作者：代兴　邮箱：503268771@qq.com　Vx 公众号：东方架构师
https://github.com/drink-cat/Book_Program-Principles

索引的含义

索引应用在软件的方方面面，组织数据、高效查询，非常重要。
比如，内存的页表使用前缀树和数组，实现虚拟内存映射到物理内存。
比如，文件的 pagecache 使用基数树，实现文件偏移映射到内存 page。
比如，关系型数据库使用 B+树，实现主键索引、普通索引、聚合索引。
比如，搜索引擎使用倒排索引，实现用关键词查找所属的全部文档。

索引的含义：使用有排序功能的集合结构来组织数据，提供高效的单值查询、范围查询。

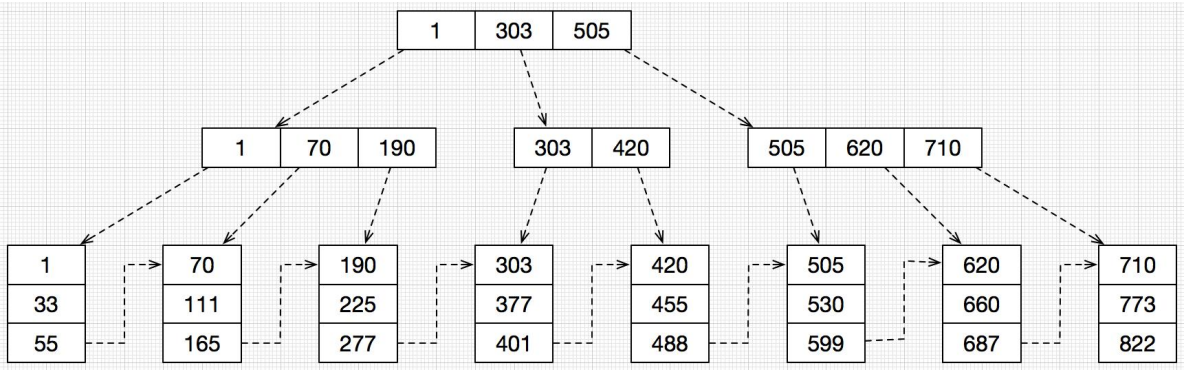
索引的特点：
依赖排序功能。有序的数据利于算法使用，常用的有序结构包括 B+树、红黑树、跳表、基数树等，追求 O(lgN)/O(NlgN)的读写效率。
支持高效查询。索引的重要作用是查询，使用分治算法、二分查找算法等优化查询性能。
支持永久存储。某些索引不使用过期时间，比如 mysql 表的索引被持久化为存储索引的磁盘文件。如果表太大，可以使用分表功能，把数据分散到多个表。
单值模式或 KV 模式。索引可以只存储 value，比如把热卖的商品 ID 存储到 redis。索引可以存储 key-value，比如 pagecache 用 key 存文件偏移，用 value 存内存 page。KV 模式更常见。
索引自动平衡。为了保证查询效率，索引需要动态调整自身结构，达到平衡。比如红黑树的左旋、右旋，B+树的节点分裂、合并。

mysql 的 B+树索引

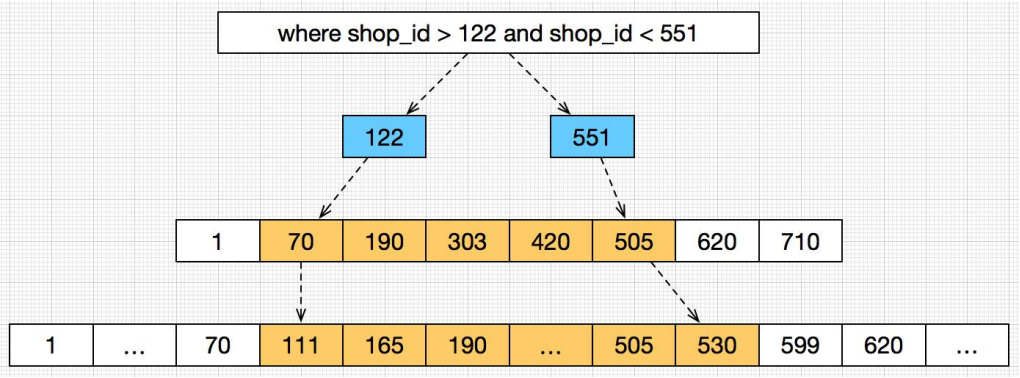
B+树索引是 mysql 等关系型数据库的核心组件，属于非常经典的索引。

B+树索引的特点：
多叉树，出度大。树型结构，一个节点可以有很多个子节点，1 个节点的数据存储在一个磁盘块，存储结构紧凑，便于把整块数据加载到内存。
树的层次少，一般不超过 4 层。优点为减少磁盘 IO 次数，每层处理一次可以减少总的处理次数。
节点的数据有序，便于高效二分查找，时间复杂度近似 O(logN)。
叶子节点左右相连，便于顺序遍历，实现范围查询。找到下界的数据，再找到上界的数据，中间的很多数据，可以依次遍历出来。
自顶向下查找。查找过程从上往下，依次查找每层的节点。

B+树索引的示意图：



简化分析：B+树索引有序，近似于有序数组。B+树索引的范围查询，近似于在有序数组中范围查询。



B+树索引的重要作用为缩小查询范围，进而减少读表次数、减少扫描行数、减少计算次数，从而降低 CPU 和内存的负载，缩小 SQL 执行时间。
这些 SQL 既包括 select 语句，也包括 update 语句、delete 语句。索引的一个重要优化点是使用小范围查询，把大范围查询改为多个小范围查询。

使用 1 个索引来分析，扫描的行数越少，查询耗时越小。
假设，商品表有 3000 万行，字段 shop_id 有索引，表结构如下。

```
CREATE TABLE `product` (
  `id` bigint(8) NOT NULL AUTO_INCREMENT,
  `name` varchar(200) NOT NULL DEFAULT '',
  `status` int(4) NOT NULL,
  `shop_id` bigint(8) NOT NULL,
  `price` double NOT NULL,
  `description` varchar(1500) NOT NULL DEFAULT '',
  `create_time` datetime DEFAULT NULL,
  `update_time` datetime DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `inx_shopid` (`shop_id`)
) ENGINE=InnoDB AUTO_INCREMENT=30000001 DEFAULT CHARSET=utf8mb4 COMMENT='商品表';
```

情况 1：shop_id 范围 300 至 700，预计扫描 1499 万行。

```
explain
```

```
select * from product where  shop_id between 300 and 700
and status in ( 2,3 ) order by price desc limit 5 ;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	product	range	inx_shopid	inx_shopid	8	NULL	14992767	Using where; Using filesort

情况 2：shop_id 范围 300 至 400，预计扫描 673 万行。

```
explain
select * from product where  shop_id between 300 and 400
and status in ( 2,3 ) order by price desc limit 5 ;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	product	range	inx_shopid	inx_shopid	8	NULL	6739604	Using where; Using filesort

情况 3：shop_id 范围 300 至 360，预计扫描 389 万行。

```
explain
select * from product where  shop_id between 300 and 360
and status in ( 2,3 ) order by price desc limit 5 ;
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	product	range	inx_shopid	inx_shopid	8	NULL	3899464	Using where; Using filesort

用 Java 生成 2D 地图的点线索引

编写代码： Point2d.java

```
package book.index.map2d;

public class Point2d implements Comparable<Point2d> {
    int valueX;// X 值
    int valueY;// Y 值

    public static Point2d fromXY(int x, int y) {
        Point2d point2d = new Point2d();
        point2d.valueX = x;
        point2d.valueY = y;
        return point2d;
    }

    // 水平线与垂直线相交，生成点
    public static Point2d fromLine(Line2d horizonLine, Line2d verticalLine) {
        Point2d point2d = new Point2d();
        point2d.valueX = verticalLine.valueX;
        point2d.valueY = horizonLine.valueY;
        return point2d;
    }
```

```
    // 点排序
    @Override
    public int compareTo(Point2d tmp) {
        int ret = Integer.compare(this.valueX, tmp.valueX);
        if (0 != ret) {
            return ret;
        }
        return Integer.compare(this.valueY, tmp.valueY);
    }

    @Override
    public String toString() {
        return "( " + valueX + " , " + valueY + " )";
    }
}
```

编写代码： Line2d.java

```
package book.index.map2d;

import java.util.TreeSet;

public class Line2d implements Comparable<Line2d> {
    boolean beHorizon;// 水平线或垂直线
    int valueX;// X 值
    int valueY;// Y 值
    TreeSet<Point2d> pointList = new TreeSet<>();// 点

    // 垂直线
    public static Line2d newVertical(int x) {
        Line2d line = new Line2d();
        line.valueX = x;
        line.valueY = 0;
        line.beHorizon = false;
        return line;
    }

    // 水平线
    public static Line2d newHorizon(int y) {
        Line2d line = new Line2d();
        line.valueX = 0;
        line.valueY = y;
        line.beHorizon = true;
        return line;
```

```
    }

    @Override
    public int compareTo(Line2d tmp) {
        if (beHorizon) {
            // 2 个水平线，只需要比较 Y 值
            return Integer.compare(this.valueY, tmp.valueY);
        } else {
            // 2 个垂直线，只需要比较 X 值
            return Integer.compare(this.valueX, tmp.valueX);
        }
    }
}
```

编写代码： Map2dIndex.java

```
package book.index.map2d;

import java.util.NavigableSet;
import java.util.TreeSet;

public class Map2dIndex {
    int intervalLine = 5;// 线的间隔
    TreeSet<Line2d> lineHorizonList = new TreeSet<>();// 水平线
    TreeSet<Line2d> lineVerticalList = new TreeSet<>();// 垂直线
    int pointCount = 0;// 点的数量

    // 用边界生成地图索引
    public static Map2dIndex createIndex(int minX, int maxX, int minY, int maxY, int intervalLine)
{
        long millis = System.currentTimeMillis();
        Map2dIndex index = new Map2dIndex();
        index.intervalLine = intervalLine;
        index.buildLine(minX, maxX, minY, maxY);// 生成直线
        index.buildPoint();// 生成点
        long costMillis = System.currentTimeMillis() - millis;
        StringBuilder buf = new StringBuilder(300).append("createIndex").append("
costMillis=").append(costMillis)
            .append("        lineHorizonList=").append(index.lineHorizonList.size()).append("
lineVerticalList=")
            .append(index.lineVerticalList.size()).append("
pointCount=").append(index.pointCount);
        System.out.println(buf.toString());
        return index;
    }
```

```
    // 生成直线
    private void buildLine(int minX, int maxX, int minY, int maxY) {
        // 全部的垂直线
        for (int tmpX = minX; tmpX <= maxX; tmpX = tmpX + intervalLine) {
            lineVerticalList.add(Line2d.newVertical(tmpX));
        }
        // 全部的水平线
        for (int tmpY = minY; tmpY <= maxY; tmpY = tmpY + intervalLine) {
            lineHorizonList.add(Line2d.newHorizon(tmpY));
        }
    }

    // 生成点
    private void buildPoint() {
        // 水平线与垂直线相交，生成点
        for (Line2d horizonLine : lineHorizonList) {
            for (Line2d verticalLine : lineVerticalList) {
                Point2d point2d = Point2d.fromLine(horizonLine, verticalLine);
                ++pointCount;
                // 点与直线的关系
                horizonLine.pointList.add(point2d);
                verticalLine.pointList.add(point2d);
            }
        }
    }

    // 范围查询。查询点列表。
    public TreeSet<Point2d> queryPoint(int minX, int maxX, int minY, int maxY) {
        long millis = System.currentTimeMillis();
        // 范围查询
        NavigableSet<Line2d> subHorizon = lineHorizonList.subSet(Line2d.newHorizon(minY), true,
Line2d.newHorizon(maxY), true);
        TreeSet<Point2d> pointList = new TreeSet<>();// 点
        for (Line2d horizonLine : subHorizon) {
            // 范围查询
            Point2d leftPoint = Point2d.fromXY(minX, horizonLine.valueY);
            Point2d rightPoint = Point2d.fromXY(maxX, horizonLine.valueY);
            NavigableSet<Point2d> subPoint = horizonLine.pointList.subSet(leftPoint, true,
rightPoint, true);
            pointList.addAll(subPoint);
        }
        long costMillis = System.currentTimeMillis() - millis;
        StringBuilder buf = new StringBuilder(300).append("queryPoint ").append("
```

```
costMillis=").append(costMillis)
            .append("                          pointCount=").append(pointList.size()).append("
rangeX=").append(minX + "~" + maxX)
            .append("  rangeY=").append(minY + "~" + maxY);
        System.out.println(buf.toString());
        return pointList;
    }
}
```

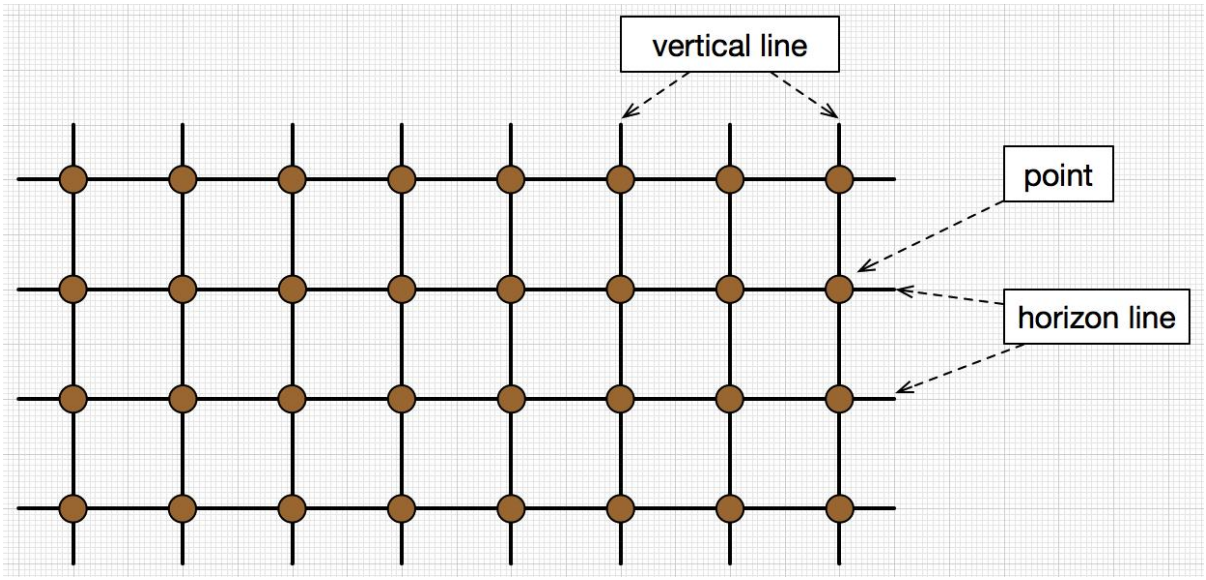
编写代码： Map2dIndexMain.java

```
package book.index.map2d;

import java.util.TreeSet;

public class Map2dIndexMain {
    public static void main(String[] args) {
        // 构建索引
        Map2dIndex index = Map2dIndex.createIndex(100, 5000, 500, 6000, 3);

        // 查询索引
        TreeSet<Point2d> pointList1 = index.queryPoint(-33, 2511, -66, 3722);
        TreeSet<Point2d> pointList2 = index.queryPoint(1511, 2566, 2753, 3787);
        TreeSet<Point2d> pointList3 = index.queryPoint(1001, 1055, 777, 888);

        System.out.println("===");
    }
}
```

运行代码：

```
createIndex  costMillis=7281  lineHorizonList=1834  lineVerticalList=1634
pointCount=2996756
queryPoint   costMillis=424  pointCount=864300  rangeX=-33~2511  rangeY=-66~3722
queryPoint   costMillis=28  pointCount=121440  rangeX=1511~2566  rangeY=2753~3787
queryPoint   costMillis=0  pointCount=666  rangeX=1001~1055  rangeY=777~888
```

示意图：



代码逻辑：
Point2d 表示点，由水平线、垂直线交叉生成。点支持排序，先使用 X 升序，再使用 Y 升序。
Line2d 表示直线，分为水平线、垂直线。直线包含很多点，使用 TreeSet 保存有序的点。直线支持排序，水平线使用 Y 升序，垂直线使用 X 升序。
Map2dIndex 表示地图索引，支持生成索引、查询索引。地图索引包含很多水平线、垂直线，使用 TreeSet 保存有序的直线。
Map2dIndexMain 表示查看效果，首先生成索引，然后查询索引。

分析结果：
生成很大范围的地图索引，耗时 7281 毫秒，包含 1834 个水平线、1634 个垂直线、2996756 个点。
使用索引查询点，耗时依次为 424 毫秒、28 毫秒、0 毫秒，返回的点的数量依次为 864300 个、121440 个、666 个。
生成索引，耗时长。
查询索引，耗时短。随着查询范围缩小，耗时越来越短。
地图索引使用红黑树保存直线、点，所以范围查询效率高。

引申思考：
地图索引支持单值查询、范围查询，经过改造也可以支持交集、并集、差集等功能。
地图索引给几何建模，应用于使用 2D 几何、3D 几何的场景，比如地图测绘、地图导航、自动驾驶、3D 建筑设计。