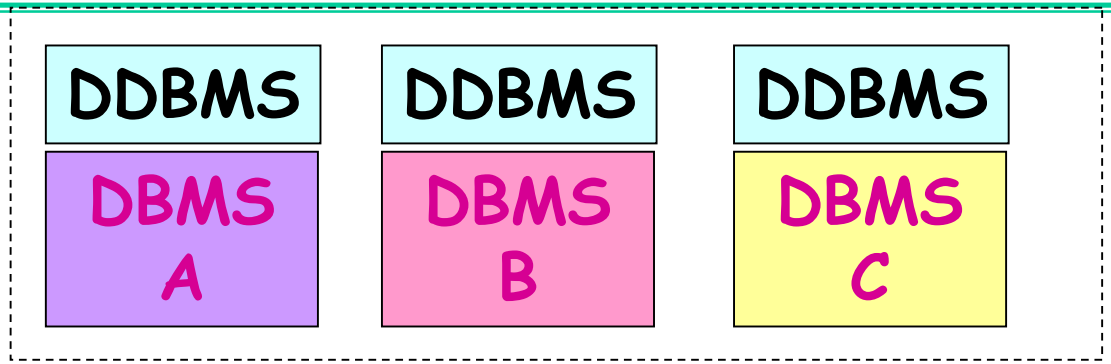
PROJ1

PNO	BUDGET
P1	150000
P2	135000
P3	250000
P4	310000
P5	500000

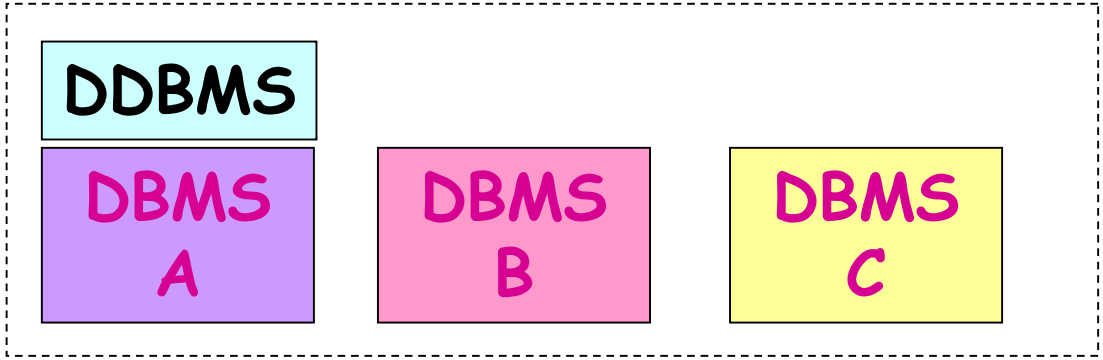


4.3 DDBMS形式

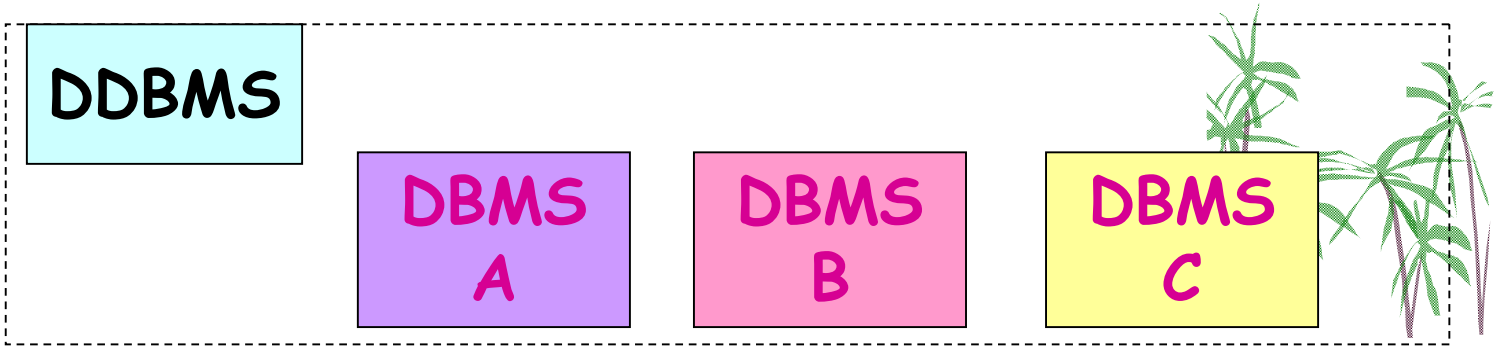
形式 1



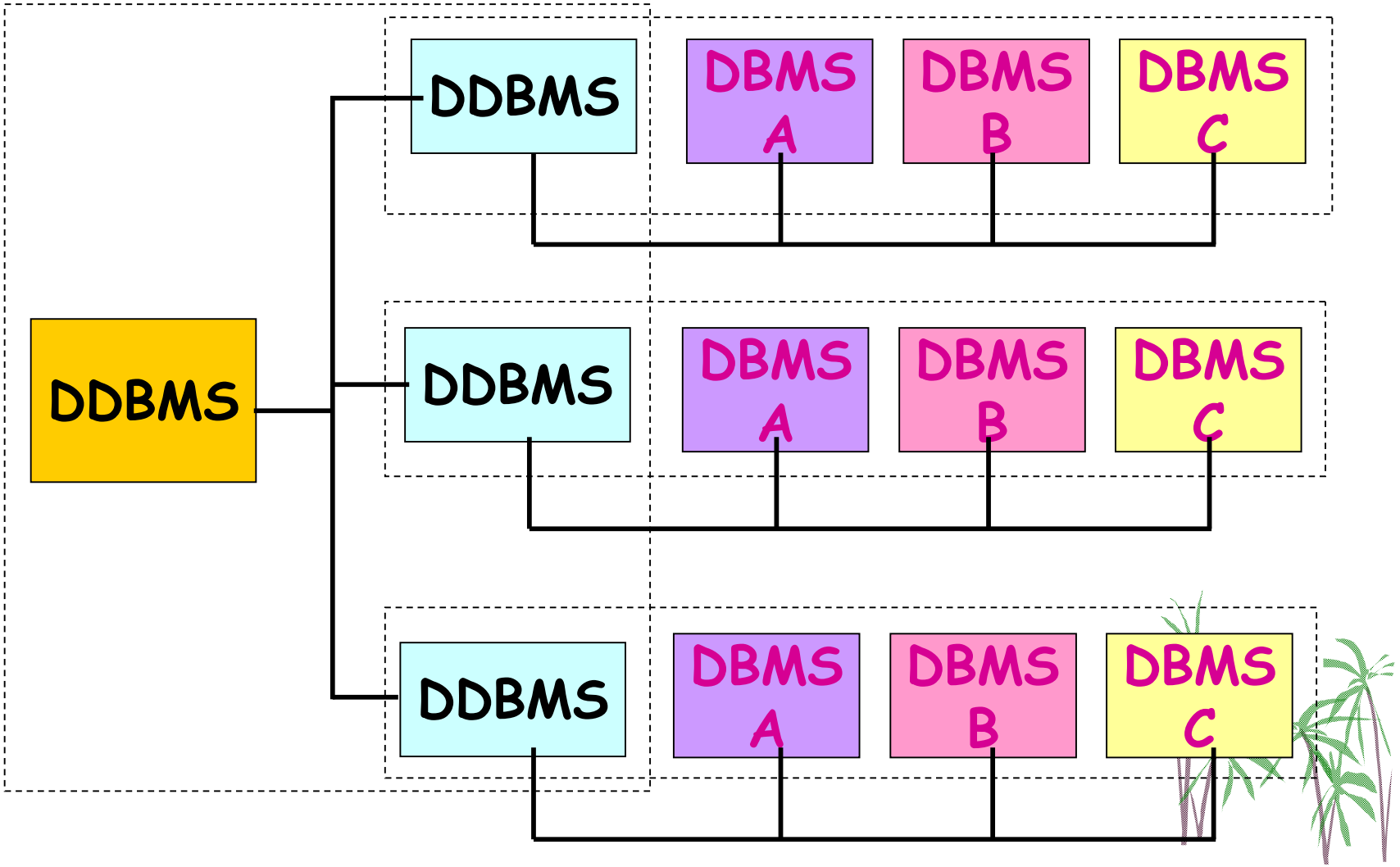
形式2



形式3

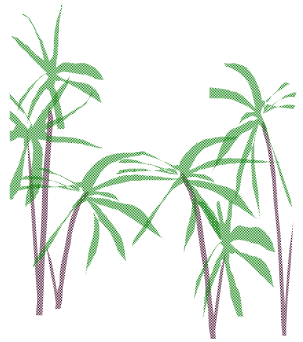


4.3 DDBMS形式(cont.)



4.4 分布式数据库中的透明问题

- 网络透明(Network Transparency)
- 复制透明(Replication Transparency)
- 分段透明(Fragmentation Transparency)



高级数据库技术

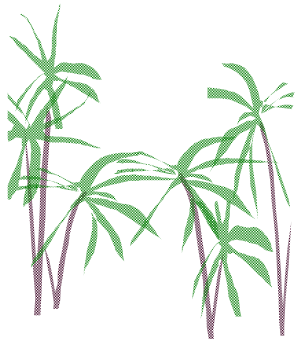
Advanced Database technology

(五) 数据仓库

湖南大学
信息科学与工程学院

杨金民

2015. 09



数据处理的类型

OLTP
On-line Transaction Processing

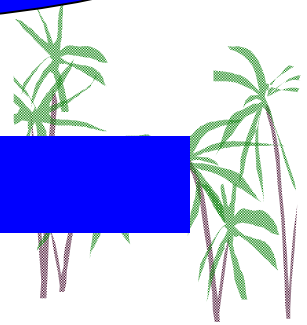
处理故障

日常事务工作

OLAP
On-line Analyzing Processing

挖掘和提炼信息

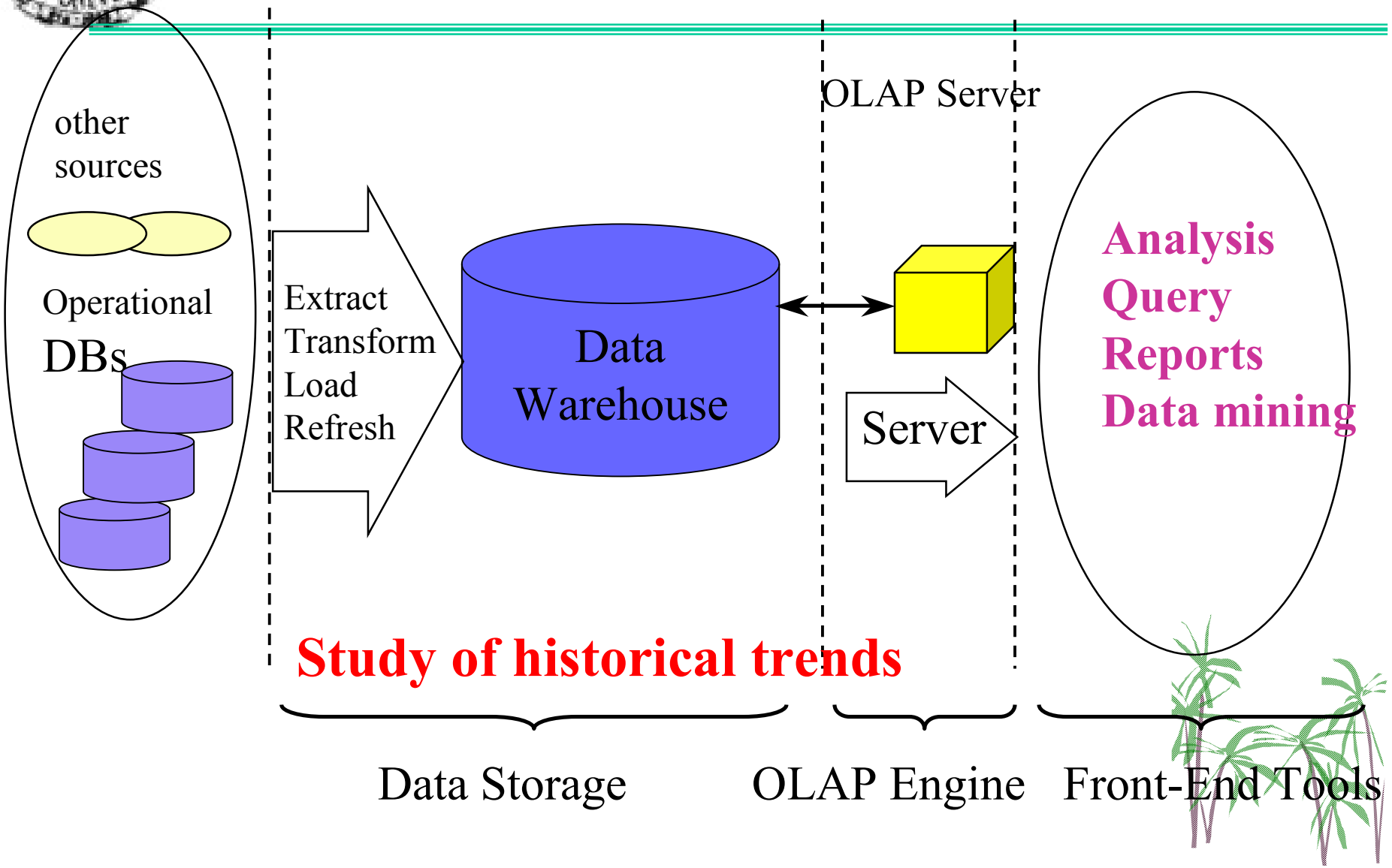
决策支持



基本概念

- **数据分析（Data analysis）**：对数据进行统计分析，找出规律特征；.
- **数据挖掘（Data mining）**：从大量的数据中区自动地发现隐藏的特征模式，为决策作支撑；
- **数据仓库（data warehouse ）**：将来自多个数据源的数据，以统一的模式，集中存储在某个站点上。其特征：历史数据，海量的数据，数据只添加，只读，无修改；.
- **信息提取（Information-Retrieval）**：对Internet上的无结构的文本数据进行查找.

DBs, Warehouse, & OLAP



数据仓库的开发过程

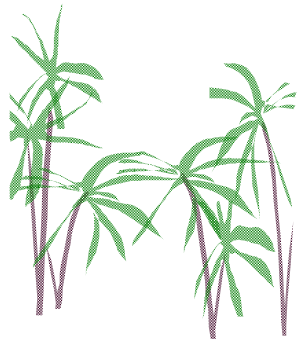
(1) 确定建立数据仓库**要解决的问题**，明确主题下的**查询分析需求**；

(2) 选择数据仓库**软件平台**：包括数据库、建模工具、分析工具。

(3) 建立数据仓库的**逻辑模型**，在选定的软件平台下**加以实现**；

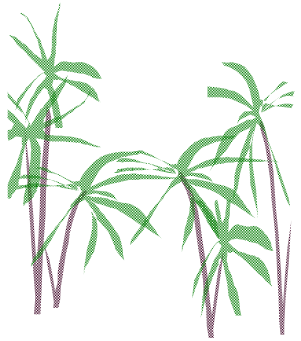
(4) 对来自不同数据源的**异构数据**，根据数据模型进行**清洗**和**转换**，**汇聚**到数据仓库；

(5) 开发数据仓库的**分析应用**；



SQL国际标准的延伸

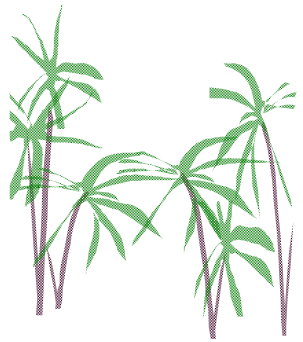
- **SQL-92** 没有考虑OLAP，例如，
  - Data cube
  - Complex aggregates (median, variance)
  - binary aggregates (correlation, regression curves)
  - ranking queries (“assign each student a rank based on the total marks”
- **SQL:1999 OLAP** extensions provide a variety of aggregation functions to address above limitations



Ranking （排名）

- 例如，对学生基于成绩排名，原有SQL没有此功能：stu-marks(stu-id, marks) :

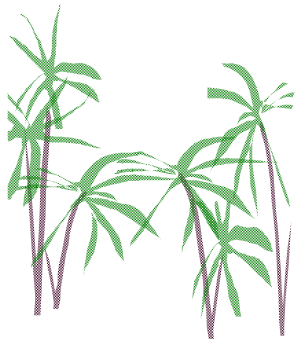
select *stu-id*, **rank ( ) over (order by** *marks* **desc) as** *s-rank* **from** *stu-marks* **order by** *s-rank;*



Ranking (Cont.)

- 例如， 先对员工工资排序，再均分成三档，求每档的平均工资:

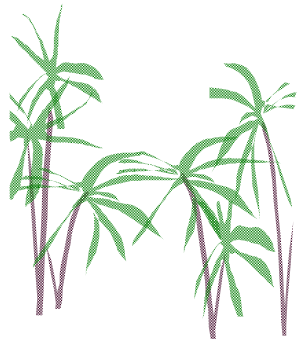
```
select threetile, avg(salary)
from (
        select salary, ntile(3) over (order by salary) as threetile
from employee
        ) as s
group by threetile;
```



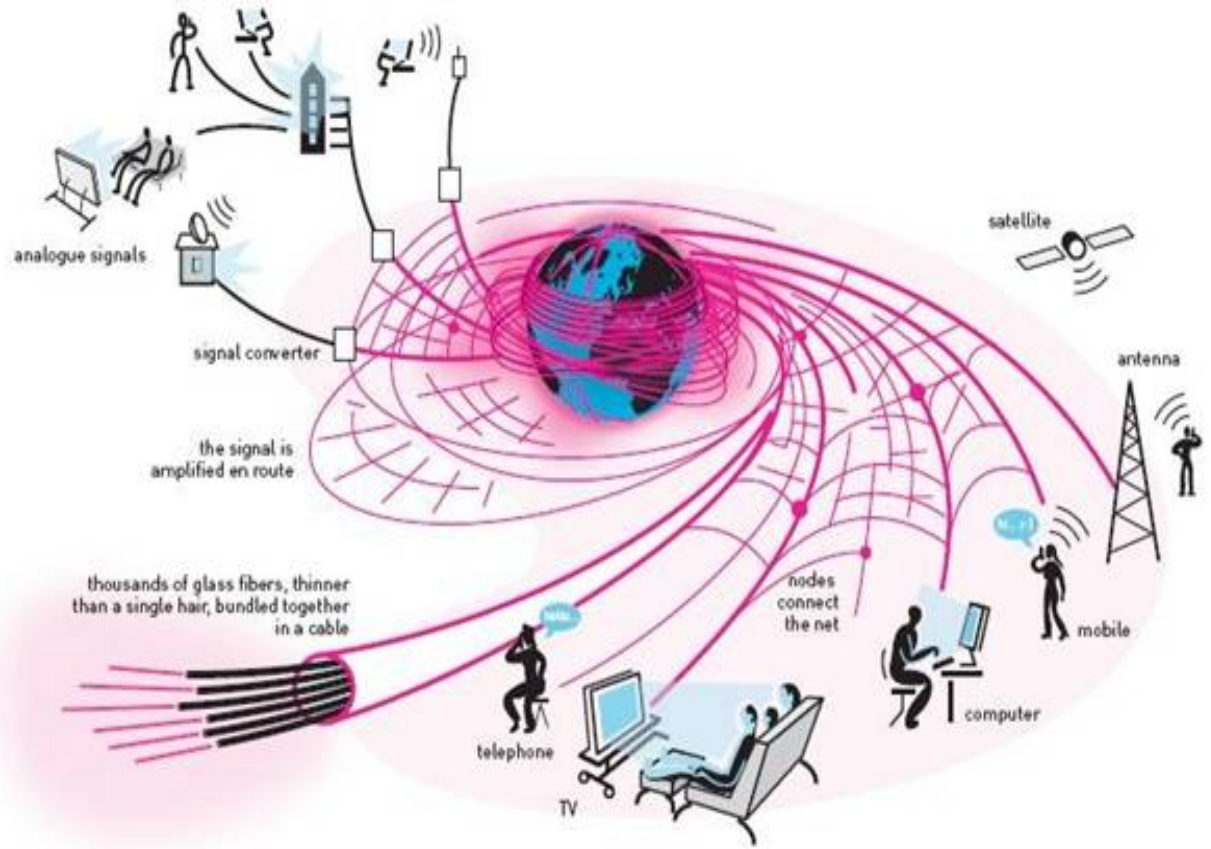
Windowing

- 交易表：transaction(account-number, date-time, value), 取款为负，存款为正。求每笔交易后，每个账号的余额是多少？

```
select account-number, date-time,
    sum(value) over
        (partition by account-number
        order by date-time
        rows unbounded preceding)
    as balance
from transaction
order by account-number, date-time
```



谢 谢！


analogue signals
signal converter
the signal is
amplified en route
thousands of glass fibers, thinner
than a single hair, bundled together
in a cable
telephone
TV
nodes
connect
the net
computer
mobile
antenna
satellite