14-数据库

刘亚雄

极客时间-Java 讲师

四、MySQL索引篇

4.1 MySQL索引简介

01-什么是索引？

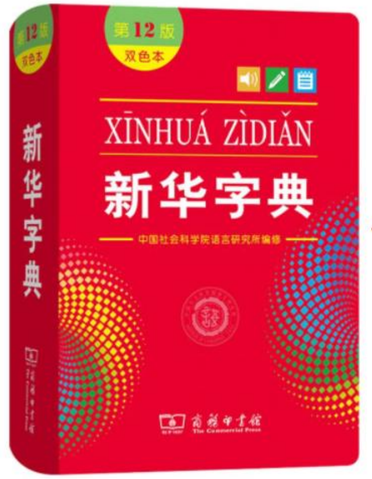
- 索引是数据库 **高效获取数据**的**数据结构，**加快查询速度，索引一般存储在表空间中，也就是磁盘里
- 常见索引：聚簇索引，辅助索引，组合索引，唯一性索引。。。
- 没有特别说明，索引一般特指B+树组织结构

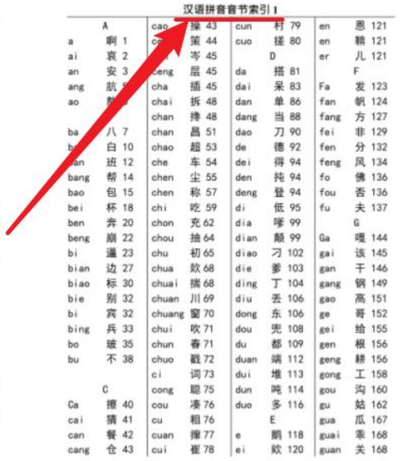
02-优势与劣势

- 优势：**两降一升**，降低磁盘IO频次、降低数据排序的成本，提高数据检索效率
  - 被索引列会自动排序【B+树叶子节点的有序特性】
  - 如果Order By的字段使用索引，效率会高特别多
- 劣势：占用更多磁盘空间（空间换时间），降低更新效率

03-不用索引行不行？

- 完全可以，时间复杂度相当于O(n)
- 选择权在谁手里？



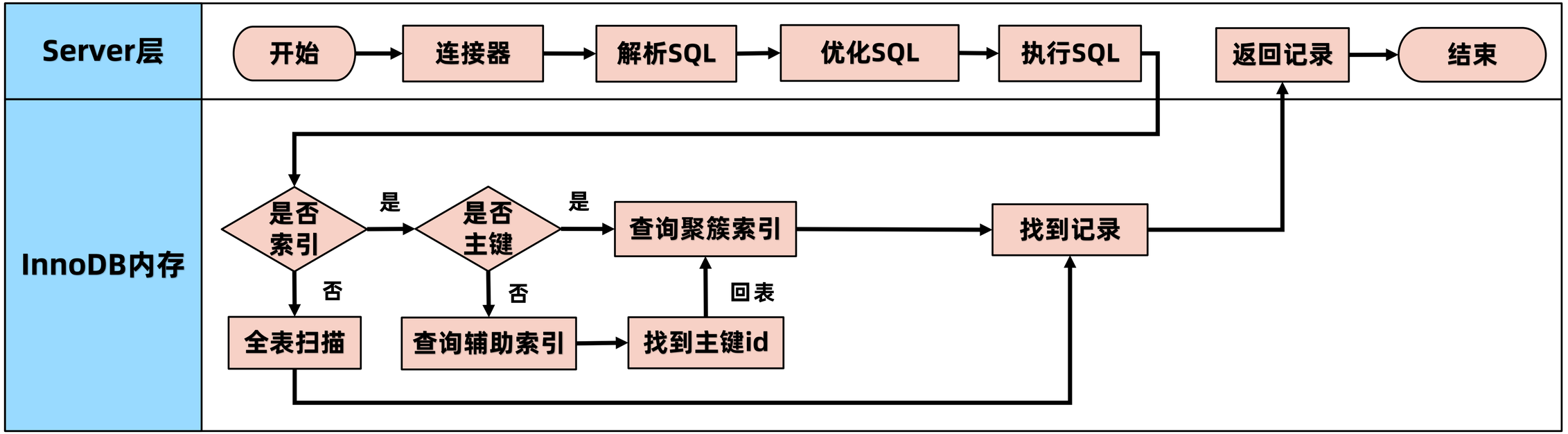


MySQL查询语句怎么用索引？

4.2 一条Select语句的执行流程

一条查询语句

```
1 select * from city WHERE city_id=1
```



4.3 索引基本使用

01-索引类型，按照数量分类

- 单列索引：索引中只有一个列
- 组合索引：使用2个以上的字段创建的索引

02-单列索引按类型细分

- 主键索引：必须唯一Unique Key，非空Not Null

```
ALTER TABLE table_name ADD PRIMARY KEY (column_name);
```

- 唯一性索引：必须唯一Unique Key，可以为空Null

```
CREATE UNIQUE INDEX index_name ON table(column_name);
```

- 普通索引：可以不唯一，可以为空Null

```
ALTER TABLE table_name ADD INDEX index_name (column_name);
```

- 全文索引：支持全文搜索的索引，只能对文本类型字段设置，不建议使用

```
ALTER TABLE `t_fulltext` ADD FULLTEXT INDEX `idx_content`(`content`);
```

- 空间索引：5.7版支持空间索引，而且支持OpenGIS几何数据模型
- 前缀索引：用字段的一部分建立索引

```
ALTER TABLE table_name ADD INDEX index_name (column1(length));
```

4.3 索引基本使用

03-组合索引

- **建议使用组合索引替代单列索引，为什么呢？**
- 最左前缀原则

```
ALTER TABLE table_name ADD INDEX index_name (column1,column2);
```

04-删除索引

```
DROP INDEX index_name ON table
```

05-查看索引

```
SHOW INDEX FROM table_name
```

4.4 索引的数据结构

01-使用索引的基本需求

至少需要支持两种最常用的查询需求：

- 等值查询：根据某个值查找数据
- 范围查询：根据某个范围区间查找数据
- 排序Order By
- 分组Group By

性价比：时间与空间

- 执行时间方面：查询时间尽可能短
- 存储空间方面：占用磁盘存储空间尽可能小

02-可选的数据结构

Hash表，二叉树，平衡二叉查找树（红黑树是一个近似平衡二叉树），B树，B+树

应该使用什么数据结构？

4.5 存储引擎的索引实现

01-MyISAM索引

MyISAM数据文件和索引文件分开存储，索引B+Tree数据结构，其中叶子节点Key为索引列值，数据为所在行的磁盘地址

表索引存储在索引文件tablename.MYI中，数据文件存储在数据文件tablename.MYD中

- **主键索引：** MyISAM查询时会将索引节点缓存在MySQL缓存中，而数据的缓存依赖于操作OS Cache
- **辅助索引：**
  - 主键索引必须唯一，辅助索引可以重复
  - 由于辅助索引重复了，所以即便是等值查询，也需要按照范围查询的方式在辅助索引树上查询数据

4.5 存储引擎的索引实现

02-InnoDB索引

每个InnoDB表都有一个**聚簇索引**，也叫聚集索引。除了聚簇索引外的其他索引都叫辅助索引

聚簇索引是B+Tree数据结构，叶子节点存储数据行，非叶子节点存储主键值

在查询时，InnoDB使用主键值在聚簇索引中搜索行记录

一般情况下主键索引就是聚簇索引，但也存在没有主键的情况，没有主键会采用ROWID构建聚簇索引

InnoDB的表数据和索引默认存储在一个文件tablename.ibd中

- **主键索引：**
  - InnoDB要求表**必须有主键索引**
  - 主键索引**叶子节点存储数据行**，**辅助索引只会存储主键值**
  - 底层叶子节点按照顺序排序
- **辅助索引：**
  - InnoDB的辅助索引只会存储主键值而非磁盘地址
  - 除聚簇索引之外的所有索引都称为辅助索引
  - 辅助索引查询记录必然经过主键索引：首先查辅助索引获取主键，根据主键在主键索引查询获得记录
  - 叶子节点按顺序排序

4.5 存储引擎的索引实现

02-InnoDB索引

- 2.1 组合索引：详情请查阅课程笔记
  - 存储结构
  - 查找方式
  - 最左前缀匹配原则
  - 组合索引的创建原则
  - 组合索引心法口诀

4.5 存储引擎的索引实现

02-InnoDB索引

➢ 2.2 什么是覆盖索引？

- 关于回表查询，思考一个问题：**回表必然每次都会查聚簇索引树，是不是使用辅助索引必然需要回表？**
- Select中的列数据如果直接在辅助索引树上可以全部获取，MySQL就不会白费力气**回表查询**
- 也就是说索引树已经**覆盖**了查询需求，这种现象称之为**覆盖索引**
- 覆盖索引可以减少磁盘IO次数，显著提升查询性能，是一种很常见的**辅助索引**查询优化手段

➢ 2.3 什么是索引下推？

- 索引条件下推： Index Condition Pushdown简称**ICP**，是**组合索引**查询优化的手段
- ICP可以减少存储引擎回表查询，降低磁盘IO，提升查询性能
- 主要作用于组合索引中不满于与最左前缀的索引条件
- 通过optimizer_switch参数控制开始和关闭

4.6 创建索引的原则

01-什么情况下需要创建索引

- 频繁出现在where 条件，order排序，group by的字段
- Select 频繁查询的列
- 多表join关联查询on两边的字段

使用索引应该注意什么问题呢？

4.6 创建索引的原则

02-索引优化建议

- **表记录很少不需要创建索引**
- **单表索引的个数不能太多，尤其是单列索引：**
  - 空间：浪费空间，每个索引都是个索引树，占据磁盘空间
  - 时间：更新变慢，1.需要更新索引树，2.太多索引也会增加优化器的选择时间
- **频繁更新的资源，不建议作为索引：**频繁更新的字段引发频繁的**页分裂**和**页合并**，性能消耗比较高
- **区分度不高的字段，不建议作为索引：**如：性别，布尔，状态
  - 区分度太低索引会导致扫描行数过多，再加上回表查询的消耗，比全表扫描的性能还要差。
  - 这些字段一般会用在组合索引中
- **无序的字段，不建议作为索引：**如：身份证，UUID等
  - 更新数据时，会出现频繁的页分裂，页内数据不紧凑浪费存储空间
- **主键索引建议使用自增的长整型，避免使用很长的字段：**主键字段越长，索引页和辅助索引的叶子节点可存储的数据量就越少，查询时磁盘IO次数会增多，降低查询效率。
- **尽量创建组合索引，而不是单列索引：**
  - 优点：一个顶多个，节省空间，可以使用覆盖索引和ICP
  - **创建组合索引原则：**组合索引把频繁使用的列、区分度高的列放在前面。**频繁使用代表利用率高**，**区分度高代表筛选粒度大**，这样做可最大限度利用索引价值，缩小筛选范围

案例：索引失效分析

THANKS

教育不是注满一桶水，而是点燃一把火