



# Chapter 5

数据表的物理实现

Physical implementation

# 理解物理实现

- 本部分将探讨表中数据不同的存储方式，以及该存储方式有利于哪类操作
- 表（table）是高级的数据容器，背后使用的技术是什么？
- 数据物理布局的优化也同样是针对设计而言的

# 物理结构的类型

- 物理结构和SQL无关，但是使用SQL的好坏却受到底层结构的影响。
  - 固定型
  - 进化型
- 设计是最关键的
- 具体数据库的实现差别很大

# 冲突的目标

- 并发用户数很大的系统
  - 尽量以紧凑的方式存储数据
  - 尽量将数据分散存储
- 没有并发的修改密集型（change-heavy）
  - 数据查询要快
  - 数据更新也要快...
- DBMS所处理的基本单元（页、块）通常不可分割
- 总结：读写不会和睦相处，怎么和谐啊～～

# 把索引当成数据仓库

- 当索引中增加额外的字段（一个或多个，它们本身与实际搜索条件无关，但包含查询所需的数据），能提高某个频繁运行的查询的速度。
- 尽量在索引中多存储数据的极限是？--允许在主键索引中存储表中所有数据，表就是索引
  - Oracle：“索引组织表（index-organized table, IOT）”
- 对IOT表插入的效率也许低于堆文件
- IOT的用途：全索引表，代码查找表，高频度的一组关联数据查询

# 记录强制排序

- IOT最大的优点：记录是排序的...（效率惊人）
- 记录有序的实现：多数数据库使用索引定义记录顺序
  - SQL Server, Sybase：聚簇索引（clustered index）
  - DB2：聚簇索引（clustering index）
- 记住一点：任何有序数据便于某些处理的同时，必将对其他处理不利
  - 表变成了树状结构.....这是失传已久的“层次型数据库”

# 数据自动分组


- 分区 (partition) 也是一种数据分组的方式
  - 提高并发性 (concurrency) 和并行性 (parallelism)
  - 从而增强系统架构的可伸缩性 (scalable)
- 循环分区：不受数据影响的内部机制
  - 分区定义为各个磁盘的存储区域
  - 可以看作是随意散布数据的机制
  - 保持更改带来的磁盘I/O操作的平衡

# 数据自动分组 (cont')

- 数据驱动分区
  - 根据一个或多个字段中的值来定义分区
  - 手工分区，一般叫分区视图 (partitioned view) ，而MySQL称为 (merge table)
- 分区的实现方式
  - 哈希分区 (Hash-partitioning)
  - 范围分区 (Range-partitioning)
  - 列表分区 (List-partitioning)



# 分区是把双刃剑

- 分区能解决并发问题吗?
- 又回到了IOT类似的问题：“冲突”
  - A. 通过分区键将数据聚集，利于高速检索；
  - B. 对并发执行的更改操作，分散的数据可以避免访问过于集中的问题
- So, A or B.....完全取决于您的需求

# 分区与数据分布

- 表非常大，且希望避免并发写入数据的冲突就一定要用分区吗？
- 例如客户订单明细表.....
- 对分区表进行查询，当数据按分区键均匀分布时，收益最大

# 数据分区的最佳方法

- 整体改善业务处理的操作，才是选择非缺省的存储选项的目标
- 更新分区键会引起移动数据，似乎应该避免这么做
  - 例如实现服务队列，类型 ( $T_1 \dots T_n$ ) 状态 ( $\{W|P|D\}$ )
  - 按请求类型分区：进程的等待降低
  - 按状态分区：轮询的开销降低
  - 取决于：服务器进程的数量、轮询频率、数据的相对流量、各类型请求的处理时间、已完成请求的移除频率
- 对表分区有很多方法，显而易见的分区未必有效，一定要整体考虑

# 预连接表

- 如何把至少两个表的数据分组到某一物理区域?
- 预链接表（pre-joining table）：通过common join key为基础连接键，在基本单元上存放两个或多个表的数据
- 这是改进查询的一种专门技术，几乎会影响所有其他数据库操作——*让我们忘掉有这个技术存在吧*

# Holy Simplicity

- 除了堆文件之外的任何存储方法，都会带来复杂性
- 选错存储方式会带来大幅度的性能降低
- 总结
  - A. 测试，测试，测试
  - B. 设计是最重要的
  - C. 任何设计都有时效性